scientific reports

Check for updates

OPEN

CLM-former for enhancing multi-horizon time series forecasting and load prediction in smart microgrids using a robust transformer-based model

S. Mozhgan Rahmatinia, Seyed-Majid Hosseini & Seyed-Amin Hosseini-Seno✉

Accurate multi-horizon load forecasting is essential for the stability and efficiency of smart grid operations, particularly in residential environments where electricity consumption patterns are shaped by both long-term trends and short-term fluctuations. Transformer-based models such as Autoformer have advanced forecasting accuracy by leveraging frequency-domain attention to capture periodic behavior. However, they often struggle with rapidly changing, localized patterns prevalent in real-world data. To address this challenge, we propose CLM-Former, a novel hybrid deep learning architecture that integrates time series decomposition, an autocorrelation-based attention mechanism, and a tailored subnetwork, CLM-subNet, which combines convolutional and recurrent layers. This design enables the model to effectively capture both seasonal dependencies and high-resolution variations in electricity usage, thereby enhancing its performance across multiple forecasting horizons. Comprehensive evaluations on real-world smart meter data demonstrate the robustness and adaptability of CLM-Former against a range of Transformer-based and deep learning baselines. By effectively modeling both long-term periodic trends and short-term dynamics, CLM-Former emerges as a promising tool for residential energy forecasting. Its robust performance offers valuable implications for demand response, distributed scheduling, and the future management of smart grids.

Keywords Smart grid, Load forecasting, Multihorizon time series prediction, Deep learning, Transformer, Autoformer

Accurate load forecasting is a cornerstone of efficient and reliable smart grid (SG) operations. By providing precise estimates of future electricity demand, load forecasting enables better strategic planning, resource allocation, and cost control across power systems. It plays a key role in a wide range of smart grid applications, including demand response, load balancing, cost optimization, theft detection, and anomaly detection in smart meters (SMs)[1,2]. Moreover, accurate forecasts contribute to reducing power generation costs and supporting grid stability and sustainability through the integration of renewable energy sources[3]. Conversely, poor forecasting can lead to grid imbalances, increased operational costs, and unanticipated outages[4].

In recent years, the rapid deployment of smart grid technologies and smart metering infrastructures has resulted in large volumes of granular consumption data. These data streams often exhibit nonlinear, nonstationary, and seasonally fluctuating patterns. As a result, traditional statistical models such as autoregressive integrated moving average (ARIMA) have shown limited effectiveness in such environments[5,6]. Despite their interpretability and widespread use, these models require extensive parameter tuning and are not well-suited to capturing complex, time-varying dynamics[7].

To address these shortcomings, machine learning techniques such as support vector regression (SVR) have been introduced. SVR offers improved flexibility over traditional models and has been applied to various time series forecasting tasks. However, its effectiveness is often hindered by sensitivity to kernel selection, challenges in parameter optimization, and difficulties in capturing temporal dependencies in large-scale datasets.

Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran. ✉email: hosseini@um.ac.ir

Deep learning has driven major advances in modeling complex, high-dimensional time series[8,9]. In particular, recurrent neural networks (RNNs)—including long short-term memory (LSTM)[10] and gated recurrent units (GRU)[11]—have shown strong ability to capture long-term temporal dependencies in load forecasting. The integration of convolutional neural networks (CNNs) with RNNs has further enhanced the ability of these models to extract local patterns while preserving sequential information. Despite their effectiveness, these models often struggle with issues such as vanishing gradients, limited parallelization, and reduced performance in very long-range forecasting.

Transformer architecture, originally developed for natural language processing[12], has recently been adopted in time series forecasting due to its ability to capture global dependencies using self-attention mechanisms. Unlike RNNs, Transformers enable parallel training and are not constrained by sequence length, which makes them attractive for modeling complex temporal relationships in electricity demand data.

Nevertheless, vanilla Transformers suffer from high computational complexity ($O(L2)$), where L is the sequence length). To address this, various Transformer variants have been proposed to improve efficiency. For example, Reformer[13] employs locality-sensitive hashing to reduce memory usage, LogTrans[14] uses log-sparse attention to retain key dependencies, and Informer[15] introduces ProbSparse attention to filter out low-impact tokens. While these models significantly reduce computational demands, they often sacrifice prediction accuracy, particularly when modeling fine-grained short-term fluctuations.

To further improve forecasting accuracy, Autoformer[16] replaces the traditional attention mechanism with an autocorrelation-based module that operates in the frequency domain. This allows the model to better capture long-term periodic patterns while maintaining a lower time complexity of $O(LlogL)$. Autoformer has shown superior performance on seasonal time series, making it a promising solution for medium- and long-term forecasting tasks. However, its reliance on periodicity limits its responsiveness to short-term variability and abrupt local changes—characteristics commonly found in residential electricity consumption data.

Given these limitations, there is a growing need for forecasting models that not only preserve the benefits of frequency-domain attention mechanisms but also effectively capture localized and short-term patterns. In this study, we propose a novel forecasting architecture, **CLM-Former**, which addresses the weaknesses of previous Transformer-based models by integrating temporal decomposition and a hybrid learning structure. Specifically, our model incorporates a convolutional–LSTM subnetwork (CLM-subNet) designed to enhance short-term pattern recognition while preserving long-term seasonal trends. The conceptual overview of CLM-Former is illustrated in Fig. 1. As shown, the input time series is decomposed into seasonal and trend components in every layer. The seasonal path is refined by the proposed CLM-subNet to capture short-term dynamics, while the trend path undergoes lightweight processing. This targeted enhancement enables CLM-Former to effectively

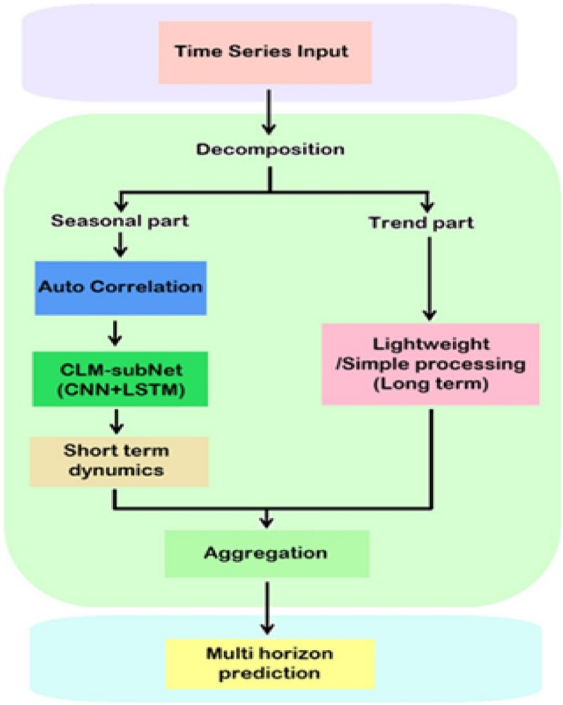


Fig. 1. Conceptual overview of CLM-Former. The input time series is decomposed (in every encoder/decoder layer) into a seasonal component (processed via Auto-Correlation for long-term periodicity and refined by the proposed CLM-subNet — CNN + LSTM — for short-term and localized dynamics) and a trend component (lightweight processing). Outputs from both paths are aggregated to produce the final multi-horizon prediction. This design allows CLM-Former to outperform pure decomposition-based Transformers by explicitly addressing short-term fluctuations while preserving efficiency.

model both long-term periodicity and localized fluctuations (see Sect. "Architectural overview" for the detailed architecture in Fig. 3).

Key contributions and novelty

The proposed CLM-Former introduces several distinctive innovations that differentiate it from existing hybrid CNN-LSTM-Transformer frameworks. The primary contributions of this work are:

- **Architectural innovation:** Instead of appending processing blocks, CLM-Former performs a targeted architectural modification by replacing the Feed-Forward Network (FFN) inside each Autoformer block with our dedicated CLM-subNet. This subnetwork (1D-CNN → LSTM) is engineered to process only the seasonal component. Crucially, it introduces a "bottleneck" design where the LSTM layer compresses the high-dimensional features (from d_{model} to a compact hidden state), allowing for robust temporal modeling while maintaining parameter efficiency comparable to, or even lower than, the standard convolutional baselines.
- **Functional innovation:** The CLM-subNet facilitates hierarchical learning on the seasonal data. The CNN layers first capture high-frequency, local variations, and the LSTM layers then model the sequential dependencies among these extracted features. This dual mechanism provides a much richer temporal representation than the simple point-wise transformations used in the original Autoformer.
- **Conceptual integration:** Our model creates a unique cross-domain synergy. It preserves Autoformer's powerful autocorrelation mechanism for identifying long-term periodicity in the frequency domain, while simultaneously introducing explicit time-domain analysis via the CLM-subNet to enhance sensitivity to local, aperiodic fluctuations.
- **Empirical contribution:** Through comprehensive experiments on real-world smart microgrid data, we demonstrate that CLM-Former consistently outperforms strong baselines, including Autoformer, Crossformer, and other deep learning models. Furthermore, our results confirm that this superior accuracy is achieved without compromising computational efficiency, as the model retains the $O(LlogL)$ complexity and exhibits inference speeds comparable to the official Autoformer implementation.

The remainder of this paper is organized as follows. Section "Related works" presents a detailed review of related work. Section "Problem definition" introduces the problem formulation. Section "Results and discussion" describes the architecture of the proposed model. Experimental results and performance evaluations are discussed in Sect. "Discussion of results". Finally, Sect. "Limitations and future work" summarizes the findings and outlines future research directions.

Related works

This section provides a detailed review of the state-of-the-art in time series forecasting for load prediction, categorized by the underlying architectural approach. We focus on the evolution from recurrent models to the diverse family of Transformer-based and graph-based architectures, highlighting the specific limitations that motivate the design of our proposed CLM-Former.

Recurrent and convolutional models: Early deep learning efforts in load forecasting were dominated by Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM)[10] and Gated Recurrent Unit (GRU)[11] networks. Their strength lies in their memory-gated mechanism, which is designed to capture temporal dependencies from sequential data. Many hybrid models combine CNNs with recurrent architectures to improve local feature extraction[17]. CNN-LSTM hybrids, for example, first extract short-term patterns via convolution, then model long-term dependencies with LSTM[18]. Other notable works include LSTNet[19], which combines CNN and RNN layers with an autoregressive component to model both short-term local dependencies and long-term periodic patterns. Despite their success, these models are inherently sequential, which limits parallelization and can lead to challenges in capturing extremely long-range dependencies due to information decay over time[20].

Transformer-Based Architectures: The Transformer architecture[12] introduced a paradigm shift by replacing recurrence with a self-attention mechanism, enabling the model to capture global dependencies in parallel. This led to its widespread adoption in time series forecasting. However, the vanilla Transformer's quadratic computational complexity $O(L^2)$ posed a significant bottleneck for long sequences. Consequently, research has focused on developing more efficient Transformer variants, which can be broadly grouped as follows.

- *Efficiency through Sparse Attention:* A major line of research focused on reducing complexity by making the self-attention mechanism sparse. Informer[15] introduced a *ProbSparse* attention mechanism to select only the most significant queries. Reformer[13] employed locality-sensitive hashing (LSH) to cluster similar queries, while LogTrans[14] used a logarithmic pattern to attend to tokens at different scales. While these methods successfully reduce complexity to near-linear ($O(LlogL)$), their sparsity can come at the cost of information loss, as critical but non-dominant tokens may be overlooked. For a conceptual overview of these different attention strategies, a visual comparison is provided in Appendix A (Fig. 12).
- *Frequency-domain approaches:* A different approach to achieving efficiency and capturing periodicity was to operate in the frequency domain. Autoformer[16] pioneered this by replacing self-attention with an *Autocorrelation* mechanism, which measures sequence similarity based on their periodic patterns. This design proved highly effective for time series with strong seasonality. Building on this, FEDformer[21] introduced a frequency-enhanced block with Fourier and wavelet transforms. However, their strong focus on periodicity limits responsiveness to aperiodic short-term fluctuations and abrupt changes — common in residential load data. Additionally, recent studies have demonstrated the efficacy of combining attention mechanisms with frequency-domain preprocessing. For instance, Rahmatinia[22] proposed an attention-based deep learning model

incorporating Fast Fourier Transform (FFT) for simultaneous multi-horizon prediction of load, price, and wind power, showing improved accuracy over traditional RNNs. Building on these insights, CLM-Former extends the frequency-domain concept by integrating a learnable Auto-Correlation mechanism and specialized sub-networks for enhanced feature extraction.

- *Hybrid and multi-scale designs:* To address the limitations of both sparse and frequency-based models, more recent works have explored hybrid designs. Crossformer[23] proposed a multi-scale attention mechanism to capture dependencies across different temporal resolutions. Other models, such as TFTformer[5], focus on fusing various types of features (static, known-future, and observed) within the Transformer architecture to improve interpretability and accuracy. More recently, this hybrid paradigm has evolved to explicitly integrate convolutional modules for enhanced local feature modeling. For instance, Sun et al.[24] proposed a framework combining VMD decomposition, Informer, and a CNN-LSTM module to capture localized peak-valley patterns in non-stationary load data. Similarly, Similarly, Li et al.[25] introduced a hybrid model merging Temporal Convolutional Networks (TCN) with channel-enhanced attention mechanisms to improve short-term load prediction accuracy. Tang et al.[26] introduced an image-based forecasting approach that synergizes Transformers for global pattern extraction with 2D-CNNs for local feature capture. Furthermore, addressing the critical need for model transparency, Xu and Chen[27] developed InterFormer, a probabilistic Transformer architecture incorporating a local variable selection network to enhance interpretability in residential net load forecasting. These models highlight a growing consensus on the necessity of specialized architectures that can handle diverse data characteristics simultaneously. Most recently, Hong et al.[28] introduced Patchformer, which integrates a patch embedding mechanism to segment multivariate time series into sub-series patches, effectively capturing both local and global semantic dependencies for long-term multi-energy forecasting. These models highlight a trend towards creating specialized architectures that can handle diverse data characteristics simultaneously.
- *Graph-Based Models for Load Forecasting:* An alternative line of research utilizes graph-based methods to explicitly model the structural and spatial relationships between different time series, such as neighboring households or interconnected grid components. For example, the EnGAT-BiLSTM framework[29] constructs a dynamic knowledge graph from load data and employs graph attention mechanisms to weigh the influence of neighboring nodes in its forecast. While powerful for capturing inter-series dependencies, the performance of these models is heavily reliant on the quality of the predefined graph structure. They can be sensitive to noise and face challenges in adapting to dynamically changing network topologies, although recent advancements like the spatiotemporal attention mechanisms proposed by Lv et al.[30] have attempted to mitigate these issues by dynamically weighing neighbor influence.

Research gap and motivation: Our review of the literature indicates that while significant progress has been made, a key challenge remains: developing a single, robust model that can jointly and effectively model both long-term, periodic patterns and short-term, aperiodic fluctuations without relying on a predefined spatial structure. Our proposed CLM-Former is designed to address this specific gap. It retains the powerful frequency-domain autocorrelation from Autoformer[16] to capture seasonality while introducing a dedicated CNN-LSTM subnetwork (CLM-subNet) to explicitly model the local, volatile patterns that other approaches often miss.

Problem definition

Long-term sequence time-series forecasting (LSTF) aims to predict multiple future time steps from historical observations. It follows the input-l-predict-k paradigm: given a look-back window of length l, forecast the next k steps. The model inputs are represented as $X^t = \left\{x_1^t, \ldots, x_l^t \middle| \, x_i^t \in \mathbb{R}^{d_x}\right\}$, and the outputs are represented as $\{y_1^t, \ldots, y_k^t\}$. Consequently, the aim is to find $f(.)$ such that it generates the desired output for the next k time steps (k being the prediction length) based on the inputs from the previous l time steps (l being the sequence length), as expressed in Eq. 1.

$$f\left(x_1^t, \ldots, x_l^t\right) = \{y_1^t, \ldots, y_k^t\} \tag{1}$$

A common approach to analyzing time series data involves decomposing it into its fundamental components: seasonal, trend, cyclical, and residual/random components. This decomposition aims to isolate and understand the underlying patterns and trends within the time series.

- *Seasonal Component:* The seasonal component represents the cyclical variations in the time series that repeat over a fixed period, such as monthly, quarterly, or yearly patterns. It captures the recurring fluctuations due to seasonal factors.
- *Trend Component:* The trend component represents the long-term direction or underlying growth of the time series. It captures the overall trend of the data over time, indicating whether it is increasing, decreasing, or remaining relatively stable.
- *Cyclical Component:* The cyclical component encompasses the broader, nonseasonal fluctuations in the time series that may not have a fixed periodicity. It captures the irregular patterns and trends that are not attributed to seasonality or the overall trend.
- *Residual/Random Component:* The residual component represents the random or unpredictable fluctuations in the time series, often referred to as "noise." It captures the deviations from the overall trend and seasonal patterns.

Figure 2 illustrates the decomposition of a time series into its constituent components. The original time series (top panel) is decomposed into cyclical, seasonal, trend, and residual/random components (panels below). Each component reveals a distinct pattern within the overall time series.

The trend represents the gradual shift in the values of a time series, causing the pattern to increase or decrease over time. Time series that do not exhibit a trend are considered stationary, while those with a trend are considered nonstationary. Trends are present in most real-world data, such as customer electricity consumption, making these data nonstationary. Since forecasting the duration of cycles is challenging, the cyclical effect is combined with the trend, forming the cyclical-trend component. This component is deterministic and represents the long-term, nonstationary fluctuations in the time series. Additionally, seasonality refers to repeating patterns in short, consecutive periods that maintain a stable magnitude and direction over time. This characteristic is also deterministic and represents the periodic variations in the time series. The random or residual component is the unexplainable part of the time series, which is stochastic and consists of the remaining series after removing the seasonal and trend components.

Two main models are used to combine these four components to extract the underlying signal. The choice of model depends on the characteristics of the time series:

- *Additive Model:* The additive model combines the four components by addition.
- *Multiplicative Model:* The multiplicative model combines the components via multiplication.

Each specific signal favors one of these models. Based on the characteristics of our electricity consumption data, we utilize the additive model.

Architectural overview

Encoder-decoder architecture has emerged as a cornerstone of deep learning models for sequence processing tasks. These architectures comprise an encoder module tasked with extracting a comprehensive representation from the input sequence, followed by a decoder module that utilizes this representation to generate the desired output sequence. The introduction of the transformer architecture[12], which incorporates multihead attention mechanisms into the encoder-decoder framework, has revolutionized the capabilities of these networks, propelling them to the forefront of various sequence processing applications.

This paper introduces CLM-Former, a novel deep learning framework for customer load forecasting. The proposed architecture builds upon the vanilla transformer model[12] with stacked autoencoders (SAEs)[31] and replaces the multihead attention mechanism with frequency-domain autocorrelation, inspired by Autoformer[16]. In our method, each encoder processes customer consumption sequences, extracts temporal–spatial dependencies, and sends the refined representations to the decoder for future consumption prediction. The consumption load of length $L\_seq = I$ is directly fed into the encoder, but for the inputs of the decoders, the load is divided into two parts—seasonal and cyclical—and then fed into the network. Additionally, decoders utilize the seasonal pattern generated by the encoders to predict the output. The overall architecture of the CLM-Former is illustrated in Fig. 3. Subsequently, we elaborate on each component of the architecture.

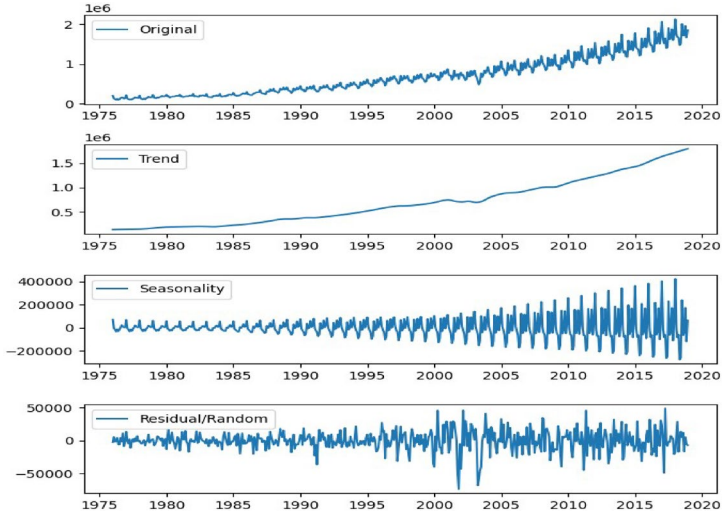


Fig. 2. Time series data decomposition into primary components. Using an additive decomposition model, the original time series (top panel) is separated into three distinct parts to isolate underlying patterns: (1) Trend Component: Illustrates the long-term progression and growth of electricity consumption over the years. (2) Seasonality Component: Captures the recurring, periodic fluctuations (e.g., annual cycles) with consistent frequency. (3) Residual (Random) Component: Represents the irregular, stochastic noise remaining after removing the trend and seasonal effects. This decomposition allows the model to process global trends and local variations independently.

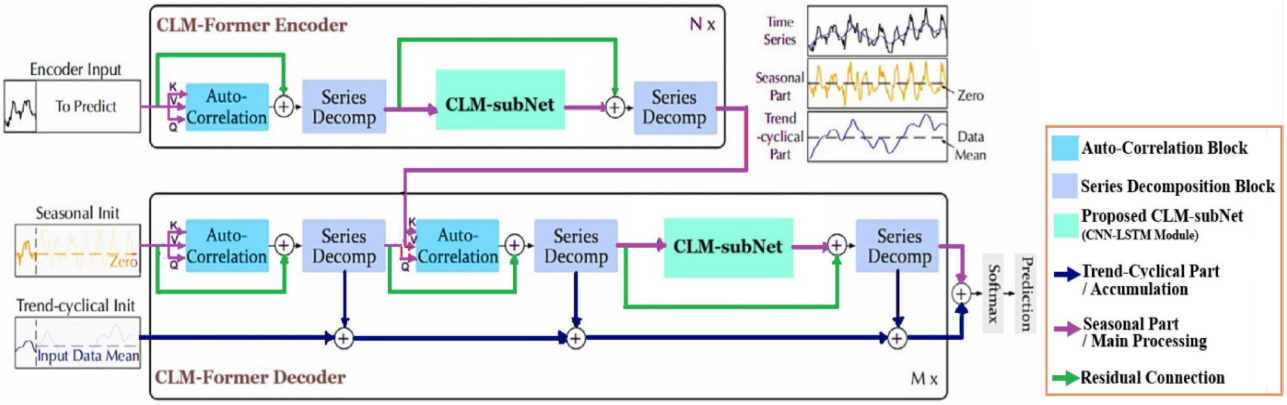


Fig. 3. Overall architecture of CLM-Former. The model follows an Encoder-Decoder structure. (Top) Encoder: Eliminates the long-term trend via Series Decomposition blocks and focuses on modeling the seasonal patterns using Auto-Correlation and the proposed CLM-subNet. (Bottom) Decoder: Progressively refines the trend and seasonal components. The CLM-subNet (highlighted in green) replaces the standard feed-forward network to capture localized, high-frequency fluctuations within the seasonal component. Data Flow: The arrows indicate the forward propagation, where the Decomposition blocks separate the series into Trend (accumulated towards the output) and Seasonal (processed by CLM-subNet) parts.

Why CLM-subNet is critical for improving load forecasting?

Accurate residential load forecasting necessitates capturing both overarching global trends and intricate local temporal dynamics within time series data. This section elucidates why the CLM-subNet is an indispensable component in achieving this comprehensive predictive capability.

Residential load forecasting models must simultaneously capture long-term dependencies, seasonal patterns, and rapid, short-term fluctuations due to the inherently complex nature of time series data in this domain. While Autoformer, relying on frequency-domain autocorrelation, has shown commendable performance in identifying long-term periodic patterns, its use of simple feed-forward layers for modeling seasonal components introduces limitations in detecting local correlations and abrupt changes.

To overcome these limitations, the proposed model incorporates a combination of CNN and LSTM within the CLM-subNet architecture. The CNN layers excel at extracting localized features, enabling the model to identify short-term variations caused by instantaneous user behaviors and environmental conditions with high precision. Meanwhile, LSTM layers, with their memory gating mechanism, are particularly adept at preserving long-term dependencies and modeling the recurring patterns characteristic of seasonal data.

This powerful combination allows the CLM-Former model to effectively address both local and long-term dynamics. By integrating the localized features extracted by CNN with the temporal dependencies captured by LSTM, the model delivers optimized performance for forecasting residential electricity consumption time series. This hybrid approach not only resolves the shortcomings of existing models in handling rapid fluctuations but also significantly enhances prediction accuracy across diverse scenarios.

In essence, the CLM-subNet provides a unique and essential synergy for accurate residential electricity consumption forecasting, a domain inherently prone to significant variations. It empowers the model to overcome Autoformer's limitations in detecting sudden changes and localized patterns, all while retaining Autoformer's excellent capabilities in modeling periodic and long-term trends. Integrating CNN and LSTM improves both local and long-range feature extraction, leading to more robust and accurate multi-horizon predictions.

Computational complexity of CLM-former

While Autoformer, as the foundational architecture of CLM-Former, benefits from a computational complexity of $O(LlogL)$ due to its autocorrelation mechanism, the integration of the CLM-subNet introduces additional computational operations. The CLM-subNet, composed of 1D Convolutional and LSTM layers, primarily contributes to a linear complexity with respect to the sequence length L (i.e., $O(L)$ or $O(L \times D^2)$ depending on architectural details).

Although these additional layers enhance the model's ability to capture short-term fluctuations and local dependencies, the dominant computational complexity of the overall CLM-Former architecture is largely retained from the Autoformer's $O(LlogL)$ self-attention replacement. However, it is important to acknowledge that the constant factor associated with this complexity increases due to the added operations within CLM-subNet.

Our experiments show that combining autocorrelation with the CLM-subNet yields significant accuracy gains with minimal overhead, confirming the effectiveness of our approach for multi-horizon load forecasting. Future work could explore further optimizations to reduce this constant factor, while maintaining the model's superior predictive performance.

CLM-former encoder

The CLM-Former Encoder comprises a multilayer structure consisting of N encoder layers. Each encoder layer in CLM-Former consists of the following main blocks:

The CLM-Former Encoder comprises a multilayer structure consisting of N encoder layers. Each encoder layer follows the core structure adopted from Autoformer, applying the Auto-Correlation mechanism followed by series decomposition blocks and our CLM-subNet for seasonal refinement. This specific order—applying Auto-Correlation before decomposition within each layer—is central to the progressive decomposition philosophy. It allows the model to first capture period-based dependencies from the combined signal, enabling a more informed and refined separation of trend and seasonality at each stage. The main blocks consist of:

- *Autocorrelation Block:* Computes the frequency-domain autocorrelation of the input sequence to capture long-range dependencies.
- Decomposition process: Decomposes the *output* of the autocorrelation (plus residual) into seasonal and cyclical trend components.
- CLM-subNet: Applied to the extracted seasonal component to capture local, hierarchical features, replacing the standard Feed-Forward network.

Before feeding the input into the encoder, an embedding operation can be applied to the raw load data. For input data of length $I \times D$, three different embeddings are performed. These embeddings include:

- *Token embedding:* Convert each token in the input sequence into a vector representation using a token embedding matrix.
- *Positional embedding:* Positional information is added to each token vector, allowing the model to learn the relative positions of tokens within the sequence.
- *Temporal embedding***:** Embeds the time variables into vectors using a temporal embedding matrix, enabling the model to capture temporal patterns in the data.

The sum of the outputs from these three embeddings, as shown in Eq. 2, is considered the encoder input, and $X_{en}^{0} \in R^{I*D}$ represents the historical load data., respectively, with x_{inp} being the initial input. The encoder input is defined as $X_{en}^{l} = Encoder(X_{en}^{L-1})$, where $l \in \{1,2,\ldots,N\}$ denotes the $l-th$ encoder layer.

$$X_{en}^{0} = token\_emb(x_{inp}) + positional\_emb(x_{inp}) + temporal\_emb(x_{inp}) \tag{2}$$

X_{en}^{0} is fed into the autocorrelation module to identify and expand the information dependencies in the frequency domain. This module decomposes the information into seasonal and trend components.

The CLM-subNet network receives the seasonal component and extracts local and deep features from the decomposed part. The combined result with the cyclical-trend part is then fed back into the decomposition subblock. Given that the encoder focuses on representing the seasonal component, the seasonal representations based on the encoder inputs are sent to the decoder for further use.

In summary, the encoder(.) is formulated as shown in Eqs. 3 and 4:

$$S_{en}^{l,1}, \_ = Decompose(AutoCorrelation\left(X_{en}^{l-1}\right) + X_{en}^{l-1}) \tag{3}$$

$$S_{en}^{l,2}, \_ = Decompose(CLM - Net\left(S_{en}^{l-1}\right) + S_{en}^{l-1}) \tag{4}$$

where $l \in \{1,2,\ldots,N\}$ is the encoder number, $S_{en}^{l,i}, i \in \{1,2\}$ is the block number, and $S_{en}^{l,1}$ is the seasonal value obtained from the input sequence, with $S_{en}^{l,2}$ being the final output of the encoder. The detailed operations of the autocorrelation subblock, which replaces the multihead attention mechanism with a frequency-based approach, and the CLM-subNet subblock are discussed in Sects. 4.4 and 4.5.

The structure of the stack of decoders consists of M layers, with each layer comprising three blocks. The first and last blocks are similar to those in the encoder, while the middle block includes a cross-autocorrelation attention subblock and a decomposition block. The cross-autocorrelation attention block aims to capture long-range dependencies between the decoder's output and the encoder's seasonal representation. It receives the seasonal output generated by the encoder as K and V and the output of the encoder block as Q. It performs its operations, which will be explained further, and its output is then passed to the decomposition subblock for decomposition.

Unlike the encoder, which directly receives the time series as input, the input to the decoder involves decomposing the input series $X_{de}^{0} \in R^{\left(\frac{I}{2}+O\right)*D}$ into two parts: seasonal $S_{de}^{0} \in R^{\left(\frac{I}{2}+O\right)*D}$ and trend $T_{de}^{0} \in R^{\left(\frac{I}{2}+O\right)*D}$. The relationship is defined as $S_{de}^{l}, T_{de}^{l} = Decoder(S_{de}^{l-1}, T_{de}^{l-1})$, where $l \in \{1,2,\ldots,M\}$ denotes the layer number in the stack of decoders, specifying the $l-th$ decoder. Equations 5–9 illustrate the operations of the decoder.

$$S_{de}^{l,1}, T_{de}^{l,1} = Decompose(AutoCorrelation\left(X_{de}^{l-1}\right) + S_{de}^{l-1}) \tag{5}$$

$$S_{de}^{l,2}, T_{de}^{l,2} = Decompose(AutoCorr\_Attention\left(S_{de}^{l,1}, S_{en}^{N}\right) + S_{de}^{l,1}) \tag{6}$$

$$S_{de}^{l,3}, T_{de}^{l,3} = Decompose(CLM - Net\left(S_{de}^{l,2}\right) + S_{de}^{l,2}) \tag{7}$$

$$T_{de}^{l} = T_{de}^{l-1} + w_{l,1} * T_{de}^{l,1} + w_{l,2} * T_{de}^{l,2} + w_{l,3} * T_{de}^{l,3} \tag{8}$$

$$X_{pred} = w_s * S_{de}^{M} + T_{de}^{M} \tag{9}$$

where $w_{l,i} i \in \{1,2,3\}$ represents the projector for the cyclical-trend component of the $i-th$ block T_{de}^{l}. The final prediction, X_{pred}, is obtained where w_s is a projection of component S_{de}^{M} to the target dimensions.

Decomposition method

In time series analysis, understanding the patterns exhibited by data values over time is crucial for accurate forecasting. The assumption is that future data patterns will align with historical patterns. Decomposition methods are a standard approach for processing complex time series data to extract predictable components[32]. In the CLM-Former model, decomposition is not only applied to the input of the decoder stack but also incorporated as submodules within both the encoder and decoder architectures. Two common approaches to time series decomposition are additive and multiplicative methods. In this paper, we focus on the additive decomposition method. We decompose the data into cyclical-trend and seasonal components, which represent the long-term progress and seasonality of the time series, respectively. Considering a length L for the input series $X \in R^{L*d}$, the function $Decompose(x)$ for creating X_t, X_s is calculated as shown in Eqs. 10 and 11.

$$X_t = AvgPooling(padding\,(X)) \tag{10}$$

$$X_s = X - X_t \tag{11}$$

The components $X_t \in R^{L*d}$ and $X_s \in R^{L*d}$ represent the trend and seasonal elements, respectively. In Eq. 10, $AvgPooling(.)$ is used for the moving average with a padding operator to maintain the sequence length.

Assume that the encoder input $X_{enc} \in R^{I*d}$ is a sequence from the past I time steps. The decoder input is obtained by decomposing $X_s\prime \in R^{(\frac{I}{2}+o)*d}$, as $X_s\prime, X_t\prime = Decompose(X_{enc\,\frac{I}{2}:I})$. Given $X_{enc} \in R^{\frac{I}{2}*D}$, the remaining sequence length for X_{dec_s} is filled with $X_0 \in R^{o*d}$, which is equivalent to a zero vector, and for $X_{dec\_t} \in R^{O*d}$, it is filled with the average of the data (Eqs. 12–14). This operation is similar to applying an infinite placeholder in the base transformer structure for unobserved sequences.

$$X_s\prime, X_t\prime = Decompose(X_{en\,\frac{I}{2}:I}) \tag{12}$$

$$X_{de\_se} = concat(X_{ds}\prime, X_0) \tag{13}$$

$$X_{de\_tr} = concat(X\prime_{dt}, X_{mean}) \tag{14}$$

Autocorrelation-based attention mechanism

In the CLM-Former model, an autocorrelation-based attention mechanism is employed to replace the self-attention mechanism in the vanilla transformer architecture. This approach leverages the concept of autocorrelation to capture temporal dependencies within the input sequence. In the following section, we will first define the concepts of single attention, attention head, and multi-head attention, and then explain the computation of autocorrelation.

Single attention

Single attention, introduced in the paper[12] under the name "Scaled Dot-Product Attention," serves as a valuable Technique in time series forecasting, enabling the model to focus on the most pertinent segments of the data during the prediction process. This technique involves computing a weight matrix that assesses the significance of each segment within the input data. The computed weight matrix subsequently dictates the influence of each input segment on the final prediction.

The calculation of single attention commences with the extraction of three vectors from the input: key (k), query (Q), and value (v). The attention score is then computed using Eq. 15:

$$attention\,(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{15}$$

where d_k represents the dimension of the key vector. This equation captures how attention is directed towards different data segments based on the similarity between the query and key vectors, and subsequently weights the value vectors proportionally to their respective importance.

Attention head

Each attention head independently processes a set of query, key, and value vectors, generating an attention matrix. Each head can to distinct and diverse aspects of the data, enabling the model to delve into the intricate details of the input.

Multi-head attention

To provide a broader perspective on the data and allow parallel attention to different aspects within the data, the multi-head attention technique is introduced as an extension of single attention[12]. In this approach, several attention mechanisms are executed in parallel, and their outputs are then combined. This method enables the model to focus on multiple diverse aspects of the data simultaneously.

In the computation process of multi-head attention, the inputs are first divided into several parts, with each part being processed by an attention head. For each head, three vectors—Query (Q), Key (K), and Value (V)—are computed, and the single attention calculation is performed separately for each head. The outputs of all heads are then concatenated to form a final matrix. The following Eqs. 16 and 17 demonstrates the calculation of multi-head attention:

$$MultiHead\left(Q,K,V\right)=Concat\left(head_1,\ldots,head_h\right)*\mathcal{W}_{output} \tag{16}$$

where:

$$head_i=Attention\left(Qw_i^Q,Kw_i^K,Vw_i^V\right) \tag{17}$$

Here, w_i^Q, w_i^K, and w_i^V are the weight matrices for the i-th head, and $\mathcal{W}_{output}$ is the final weight matrix for combining the outputs of all heads.

Autocorrelation

Based on the theory of stochastic processes[33,34], the similarity between the discrete-time processes X_t and X_{t-T} with lag τ can be obtained using the autocorrelation function $R_{XX}=\lim_{l\to}\frac{1}{L}\sum^{L}X_tX_{t-\tau}$, where R_{XX} is calculated using the fast Fourier transform (FFT). To replace the self-attention method in the vanilla transformer with autocorrelation, we first compute the top k lags between Q and K, where $k=\lfloor log\left(L\right)\rfloor$. Then, the obtained $R_{Q,K}\left(\tau\right)$ values are normalized using the softmax function. Finally, the similarity between them is calculated using the autocorrelation function $AutoCorrelation(Q,K,V)$, as formulated in Eqs. 18–20:

$$\tau_1,\ldots,\tau_k=argTopK\left(\mathcal{R}_{\mathcal{Q},\mathcal{K}}\left(\tau\right)\right) \tag{18}$$

$$\widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}\left(\tau_1\right),\ldots,\widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}\left(\tau_k\right)=softmax\left(\mathcal{R}_{\mathcal{Q},\mathcal{K}}\left(\tau_1\right),\ldots,\mathcal{R}_{\mathcal{Q},\mathcal{K}}\left(\tau_{\mathrm{k}}\right)\right) \tag{19}$$

$$AutoCorrelation\left(Q,K,V\right)=\sum_{i=1}^{k}Roll\left(\mathcal{V},\tau_i\right)\widehat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}}\left(\tau_i\right) \tag{20}$$

The function $Roll\left(\mathcal{V},\tau_i\right)$ represents the sequential selection of sequences of length τ_i from V.

The multihead attention version of the above equation is represented as Eqs. 21 and 22:

$$MultiHead\left(Q,K,V\right)=\mathcal{W}_{output}*Concat\left(head_1,\ldots,head_h\right) \tag{21}$$

where:

$$head_i=Auto-Correlation\left(Q_i,K_i,V_i\right) \tag{22}$$

where h is the number of heads, d_{model} is the dimension of the hidden states, and the query, key, and value for head i are represented as $\mathcal{Q}_i,\mathcal{K}_i,\mathcal{V}_i\in\mathbb{R}^{L\times\frac{d_{model}}{h}}$, where $i\in\{1,\ldots,h\}$. It was proven in[16] that the time complexity of autocorrelation is $O(l\log l)$.

In conclusion, the autocorrelation mechanism in Autoformer primarily functions as a global pattern extractor, capturing long-range periodic dependencies and revealing the dominant temporal rhythms within the entire sequence. By correlating the series with its time-shifted versions, it builds a coherent frequency-domain representation that highlights the major cyclical structures.

Importantly, this global modeling also produces an intermediate representation—particularly for the seasonal component—that encodes both the stable periodicities and the residual fine-scale variations that remain after decomposition. In this sense, the autocorrelation block not only summarizes the overarching temporal dynamics but also organizes the data in a way that exposes subtle, localized fluctuations suitable for further analysis.

This property provides a natural bridge to the subsequent CLM-subNet, which builds upon the globally informed seasonal representation to perform targeted, local spatio-temporal feature extraction. Hence, the autocorrelation mechanism effectively prepares the signal, enabling the CLM-subNet to refine and model the high-frequency, aperiodic behaviors that are beyond the scope of purely frequency-based modeling.

CLM-subNet

To enhance the network's ability to extract both local and global features from time series data, we introduce the CLM-subNet subnetwork, which incorporates convolutional layers followed by an LSTM layer, into the encoder-decoder stack of CLM-Former. This subnetwork aims to improve the network's performance in extracting salient features and enables it to handle time horizon predictions of varying lengths. The CLM-subNet subnetwork consists of convolutional layers, LSTM, and an MLP. It offers several advantages:

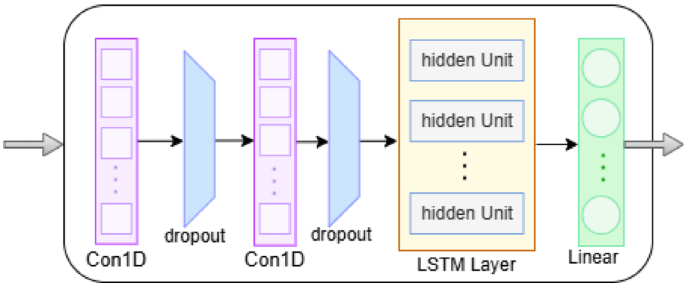


Fig. 4. The internal architecture of the proposed CLM-subNet. This module is designed to extract hierarchical spatio-temporal features from the seasonal component. It consists of: (1) 1D-CNN Layers: Two stacked 1D-Convolutional layers with dropout for extracting local high-frequency patterns and reducing noise. (2) LSTM Layer: Captures sequential dependencies within the extracted features. (3) Linear Layer: Projects the refined representation back to the model's dimension $\mathrm{d}_{\mathrm{model}}$.

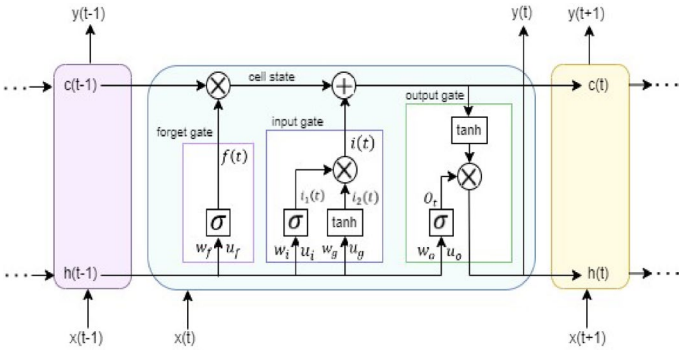


Fig. 5. Internal structure of the Long Short-Term Memory (LSTM) unit within the CLM-subNet. The diagram illustrates the gating mechanisms (forget, input, and output gates) that regulate information flow. This recurrent structure enables the CLM-subNet to effectively model temporal dependencies and sequential patterns within the extracted seasonal features.

Enhanced Feature Extraction: The combination of convolutional and LSTM layers enables the network to extract both local and global features from time series data, providing a more comprehensive representation of the underlying patterns and trends.

- *Time horizon flexibility:* The CLM-subNet subnetwork empowers the network to handle time horizon predictions of varying lengths, ranging from very short-term (VSTLF) to medium-short-term (MSTF) forecasts. This versatility makes the network applicable to a wider range of demand forecasting scenarios.
- *Scalability to multiple customers:* The CLM-subNet subnetwork can be effectively utilized for forecasting the load of multiple customers simultaneously. This capability is particularly valuable in real-world applications where demand patterns across a large customer base need to be considered.

The CLM-subNet subnetwork is applied to seasonal data in both the encoder and decoder stacks. The decomposed output of the previous subnetwork, with dimension d_{model}**,** is first fed into CLM-subNet. Within CLM-subNet, a one-dimensional convolutional layer first transforms the seasonal input from d_{model} to an intermediate dimension. Then, the second-dimensional convolutional layer extracts features from the data and converts them back to d_{model}. Dropout layers are applied to both convolutional layers to prevent overfitting. The output of the convolutional layers is then passed to the LSTM layers. The structure of CLM-subNet is illustrated in Fig. 4.

Figure 5 illustrates the internal structure of an LSTM cell. This structure is specifically designed to capture temporal relationships between past information and current conditions.

In the current work, a stacked LSTM architecture is integrated into the CLM-subNet for both encoder and decoder. Specifically, the encoder's CLM-subNet utilizes two LSTM layers, while the decoder's CLM-subNet employs one LSTM layer, with each LSTM layer comprising 64 hidden units. For each component in the input sequence, the components of each layer include the input gate i_t, forget gate f_t, cell state c_t, and output gate O_t as Eqs. 23 and 29:

$$i_1(t) = \sigma(x(t)\,u_i + h(t-1)\,w_i) \tag{23}$$

$$i_2(t) = \tanh(x(t)\,u_g + h(t-1)\,w_g) \tag{24}$$

$$i(t) = i_1(t) * i_2(t) \tag{25}$$

$$f(t) = \sigma(x(t)\,u_f + h(t-1)\,w_f) \tag{26}$$

$$c(t) = \sigma(f(t) * c(t-1) + i(t)) \tag{27}$$

$$o(t) = \sigma(x(t)\,u_o + h(t-1)\,w_o) \tag{28}$$

$$h(t) = \tanh(c_t) * o(t) \tag{29}$$

Here, δ represents the sigmoid activation function, and $tanh$ is the hyperbolic tanh activation function. $x(t)$ is the input unit at time t, and u and w are the weight matrices.

Algorithmic distinction from autoformer

While CLM-Former is built upon the robust Autoformer framework, it introduces a significant algorithmic modification to the internal processing of each transformer block. In the original Autoformer, the decomposed seasonal component is processed through a standard Feed-Forward Network (FFN), which applies transformations point-wise and cannot inherently model relationships across adjacent time steps.

In contrast, CLM-Former replaces this point-wise FFN with our specialized CLM-subNet. This modification fundamentally alters how the seasonal component is represented by introducing a hierarchical, sequence-aware processing path:

- First, the 1D-CNN layers act as local feature detectors, extracting salient high-frequency patterns from the seasonal data.
- Next, the LSTM layers model the temporal dependencies among this sequence of extracted features.

This architectural refinement transforms the processing from a series of independent operations into a dynamic, spatio-temporal analysis. As a result, CLM-Former preserves the core strengths of Autoformer (series decomposition and autocorrelation) while fundamentally enhancing its capability to model the localized, aperiodic dynamics critical for accurate load forecasting. This targeted enhancement leads to a more powerful and empirically superior model, as demonstrated in our results.

Results and discussion

In this section, we present the results of the proposed CLM-Former model for simultaneous load forecasting across 321 residential households. The dataset utilized in this study is the publicly available *Electricity* dataset, which contains 26,305 records of hourly electricity consumption data spanning three years, from July 1, 2016, at 2:00 AM to July 2, 2019, at 1:00 AM.

A key aspect of this methodology is the use of a global, multivariate household load forecasting. Instead of training 321 separate, individual models (one for each household), we train a single, unified CLM-Former model that learns concurrently from the data of all 321 households. This approach treats the electricity consumption of all households as a single, high-dimensional (321-variable) multivariate time series. This allows the model to capture complex shared seasonal patterns and inter-household dependencies, leading to a more robust and generalizable forecasting solution. Therefore, this single model simultaneously forecasts the future electricity consumption for all 321 households across the specified horizons (96, 192, and 336 future time steps).

To assess the performance of the CLM-Former, the model's accuracy is benchmarked against several state-of-the-art models. For this purpose, we selected six models from the transformer family and six deep learning-based models, comparing their forecasting accuracy with that of the CLM-Former. These benchmark methods include Autoformer, Crossformer, Informer, Reformer, LogTrans, Transformer, TiDe, SCINet, LSTNet, TCN, LSTM and CNN-LSTM.

In the following sections, we will discuss the evaluation metrics employed in this study and provide a detailed analysis of the experimental results obtained.

Data preparation

To ensure methodological consistency, reproducibility, and a fair comparison across all experiments, a standardized data preparation pipeline was applied *uniformly across all datasets*—our primary Electricity benchmark as well as the Weather, ETT and Traffic datasets used for generalizability testing. The following steps summarize the complete data preparation pipeline:

- **Data splitting:** Each dataset was divided chronologically into three subsets: 70% for training, 10% for validation, and 20% for testing. This temporal split preserves the natural sequential order of time series data and prevents any potential leakage of future information into the training process.
- **Handling of missing data:** We first performed an integrity check on all datasets. We confirmed that these standard benchmarks were complete and contained no missing values. Consequently, no data imputation or interpolation steps were necessary.
- **Normalization:** All input features were standardized using Z-score normalization, where each data point x was transformed as $x' = (x - \mu)/\sigma$. To avoid any risk of data leakage, the mean (μ) and standard deviation (σ) were computed exclusively from the training set of each dataset, and the same statistics were subsequently

used to normalize the corresponding validation and test sets. This approach ensures consistent scaling across all phases of model evaluation while maintaining data integrity.

- **Seasonal decomposition:** It is important to note that seasonal decomposition is not a static pre-processing step in our pipeline. Instead, it operates as an integrated, learnable module within the CLM-Former architecture, dynamically decomposing the input sequences into *seasonal* and *cyclical–trend* components during training and inference. The implementation details of this mechanism are provided in Section Decomposition method.

Experimental environment and metrics

The CLM-Former model was implemented using Python 3.11.7, Torch 2.2.2 + cu121, NumPy 1.26.4, and Pandas 2.1.4. The experiments were conducted on a system equipped with a 13th Gen Intel Core i5-13600KF CPU, an NVIDIA GeForce RTX 4080 GPU with 16 GB of GPU memory, and 64 GB of system memory. This setup ensured seamless execution across all benchmark models.

To evaluate all the models on the abovementioned datasets, commonly used statistical metrics, including the mean squared error (MSE) and mean absolute error (MAE), were utilized. These metrics were deliberately chosen to ensure direct methodological comparability with the foundational baselines (e.g., Autoformer), to align with our MSE loss function, and for their numerical stability, as relative metrics like MAPE are unreliable when true data values approach zero. These evaluations were conducted using the MSE and MAE, as defined in Eqs. 30 and 31, respectively:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y}_i)^2 \tag{30}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \widehat{y}_i| \tag{31}$$

where N is the number of samples, y_i is the $i-th$ actual sample, and $\widehat{y}_i$ is the $i-th$ predicted sample.

To ensure the proposed CLM-Former model operates at its optimal performance, a rigorous hyperparameter optimization (HPO)[35] process was conducted. Given the complexity and potential impact of hyperparameters on model efficacy, we employed the random search method, which has been shown to be effective in exploring a vast parameter space more efficiently than grid search for deep learning models[36].

The objective metric for this search was the Mean Squared Error (MSE) on the validation set. We did not employ k-fold cross-validation, as this method is methodologically inappropriate for time-series data due to its violation of temporal dependencies. Instead, we used the standard fixed chronological training and validation split. We performed 250 random trials for each prediction horizon to thoroughly explore the defined search space. The stability of this search was ensured by this large number of trials and the consistent application of early stopping with a patience of 3 epochs, which prevents overfitting and ensures the selected parameters are robust and generalizable.

The comprehensive range of values considered for each hyperparameter, including sequence length, label length, batch size, dropout rates, and the number of layers for encoders, decoders, and LSTMs, is detailed in Appendix B, Table 5. Sample results from various hyperparameter configurations explored during this process are provided in Appendix B, Table 6, illustrating the impact of different settings on model performance.

Throughout the optimization, early stopping was consistently applied to prevent overfitting and ensure that the model captured the most generalizable patterns from the training data. The specific set of hyperparameters that yielded the lowest average MSE on the validation set across all tested prediction horizons was carefully selected as the optimal configuration for the final CLM-Former model used in our comparative analysis. These final optimal settings are precisely listed in Table 1.

Parameters	Value
Sequence length	128
Label length	48
Batch size	64
Dropout	0.05
Dropout for LSTM	0
Loss function	MSE
Encoder layers	4
Decoder layers	4
Num layer for LSTM in encoder	2
Num layer for LSTM in decoder	1
Activation	GeLU

Table 1. Parameter settings.

(a) Transformer family comparison														
Models	**CLM-Former**		**Autoformer (2022)**		**CrossFormer (2023)**		**Informer (2021)**		**Reformer (2020)**		**logTrans (2019)**		**Transformer (2017)**	
Metrics	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
96	**0.196**	**0.313**	0.201	0.317	0.219	0.314	0.274	0.368	0.312	0.402	0.258	0.357	0.260	0.358
192	**0.203**	**0.316**	0.222	0.334	0.231	0.322	0.296	0.386	0.348	0.433	0.266	0.368	0.266	0.367
336	**0.229**	0.341	0.231	0.338	0.246	**0.337**	0.300	0.394	0.350	0.433	0.280	0.380	0.280	0.375
AVG	**0.209**	**0.323**	0.218	0.329	0.232	0.324	0.290	0.382	0.336	0.422	0.268	0.368	0.268	0.366
Avg improvement %			4.13	1.82	9.91	0.31	27.93	15.45	37.80	23.46	22.10	12.23	22.02	11.75
(b) Deep learning family comparison														
Models	**CLM-Former**		**TiDE (2023)**		**SCINet (2022)**		**LSTNet (2018)**		**TCN (2018)**		**LSTM**		**CNN-LSTM**	
Metrics	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
96	**0.196**	**0.313**	0.237	0.329	0.247	0.345	0.680	0.645	0.985	0.813	0.377	0.438	0.353	0.412
192	**0.203**	**0.316**	0.236	0.330	0.257	0.355	0.725	0.676	0.996	0.821	0.443	0.471	0.409	0.441
336	**0.229**	**0.341**	0.249	0.344	0.269	0.369	0.828	0.727	1.000	0.824	0.438	0.472	0.412	0.446
AVG	**0.209**	**0.323**	0.240	0.334	0.257	0.356	0.744	0.682	0.993	0.819	0.419	0.460	0.391	0.433
Avg improvement %			12.92	3.29	18.68	9.27	71.91	52.64	78.95	60.56	50.11	29.78	46.5	25.4

Table 2. Accuracy evaluation of models based on the MAE and MSE metrics. Bold values indicate the best performance.

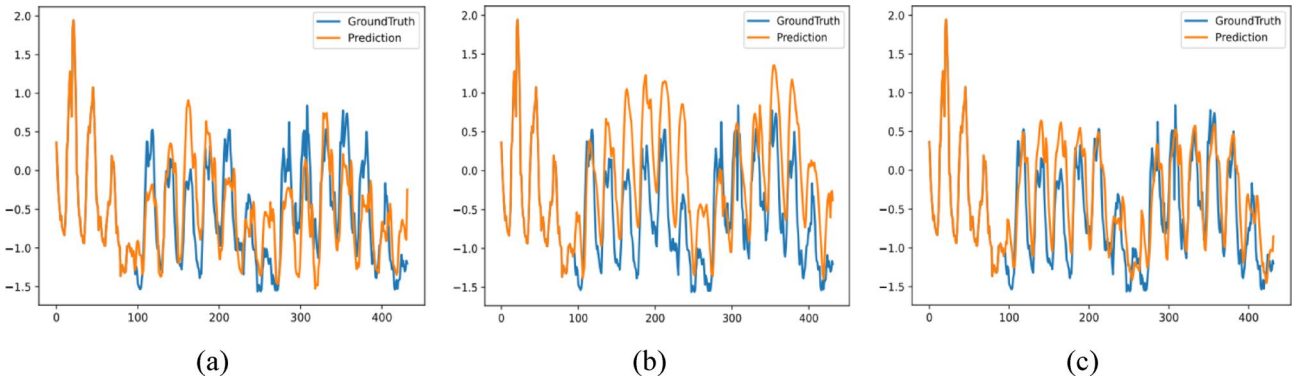


Fig. 6. Qualitative comparison of forecasting results for the 336-step horizon. The plots illustrate the Ground Truth (blue lines) versus the Predictions (orange lines) generated by: (**a**) The proposed CLM-Former, (**b**) Informer, and (**c**) Transformer. Observation: As highlighted, CLM-Former demonstrates a superior alignment with the ground truth, particularly in capturing rapid fluctuations and peak load values, visually corroborating the quantitative performance gains reported in Table 2.

Experimental results

The performance results of the proposed CLM-Former model compared to state-of-the-art methods are presented in Table 2. Table 2a provides a comparison between CLM-Former and several models in the Transformer family (Autoformer, CrossFormer, Informer, Reformer, logTrans, and the base Transformer) for time series forecasting. Table 2b highlights the performance of CLM-Former against deep learning models, including TiDE, SCINet, LSTNet, TCN, and CNN-LSTM. The comparison is conducted for three prediction intervals (96, 192, and 336), with Mean Squared Error (MSE) and Mean Absolute Error (MAE) used as evaluation metrics. Additionally, the average improvement percentage of the CLM-Former model relative to the other models under investigation has been calculated using Eq. 32.

$$imp = \left(\frac{V_{baseline} - V_{CLM-Former}}{V_{baseline}}\right) \times 100 \tag{32}$$

To evaluate the predictive performance of CLM-Former, we benchmarked it against both Transformer-based and conventional deep learning models over multiple forecasting horizons (96, 192, and 336 steps). Performance metrics included Mean Squared Error (MSE) and Mean Absolute Error (MAE).

To ensure that the reported improvements are not due to randomness or initialization noise, we performed a statistical significance analysis based on five independent runs using different random seeds. For each forecasting horizon, both CLM-Former and Autoformer were trained with identical seeds to enable paired comparison.

We report the mean and standard deviation (mean ± std) of the MAE over the 5 runs. In addition, we conducted two complementary paired statistical tests: (1) a paired t-test and (2) a Wilcoxon signed-rank test. Across all forecasting horizons (96, 192, and 336), both tests yielded p-values below 0.05, confirming that CLM-Former significantly outperforms Autoformer. (Detailed statistical test results, including t-statistics and exact p-values, are provided in Appendix C, Table 7).

To complement the quantitative results presented in Table 2, Fig. 6 provides a visual comparison of the forecasting capabilities of CLM-Former against key baselines (Informer and Transformer) for the 336-step prediction horizon. This visualization showcases the model's ability to capture the complex dynamics inherent in the residential load data.

As illustrated, while all models attempt to follow the general pattern, CLM-Former demonstrates a closer alignment with the ground truth, particularly in capturing the peaks and troughs of electricity consumption. This qualitative evidence supports the superior quantitative performance reported earlier, highlighting CLM-Former's enhanced effectiveness in modeling both long-term trends and short-term fluctuations. (Further visualizations comparing the models across different horizons are available in Appendix D).

Comparison with transformer-based models

As detailed in Table 2a, CLM-Former demonstrates consistent superiority over state-of-the-art Transformer variants, achieving the lowest average MSE (0.209) and MAE (0.323) across all prediction horizons. Compared to the canonical Vanilla Transformer, our model reduces the average MSE by approximately 22.0%, highlighting the fundamental effectiveness of the decomposition-based architecture. More notably, when benchmarked against Autoformer—the strongest Transformer-based baseline and the direct foundation of our work—CLM-Former achieves a statistically significant improvement, reducing the average MSE by 4.13% and MAE by 1.82%. This performance gain is particularly pronounced in longer horizons (e.g., a 5.15% reduction in MAE at 336 steps), confirming that the integration of the CLM-subNet successfully captures the rapidly changing, localized patterns that the standard frequency-domain attention of Autoformer often misses. Furthermore, CLM-Former

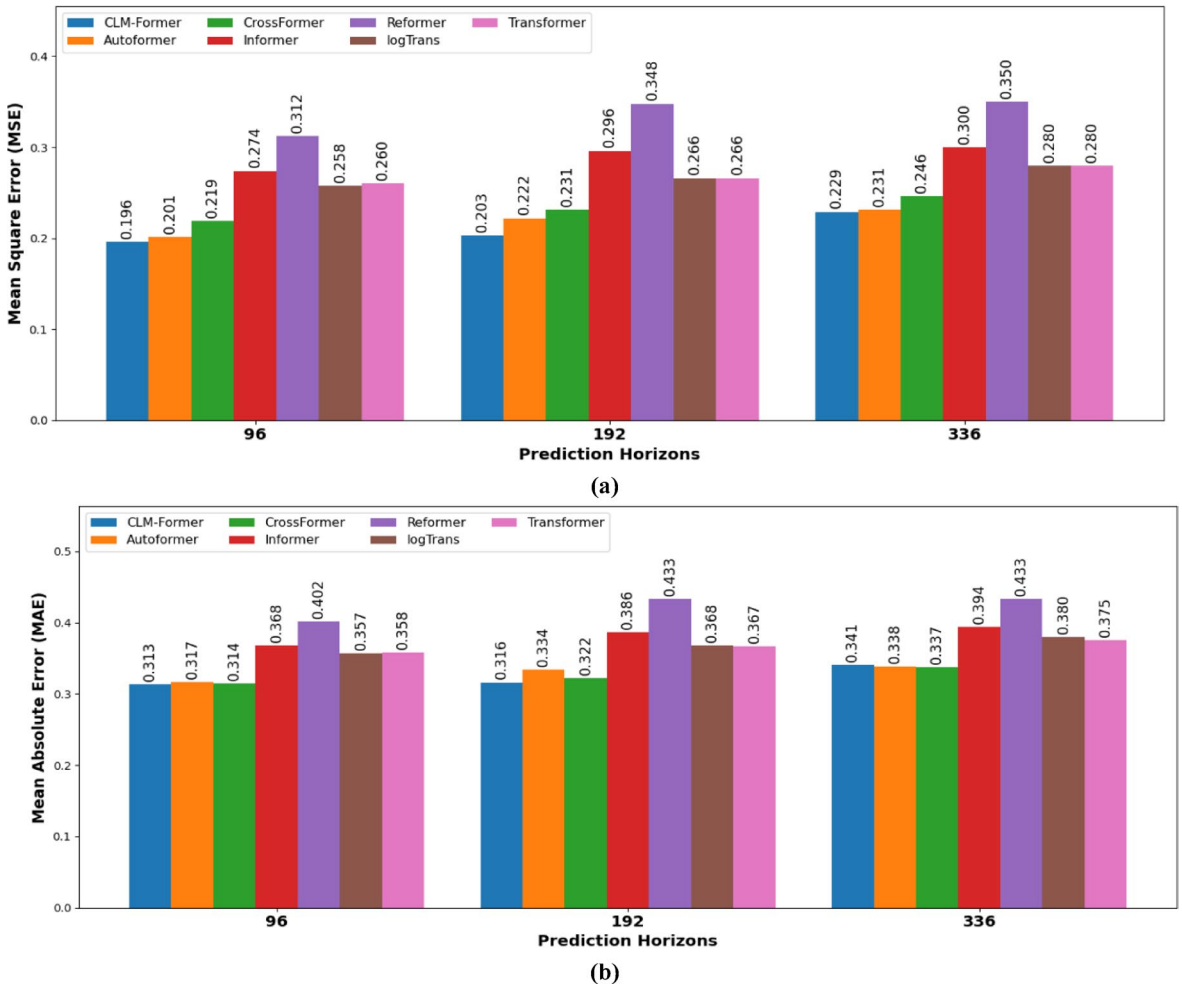


Fig. 7. Performance Comparison of CLM-Former against Transformer-Based Models**.** The figure shows a comparison of **(a)** Mean Square Error (MSE) and **(b)** Mean Absolute Error (MAE) for CLM-Former and other Transformer-based models across the 96, 192, and 336-step prediction horizons.

outperforms recent competitive models such as Crossformer (MSE 0.232) and Informer (MSE 0.290), proving its robustness in modeling complex temporal dependencies.

The superior performance of CLM-Former is visually confirmed in Fig. 7. The plots illustrate the performance gap across all forecasting horizons, showing that our model consistently achieves the lowest Mean Square Error (a) and Mean Absolute Error (b). This highlights its enhanced capacity for modeling both local and long-term dependencies.

Comparison with deep learning models

Table 2b presents the performance of CLM-Former against conventional deep learning architectures, including TiDE, SCINet, LSTNet, TCN, LSTM, and CNN-LSTM. A critical finding in this analysis is the substantial margin by which CLM-Former outperforms traditional hybrid models. Even after rigorously optimizing the CNN-LSTM baseline (yielding an improved MSE of 0.391), CLM-Former still achieves a remarkable 46.5% reduction in average MSE. This drastic difference underscores the structural advantage of integrating convolutional and recurrent layers within a decomposition-based Transformer block, as opposed to simply stacking them sequentially. Additionally, our model surpasses recent MLP-based architectures such as TiDE (achieving a 12.92% average improvement in MSE) and SCINet (18.68% improvement). While traditional models like LSTNet and TCN often struggle with stability in long-term forecasting, CLM-Former maintains consistent accuracy, validating its ability to hierarchically process both short-term local features and long-term global trends effectively.

Figure 8 complements the tabular results by presenting the comparative error metrics against the deep learning baselines. The visualizations clearly demonstrate the consistent outperformance of CLM-Former, which maintains the lowest prediction errors for both MSE (a) and MAE (b) across all forecasting horizons.

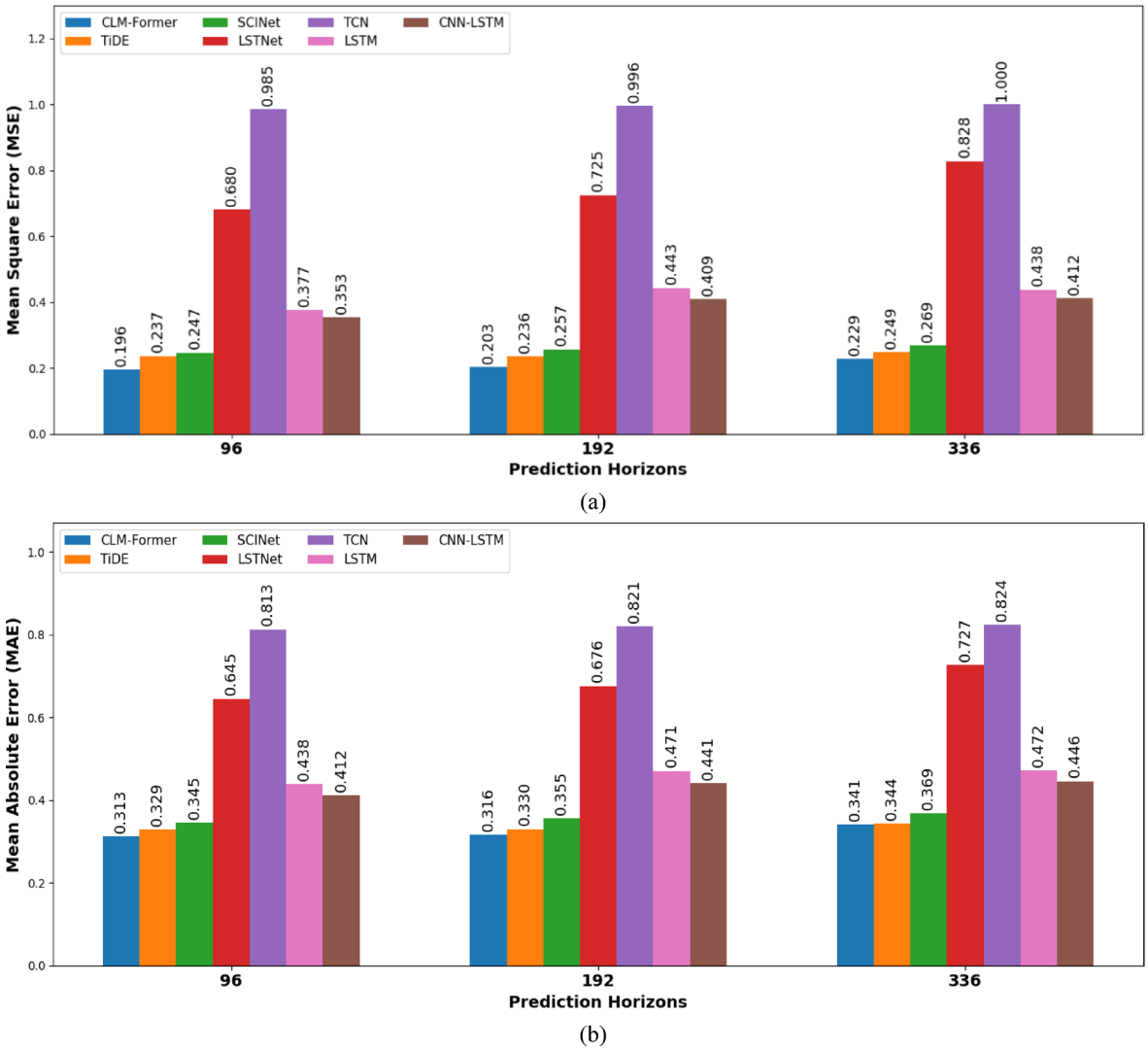


Fig. 8. Performance Comparison of CLM-Former against Deep Learning Models. The figure shows a comparison of (**a**) Mean Square Error (MSE) and (**b**) Mean Absolute Error (MAE) for CLM-Former and other deep learning baselines across the 96, 192, and 336-step prediction horizons.

Generalizability analysis on additional domains

To verify that the performance improvements of CLM-Former are not limited to residential load forecasting and can generalize to other domains, we extended our experimental evaluation to three additional widely-used benchmarks: Traffic, ETTm2 (Electricity Transformer Temperature), and Weather.

Evaluation Strategy: Given that CLM-Former is a direct architectural enhancement of Autoformer our primary objective in this analysis was to isolate and validate the specific contribution of this enhancement across diverse data distributions. Therefore, we focused the comparative analysis on CLM-Former versus the Official Autoformer baseline across three prediction horizons (96, 192, 336).

Discussion of Results: The results summarized in Table 3 demonstrate the robust generalizability of the proposed architecture:

ETTm2 (Energy Domain): CLM-Former achieves remarkable improvements in the energy domain. For instance, at the 96-step horizon, it reduces MSE by approximately 45% (0.1847 vs. 0.3354) and MAE by 22.5%, confirming that the CLM-subNet is highly effective at capturing thermal inertia and load dynamics.
Traffic (Spatial–Temporal Domain): The model consistently outperforms the baseline in MSE across all horizons (e.g., 4.3% reduction at horizon 96), indicating its ability to handle the complex, non-stationary patterns of traffic flow.
Weather (Meteorological Domain): While the baseline performs slightly better at the shortest horizon (96), CLM-Former demonstrates superior long-term forecasting capability. At the 336-step horizon, it achieves a substantial 15.5% reduction in MSE (0.3787 vs. 0.4482), suggesting that the hierarchical features extracted by CLM-subNet become increasingly valuable for modeling complex weather dependencies over longer periods.

In conclusion, CLM-Former consistently enhances the performance of the Autoformer backbone across diverse domains, validating that the proposed architectural modification provides generalized benefits beyond the primary electricity dataset.

Ablation study and computational complexity analysis

To provide a comprehensive evaluation of the CLM-Former architecture, we conducted an integrated study that simultaneously isolates the contribution of individual components (Ablation Study) and assesses the practical trade-offs regarding model size and speed (Computational Complexity Analysis). This addresses the need for empirical validation of our efficiency claims compared to the baseline. We benchmarked the full CLM-Former against several architectural variants on the Electricity dataset across three horizons (96, 192, 336).

Experimental setup

The variants evaluated are:

1. **Vanilla autoformer:** Replaces the CLM-subNet with a standard point-wise Feed-Forward Network (FFN), serving as the theoretical baseline described in the original paper.
2. **Autoformer (Official/GitHub):** Corresponds to the official convolution-based implementation (denoted as **"w/o LSTM"**). This serves as our primary strong baseline.
3. **w/o CNN:** A variant removing the Convolutional layers from the CLM-subNet to test local feature extraction.
4. **Encoder-Only/Decoder-Only:** Partial integration strategies where CLM-subNet replaces the FFN only in the Encoder or Decoder.
5. **CLM-Former:** The full proposed architecture.

To ensure reproducibility and provide a standardized benchmark for computational efficiency, the ablation study and complexity analysis were conducted in a Google Colab environment utilizing an NVIDIA T4 GPU (16 GB GDDR6 VRAM, Turing architecture) with PyTorch 2.9.0. Table 4 summarizes the predictive performance alongside wall-clock computational metrics measured in this environment (batch size 32).

Discussion of results

Architectural contribution (ablation analysis)

The results confirm the necessity of the proposed hybrid design.

Model	Metrics	Traffic			ETTm2			Weather		
		96	192	336	96	192	336	96	192	336
Autoformer	MSE	0.6943	0.6490	0.6380	0.3354	0.2044	0.2513	**0.3004**	0.3699	0.4482
	MAE	0.4445	0.4003	0.3961	0.3900	0.3166	0.3551	**0.3737**	0.4259	0.4722
CLM-Former	MSE	**0.6642**	**0.6339**	**0.6343**	**0.1847**	**0.2014**	**0.2340**	0.3114	**0.3637**	**0.3787**
	MAE	**0.4220**	**0.3970**	**0.3970**	**0.3022**	**0.3152**	**0.3397**	0.3841	**0.4212**	**0.4066**

Table 3. Performance comparison on Traffic, ETTm2, and Weather datasets (MSE/MAE). Bold values indicate the best performance.

PLen	Model variant	MAE	MSE	Param (M)	Train time (s)	Inference time (s)	Peak GPU memory train (MB)	Peak GPU memory inference (MB)
96	Vanila Autoformer[1]	0.2416	0.3499	17.84	1807.47	33.51	4448.16	771.99
	w/o CNN (Enc/Dec)	*0.1991*	0.3151	17.19	2295.50	33.65	4524.82	852.94
	w/o LSTM (Enc/Dec)[2]	0.1994	*0.3140*	34.62	2133.81	39.68	6168.39	900.50
	Linear Enc/ Comp Dec	0.2034	0.3202	25.90	2226.49	38.58	5453.35	835.15
	Comp Enc/ Linear Dec	0.2080	0.3230	25.90	2440.61	35.98	5239.70	833.15
	CLM-Former	**0.1934**	**0.3104**	33.96	3126.82	39.26	6247.66	1042.58
192	Vanila Autoformer[1]	0.2345	0.3422	17.84	3196.31	50.15	6816.44	1131.48
	w/o CNN (Enc/Dec)	0.2306	0.3445	17.19	2285.17	50.61	6928.79	1126.50
	w/o LSTM (Enc/Dec)[2]	*0.2124*	*0.3248*	34.62	4100.67	55.13	8953.38	1258.37
	Linear Enc/ Comp Dec	0.2200	0.3344	25.90	5445.66	53.08	8264.00	1190.88
	Comp Enc/ Linear Dec	0.2252	0.3357	25.90	4205.00	52.41	7608.64	1191.11
	CLM-Former	**0.2092**	**0.3208**	33.96	4443.01	54.24	9060.80	1253.75
336	Vanila Autoformer[1]	0.2478	0.3530	17.84	6471.56	70.96	9932.10	1669.09
	w/o CNN (Enc/Dec)	0.2394	0.3516	17.19	3196.53	70.54	10,098.42	1665.68
	w/o LSTM (Enc/Dec)[2]	*0.2290*	*0.3417*	34.62	4871.64	80.73	12,727.99	1798.62
	Linear Enc/ Comp Dec	0.2300	0.3430	25.90	4728.91	79.77	12,116.58	1731.39
	Comp Enc/ Linear Dec	0.2485	0.3527	25.90	6437.61	72.98	10,725.38	1731.41
	CLM-Former	**0.2172**	**0.3302**	33.96	5579.44	81.05	12,510.26	1944.14

Table 4. Ablation study on the contribution of each component in CLM-subNet (Electricity dataset). Best results in bold, second best in italics. [1]Vanilla Autoformer: Strictly follows the original paper's illustrated feed-forward network (no Conv1D layers). [2]WO LSTM (Enc/Dec): Equivalent to the official Autoformer implementation with two Conv1D layers (current standard in literature).

- **Superiority of CLM-subNet:** Across all three forecasting horizons, CLM-Former consistently yields superior predictive accuracy compared to all other variants, including the strong "w/o LSTM" baseline (the official Autoformer implementation). Specifically, relative to the official Autoformer, CLM-Former reduces MAE from 0.1994 to 0.1934 (3.01% improvement) at the 96-step horizon, and from 0.2124 to 0.2092 (1.51% improvement) at the 192-step horizon. Notably, the model achieves its largest gain at the 336-step horizon, reducing MAE from 0.2290 to 0.2172, a significant 5.16% improvement. A similar trend is observed for MSE. These consistent gains demonstrate that the synergistic combination of CNN (for local pattern extraction) and LSTM (for temporal refinement) provides a significantly richer seasonal-path representation than the official Autoformer baseline alone.
- **Impact of individual components:** The ablation analysis explicitly highlights the critical contribution of each architectural element. Removing the CNN layers (w/o CNN) leads to a substantial deterioration in performance, with MAE increasing from 0.2092 to 0.2306 at the 192-step horizon—a relative error increase of 10.23%. This underscores the indispensable role of convolutions in capturing high-frequency local variations that simpler structures miss. Similarly, the exclusion of LSTM layers (w/o LSTM) consistently degrades prediction accuracy, particularly at longer horizons; for instance, at the 336-step horizon, MAE rises from 0.2172 to 0.2290 (5.43% increase), confirming that recurrent modeling is essential for preserving long-range temporal dependencies within the extracted features.
- **Placement strategy:** The ablation results strongly validate the full symmetrical integration of the CLM-subNet in both encoder and decoder blocks. While partial integration strategies offer improvements over the baseline, the full CLM-Former consistently outperforms the best partial variant (Decoder-Only integration) by approximately 4.9% to 5.6% in terms of MAE across medium and long horizons. This confirms that capturing localized features at both the encoding stage (for robust representation learning) and the decoding stage (for precise generative prediction) is essential for maximizing forecasting accuracy. To visually consolidate these findings across different forecasting scales, Fig. 9 illustrates the comparative accuracy (MSE and MAE) of the architectural variants for both the representative medium-term (192-step) and long-term (336-step) horizons.

Computational efficiency and trade-offs

Contrary to concerns that adding recurrent layers might drastically increase cost, the empirical data reveals a highly favorable profile:

- **Parameter efficiency via bottleneck:** Remarkably, CLM-Former (33.96M) requires fewer parameters than the official Autoformer baseline (34.62M), despite the added LSTM layer. This counter-intuitive result arises from the bottleneck structure introduced by the LSTM. With a hidden size of 64, the LSTM compresses features from $d_{model} = 512$ into a low-dimensional representation, and the subsequent Linear layer maps $64 \rightarrow 512$. This significantly reduces the number of parameters compared to the "w/o LSTM" architecture, where projections operate directly at the full d_{model}. Thus, CLM-subNet enhances feature extraction while simultaneously improving parameter efficiency through effective feature compression.

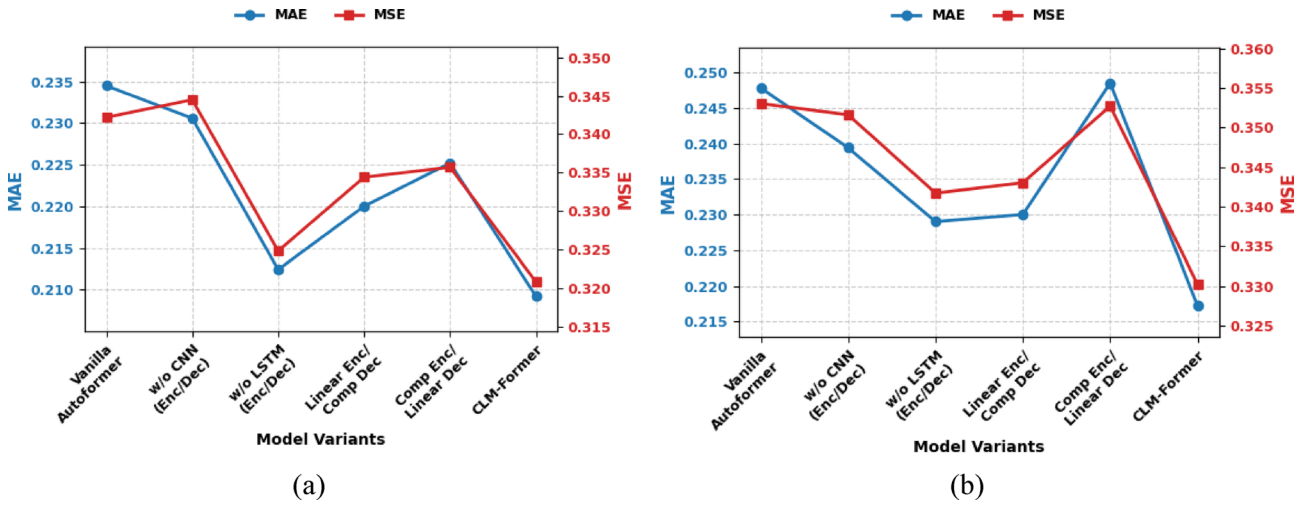


Fig. 9. Impact of architectural components on forecasting accuracy. The bar charts compare the MSE and MAE of different architectural variants across two key horizons: (**a**) Medium-term horizon (192 steps). (**b**) Long-term horizon (336 steps). Observation: In both scenarios, the full CLM-Former (blue bars) consistently achieves the lowest error rates, visually confirming that the synergistic contribution of the CLM-subNet components becomes increasingly vital for robust performance in longer forecasting tasks.

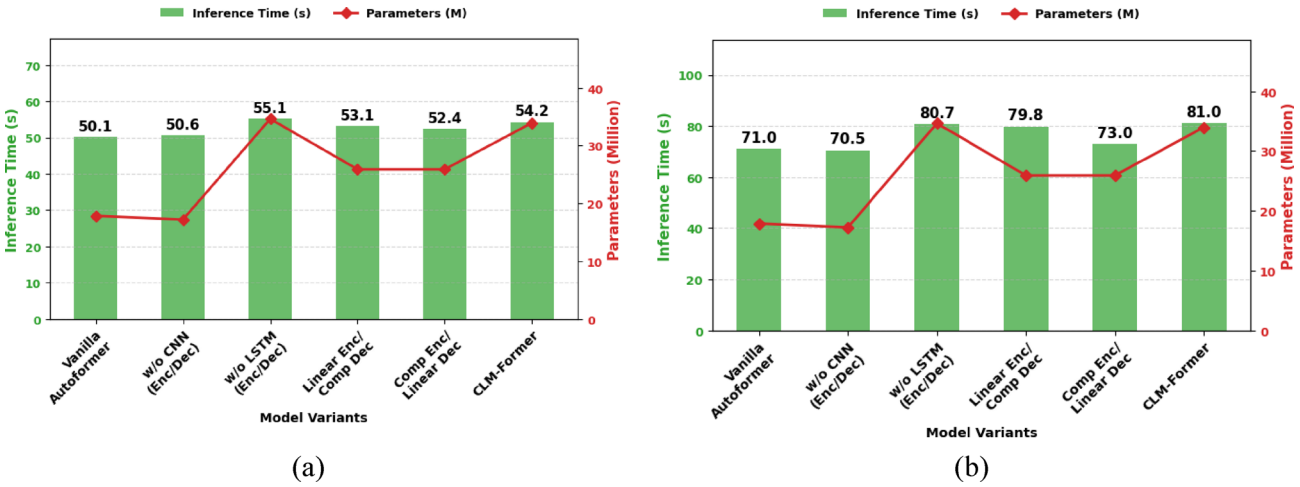


Fig. 10. Computational efficiency analysis. The charts contrast the Inference Time (bars, left axis) with the Parameter Count (line, right axis) for each model variant: (**a**) Medium-term horizon (192 steps). (**b**) Long-term horizon (336 steps). Observation: Notably, CLM-Former maintains an inference speed comparable to the baseline while offering a highly efficient parameter footprint (lower than the 'w/o LSTM' variant) across both horizons, validating its suitability for real-time deployment in diverse forecasting scenarios.

- **Inference speed (Deployment):** Importantly, despite incorporating recurrent operations, CLM-Former maintains inference efficiency comparable to—and in some cases better than—the official Autoformer baseline. At the 96-step and 192-step horizons, CLM-Former achieves slightly faster inference times (−1.06% and −1.61%, respectively), while at the longest 336-step horizon the overhead remains negligible (+0.40%). Averaged across all horizons, CLM-Former delivers a 0.78% reduction in inference time relative to the baseline. This confirms that the proposed architecture preserves the $O(LlogL)$ efficiency of Autoformer and introduces no meaningful latency penalty during deployment. Figure 10 further contextualizes this efficiency by plotting the inference time against the model parameter count for both the 192-step and 336-step horizons, highlighting the consistently favorable trade-off achieved by CLM-Former even as the prediction length extends.

- **Training dynamics:** Incorporating recurrent layers within the CLM-subNet naturally introduces additional computational operations, resulting in increased training times compared to the convolution-only "w/o LSTM" baseline (e.g., an increase of approximately 46.5% at the short 96-step horizon). However, this computational overhead varies but remains moderate as the forecasting task becomes more complex; the relative increase drops to + 8.3% at the 192-step horizon and stabilizes at + 14.5% for the extended 336-step horizon. This favorable trend is attributed to more efficient convergence: the superior feature extraction capabilities of the CLM-subNet enable the model to capture complex patterns more effectively, thereby satisfying the early stopping criterion in fewer epochs. Consequently, the moderate increase in offline training time is effectively counterbalanced by the significant gain in predictive accuracy (e.g., a 5.2% reduction in MAE at the 336-step horizon), representing a highly efficient trade-off for long-term forecasting.
- **Peak GPU memory footprint:** Regarding memory efficiency, the peak GPU memory usage during both training and inference shows only a marginal increase (averaging ~ 3.5%) compared to the baseline. This confirms that CLM-Former remains well within the memory limits of standard hardware accelerators (e.g., NVIDIA T4), ensuring that the enhanced model capacity does not impose prohibitive resource overhead.

This integrated analysis demonstrates that CLM-Former does not trade accuracy for efficiency. Instead, it achieves statistically significant accuracy gains with a negligible (or even favorable) impact on computational resources and deployment speed, offering a robust solution for practical forecasting tasks.

Scalability to longer horizons

Our experiments evaluated forecasting horizons up to 336 steps, consistent with prior benchmark settings. As expected, the results in Table 2 show that prediction error (MSE and MAE) increases as the horizon extends, reflecting the inherent uncertainty in long-range forecasting. While the rate of error growth slightly intensifies between the 192-step and 336-step horizons, the increase remains gradual. Critically, CLM-Former consistently maintains its performance advantage over the baselines across all tested horizons, demonstrating its relative robustness.

Beyond accuracy, the model's architectural efficiency further supports its scalability. As highlighted in the ablation study, the CLM-subNet utilizes a bottleneck design that reduces the total parameter count compared to the baseline and maintains stable inference times even for longer sequences. This suggests that CLM-Former is not only accurate but also computationally viable for extended horizons. Although explicit testing beyond 336 steps was outside the scope of this study, these characteristics imply that the model is inherently suited for longer patterns without risking sharp instability or prohibitive computational costs. Further investigation into extended horizons (e.g., 720 steps) is planned for future work.

Model interpretability and visualization

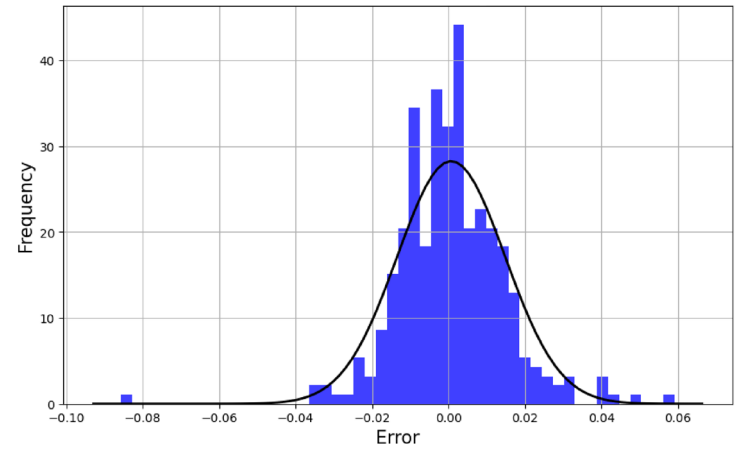
To validate the internal mechanism of CLM-Former and address the need for model interpretability, we conducted a visual analysis of the learned dependencies. Unlike standard Transformers that rely on point-wise attention, CLM-Former utilizes Auto-Correlation Maps to discover period-based dependencies. As detailed in Appendix E, these maps visualize how the model focuses on specific time lags corresponding to the inherent periodicity of the data (e.g., daily or weekly cycles). Furthermore, we generated Saliency Maps via gradient analysis to identify which historical time steps contribute most significantly to the prediction. These visualizations confirm that the model effectively attends to critical seasonal patterns and local variations rather than processing the input sequence uniformly.

Model stability and error analysis

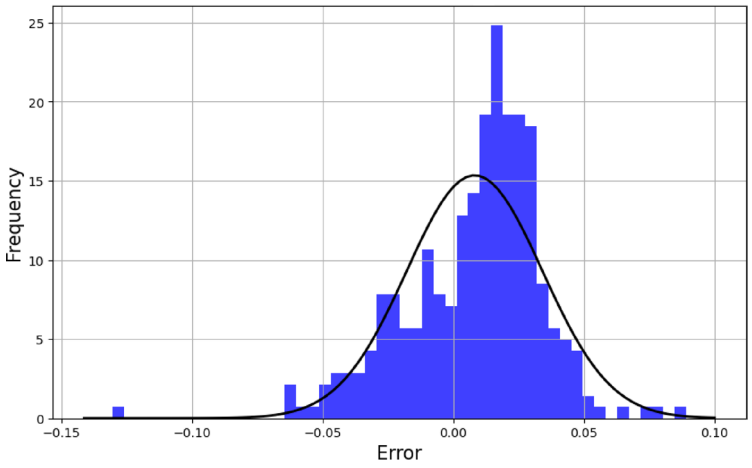
In addition to aggregate accuracy, forecasting reliability and model robustness were examined through error distribution analysis across all 321 residential households.

Figure 11 illustrates the histogram of prediction errors over the three horizons. Across all settings, CLM-Former maintains a symmetric and centered error distribution, with means consistently close to zero. This indicates an absence of systematic bias—i.e., no tendency toward consistent over- or under-prediction.

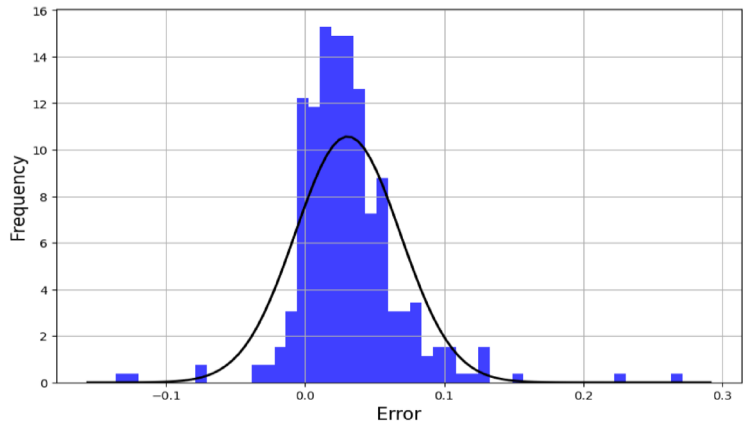
Moreover, the narrow spread and low standard deviation reflect strong stability, even under varying consumption patterns and extended horizons. Notably, the shape and concentration of the error distribution remain consistent as the prediction window increases, confirming the model's robust generalization capability and resilience to temporal variability. Overall, the error distribution analysis strongly supports the claim that CLM-Former provides reliable and robust forecasts across diverse conditions.



(a)



(b)



(c).

Fig. 11. Distribution of prediction errors for CLM-Former across all households. The histograms display the frequency of errors for prediction horizons of (**a**) 96 steps, (**b**) 192 steps, and (**c**) 336 steps. Visual Components: The blue bars represent the frequency of error values, and the black curve indicates the fitted normal distribution density. Observation: In all three horizons, the distributions are symmetric and centered around zero, indicating a near-zero systematic bias. The narrow spread (low standard deviation) confirms the model's stability and robustness against temporal variability, even as the forecasting horizon extends.

Limitations and future work

While CLM-Former demonstrates significant improvements, it is important to acknowledge certain architectural characteristics and potential limitations, which also suggest avenues for future research:

- ***Performance on non-periodic data*:** The autocorrelation mechanism strongly depends on periodicity detection. Although the CLM-subNet explicitly addresses local and aperiodic fluctuations, performance may still degrade on highly irregular series lacking clear seasonality. Further testing on such datasets is planned.
- ***Interpretability*:** The inherent series decomposition provides a degree of interpretability by separating trend and seasonality. However, fully dissecting the intricate interplay between the frequency-based Autocorrelation mechanism and the spatio-temporal features captured hierarchically by the CLM-subNet remains challenging. Developing more advanced interpretability techniques tailored to this hybrid architecture could yield valuable insights.

Addressing these aspects offers exciting opportunities for future research. Beyond exploring efficiency enhancements and advancing model interpretability, we plan to extend our benchmarking to include emerging architectural paradigms, such as Patching-based Transformers (e.g., PatchTST) and Linear-based models (e.g., DLinear), to further evaluate the comparative strengths of the CLM-Former architecture.

Conclusion

In this study, we proposed CLM-Former, a novel hybrid forecasting architecture designed to overcome a fundamental challenge in smart grid management: the simultaneous modeling of long-term seasonal trends and rapid, short-term fluctuations in electricity consumption. By synergizing a frequency-domain autocorrelation mechanism with a specialized CLM-subNet (combining convolutional and recurrent layers), the model effectively captures multi-scale temporal dependencies that existing architectures often miss.

Through comprehensive experiments on real-world residential load data, CLM-Former demonstrated consistent superiority over both state-of-the-art Transformer variants (including Autoformer, FEDformer, Informer, and Crossformer) and optimized deep learning baselines (such as TiDE and CNN-LSTM). Crucially, our empirical analysis confirms that this superior performance is achieved with high computational efficiency. Contrary to the trade-offs typically associated with hybrid models, CLM-Former maintains an inference speed virtually identical to the Autoformer baseline and requires fewer parameters due to its optimized bottleneck design. Furthermore, the model exhibits faster training convergence in complex long-horizon tasks, highlighting the effectiveness of the CLM-subNet in extracting robust features.

These results, validated by rigorous statistical significance tests ($p < 0.05$), confirm CLM-Former as a robust, accurate, and deployment-ready solution for demand response programs and distributed energy scheduling. Beyond its empirical strength, the architecture offers modularity and interpretability, bridging the gap between theoretical complexity and practical utility.

In future work, we plan to extend our benchmarking to include emerging paradigms such as Patching-based Transformers (e.g., PatchTST) and Linear-based models. Additionally, we aim to incorporate contextual and exogenous variables (e.g., weather data) to further refine prediction accuracy, and explore federated learning and model compression techniques to enhance scalability and privacy-preserving capabilities in real-time smart grid deployment.

Data availability

The dataset analyzed during the current study is the publicly available ElectricityLoadDiagrams20112014 dataset, which can be found in the UCI Machine Learning Repository: [https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014], (https:/archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014). The source code for the **CLM-Former** model developed in this study is publicly available on GitHub at [https://github.com/mozhgan-Rahmatinia/CLM_Former_to_enhancing_load_prediction], (https:/github.com/mozhgan-Rahmatinia/CLM_Former_to_enhancing_load_prediction) .

Appendix A: Comparative overview of attention strategies

This appendix provides a conceptual visualization of different attention and correlation mechanisms discussed in this paper. Each strategy offers a different trade-off between computational efficiency and the ability to capture temporal dependencies in time series data.

- *Full attention (Transformer):* As depicted in Fig. 12a, the standard self-attention mechanism computes pairwise interactions between all tokens across the sequence. While comprehensive in capturing all possible dependencies, this approach has a quadratic computational complexity ($O(L^2)$), making it prohibitive for long sequences.
- *Sparse attention (Informer):* To improve efficiency, sparse attention mechanisms limit computations to a subset of tokens. For example, the *ProbSparse* attention used in Informer (Fig. 12b) selects only the most significant queries, reducing complexity. However, this risks neglecting critical long-range dependencies that fall outside the sparse set, which can impact prediction accuracy.
- *LogSparse attention (LogTrans):* This strategy (Fig. 12c) offers a compromise by selectively attending to both nearby and distant tokens using a logarithmic pattern. It maintains essential long-term dependencies while significantly reducing complexity. Its performance, however, may diminish for shorter sequences where its fixed logarithmic structure might not be optimal.
- *Autocorrelation (Autoformer):* Unlike attention-based methods, the autocorrelation mechanism (Fig. 12d) operates in the frequency domain to identify periodic patterns. This approach efficiently captures long-term,

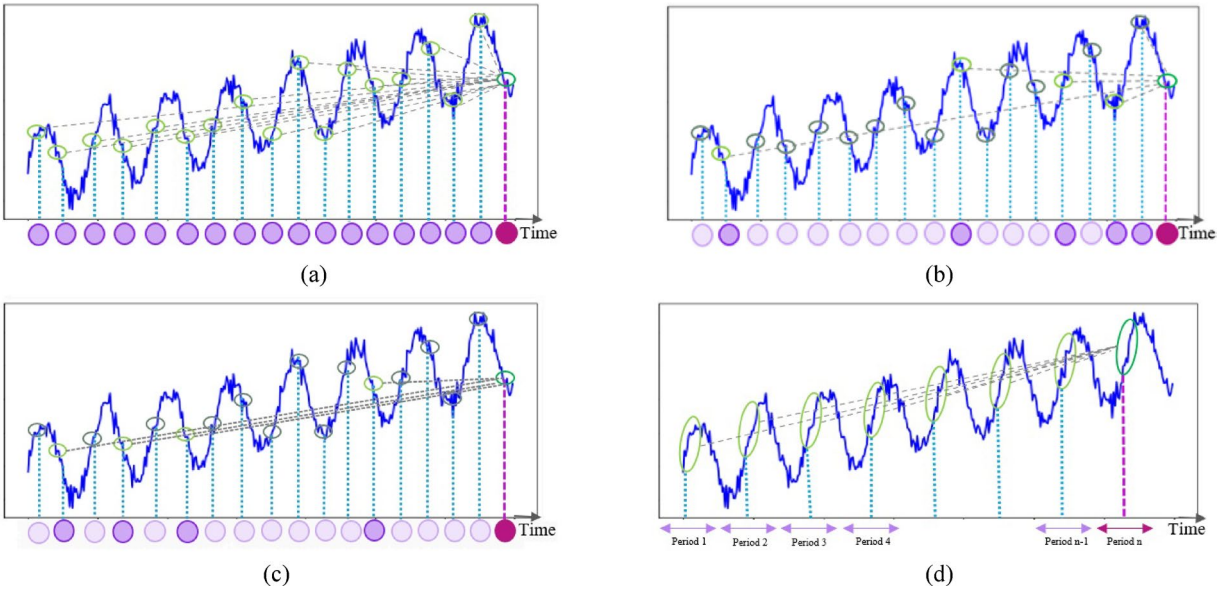


Fig. 12. A Visual Comparison of Attention and Correlation Mechanisms. The diagrams illustrate how different strategies model temporal dependencies. (**a**) Full attention connects every point to every other point. (**b**) LogSparse attention captures both local and distant (logarithmically spaced) connections. (**c**) Sparse attention focuses only on a few significant connections. (**d**) Autocorrelation identifies similarities based on periodic lags rather than direct point-to-point attention.

recurring dependencies and is highly suited for seasonal time series. Nevertheless, its reliance on periodicity makes it less effective at modeling rapid, short-term fluctuations or non-periodic variations, which are critical in domains like residential load forecasting.

Appendix B: Hyperparameter optimization details

Given the wide range of possible hyperparameter settings and their direct impact on the learning process and performance improvement, we utilize the random search method for hyperparameter optimization (HPO)[35]. Random search is a black-box optimization method for HPO that can yield better results than grid search[36]. To perform a random search of the model's hyperparameters, we have selected a range of values for the search, as detailed in the Table 5:

Sample results from the hyperparameter optimization trials are provided in Table 6, illustrating the impact of different settings on model performance.

Following numerous trials, the optimal hyperparameter values, as listed in Table 1, were obtained using the HPO method.

Parameters	Values
Sequence length	48, 96, 128
Label length	48, 96
Prediction length	96, 192, 336
Batch size	32, 64, 128
Dropout	0, 0.001, 0.005, 0.05, 0.5
Dropout for LSTM	0, 0.001, 0.005, 0.05. 0.5
Loss function	MSE, MAE
Encoder layers	1, 2, 4, 8
Decoder layers	1, 2, 4, 8
Num layer for LSTM in encoder	1, 2, 4
Num layer for LSTM in decoder	1, 2, 4
Activation	ReLU, GeLU

Table 5. Different hyper parameters.

Encoder layers	Decoder layers	Num layer for LSTM encoder	Num layer for LSTM decoder	Loss function	Dropout	Dropout for LSTM	Batch size	Activation	Sequence length	Label length	Pred length	MAE	MSE
2	1	2	1	MSE	0.5	0	128	GeLU	96	96	192	0.284	0.377
1	1	3	1	MSE	0.1	0	64	GeLU	96	48	96	0.197	0.31
4	2	2	1	MSE	0.05	0	64	ReLU	96	96	192	0.233	0.341
2	2	4	8	MAE	0.05	0.001	64	GeLU	96	96	96	0.236	0.339
4	2	2	1	MAE	0.05	0.01	32	ReLU	96	96	192	0.241	0.344
2	2	8	8	MAE	0.1	0.05	64	GeLU	128	96	192	0.229	0.337
4	2	4	1	MSE	0.001	0.5	64	GeLU	96	48	192	0.223	0.334
2	2	2	2	MSE	0.005	0.05	128	ReLU	128	48	336	0.281	0.374
2	1	2	4	MSE	0.01	0.01	128	GeLU	96	96	336	0.273	0.366
4	4	2	1	MAE	0.1	0.05	128	GeLU	128	48	96	0.201	0.317
1	2	4	8	MAE	0.05	0	64	GeLU	96	48	192	0.239	0.344
4	4	2	1	MSE	0	0.05	64	GeLU	128	48	96	0.196	0.313
4	4	2	1	MSE	0	0.05	64	GeLU	128	48	192	0.203	0.316
4	4	2	1	MSE	0	0.05	64	GeLU	128	48	336	0.229	0.341

Table 6. Sample hyperparameter combinations and their corresponding performance metrics during optimization.

Appendix C: Detailed statistical significance analysis

To rigorously validate the performance improvements of CLM-Former over the foundational Autoformer baseline, we conducted paired statistical significance tests. Both models were trained and evaluated over 5 independent runs using identical random seeds to ensure a fair, paired comparison.

Table 7 presents the detailed statistical metrics, including the Mean Absolute Error (MAE) (reported as Mean $\pm$ Standard Deviation), the T-statistic and *P*-values from the Paired *t*-test (both one-tailed and two-tailed), and the *P*-value from the Wilcoxon signed-rank test (one-tailed).

As shown in the Table 7, for all prediction horizons (96, 192, and 336 steps), the p-values for both parametric and non-parametric tests are consistently below the significance level of 0.05. This statistically confirms that the reduction in forecasting error achieved by CLM-Former is significant and not attributable to random initialization noise.

Model	MAE (mean ± std)	Horizon	T-Statistic *t*-test	*P*-value *t*-test (one tail)	*P*-value *t*-test (two tail)	*P*-value Wilcoxon	Statistically significant?
CLM-Former	0.19676 ± 0.00165	96	− 6.3547	0.001571	0.003142	0.031250	Yes ($p < 0.05$)
Autoformer	0.19920 ± 0.00132						
CLM-Former	0.21039 ± 0.00447	192	− 4.3530	0.006064	0.012128	0.031250	Yes ($p < 0.05$)
Autoformer	0.22225 ± 0.00783						
CLM-Former	0.22161 ± 0.00406	336	− 5.5361	0.002602	0.005204	0.031250	Yes ($p < 0.05$)
Autoformer	0.22642 ± 0.00495						

Table 7. Detailed results of paired statistical significance tests comparing CLM-Former and Autoformer (MAE metric) across 5 independent runs.

Appendix D: Supplementary visualization of forecasting results

This appendix provides supplementary visual evidence to complement the quantitative results presented in the main text. While a representative visualization comparing model predictions for the 336-step horizon has been included in the main Results section, this appendix offers further qualitative comparisons for additional forecasting horizons.

Figures 13 and 14 illustrate the normalized ground truth load curves versus the predictions generated by CLM-Former, Informer, and Transformer for the 720-step and 192-step horizons, respectively, for a sample customer. These plots offer a visual assessment of how effectively each model captures the complex, multi-scale dynamics inherent in residential electricity consumption data.

As visually confirmed across these different horizons (Figs. 13, 14, and the corresponding figure in the main text), CLM-Former consistently demonstrates a closer adherence to the actual load patterns compared to the baseline models. Specifically, CLM-Former appears more adept at capturing both the overall cyclical trends and the sharper, short-term fluctuations (peaks and troughs) in consumption.

This qualitative superiority aligns with our architectural design:

- The **autocorrelation mechanism** enables the model to effectively identify and leverage the underlying periodicities in the data.
- The **series decomposition**, integrated within the model, helps separate long-term trends from seasonal variations, allowing for more focused modeling.

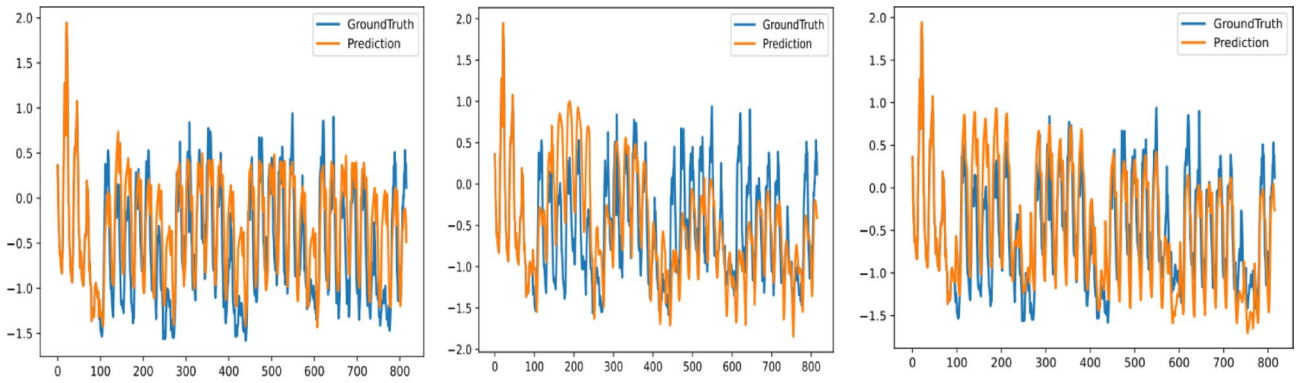


Fig. 13. Load Prediction by (**a**) CLM-Former, (**b**) Informer, and (**c**) Transformer for the Next 720 Time Intervals.

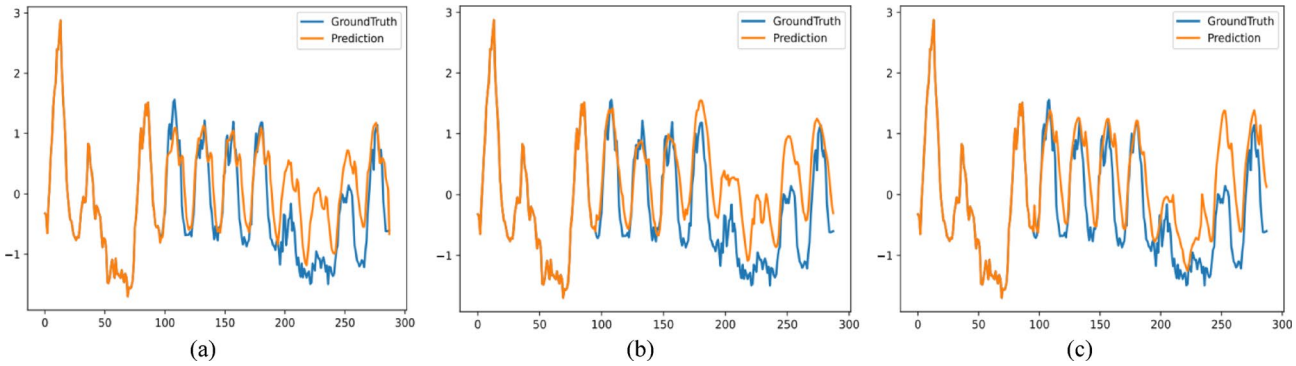


Fig. 14. Predictions Made by Three Models: (**a**) CLM-Former, (**b**) Informer, and (**c**) Transformer for the Next 192 Time Intervals.

- Crucially, the **CLM-subNet** provides the necessary capacity to capture the fine-grained, local dynamics within the seasonal component, which simpler feed-forward layers in standard Autoformer might miss.

Together, these visualizations provide compelling qualitative support for the quantitative findings, illustrating CLM-Former's enhanced capability in multi-horizon load forecasting.

Appendix E: Visualization of learned dependencies and feature importance

This appendix provides a visual inspection of how CLM-Former processes temporal data, offering insights into its "black-box" decision-making process and the model's focus on multi-scale temporal dependencies.

Auto-correlation maps

To investigate how the model identifies temporal dependencies, we visualize the learned Auto-Correlation maps from two distinct encoder layers. Figure 15 displays the attention patterns for (a) Layer 1 and (b) Layer 2. In these heatmaps, the x-axis represents the Time Lag (key positions, ranging from 0 to 60 steps), and the y-axis represents the Query Time Step(input positions). Brighter colors (yellow/green) indicate higher positive correlation scores, signifying stronger temporal dependencies, while darker regions (purple/blue) correspond to negative or weak correlations.

As observed in both Layer 1 and Layer 2, distinct vertical bands of high activation appear consistently at specific lags (e.g., prominent bands around lag indices 20–25 and 45–50). This vertical alignment indicates that regardless of the current query time step, the model consistently attends to the same historical intervals. This visually confirms that the network has successfully learned the dominant periods (periodicity) inherent

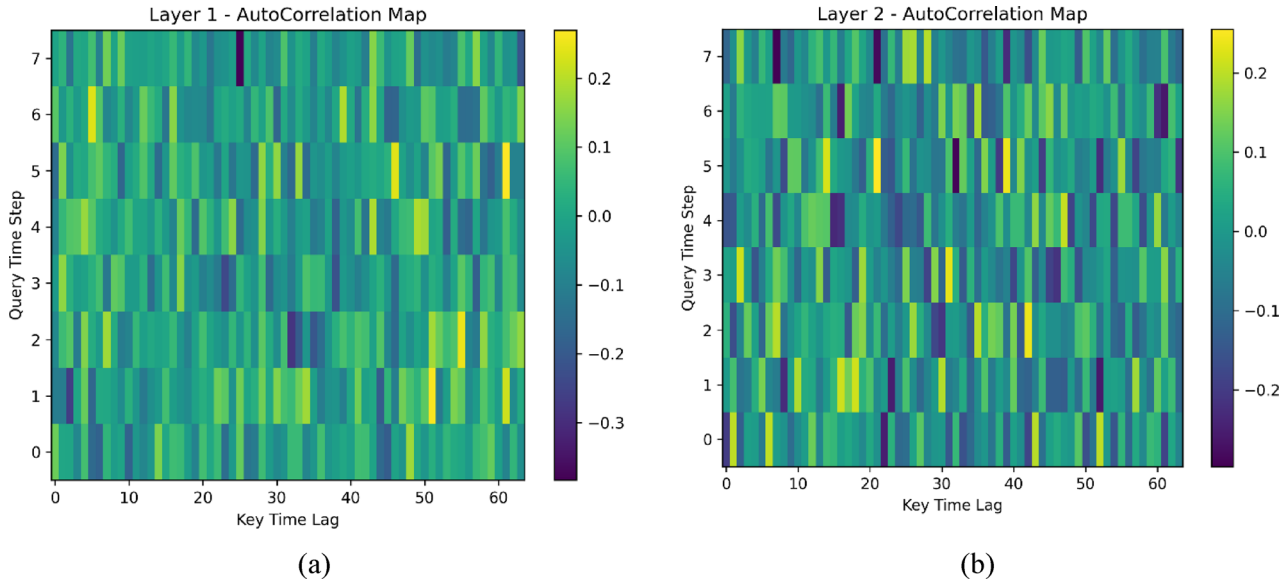


Fig. 15. Visualization of learned Auto-Correlation Maps from the Encoder. The heatmaps illustrate the correlation intensity between query time steps and key time lags for (**a**) Layer 1 and (**b**) Layer 2. The consistent vertical bright bands across different layers demonstrate that the model robustly identifies and focuses on specific periodic lags (e.g., daily or weekly cycles) to capture the underlying seasonality of the load data.

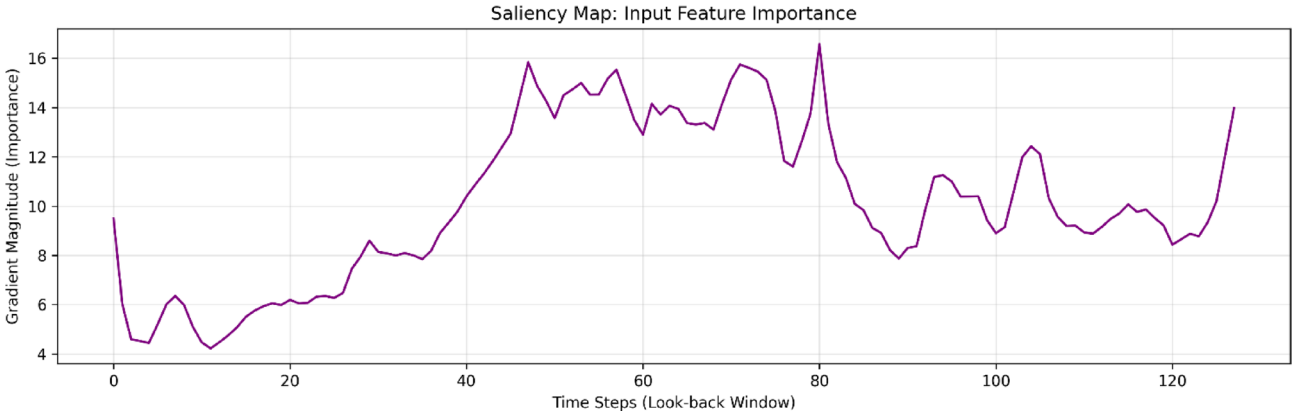


Fig. 16. Saliency Map showing input feature importance. The plot displays the gradient magnitude (y-axis) across the input look-back window (x-axis). The significant fluctuations and prominent peaks indicate that the model selectively attends to specific, high-impact historical time steps, effectively filtering noise and focusing on critical temporal dynamics for prediction.

in the electricity consumption data and utilizes these recurring patterns across multiple layers for forecasting. This behavior aligns perfectly with the theoretical design of the Autoformer architecture, which prioritizes period-based dependencies over point-wise attention.

To further interpret the model's decision-making process, we computed the Saliency Map using gradient back-propagation[37]. Figure 16 visualizes the absolute gradient magnitudes for each time step in the input look-back window with respect to the model's output. Higher magnitudes indicate greater sensitivity, meaning perturbations in those historical steps have a stronger influence on the forecast.

As illustrated in Fig. 16, the importance distribution is highly non-uniform, exhibiting distinct peaks (e.g., around time steps 45–55 and 75–80, reaching magnitudes up to 16) and valleys. This indicates that CLM-Former does not treat all historical data equally; instead, it selectively prioritizes high-impact segments—such as specific past events or seasonal phases—while suppressing less relevant noise. This selective mechanism, driven by the CLM-subNet's hierarchical processing (local CNN filtering and sequential LSTM weighting), enhances robustness to non-stationary patterns and directly contributes to the observed accuracy gains.

Received: 7 August 2025; Accepted: 31 December 2025
Published online: 28 January 2026

References

1. Habbak, H., Mahmoud, M., Metwally, K., Fouda, M. M. & Ibrahem, M. I. Load forecasting techniques and their applications in smart grids. *Energies* **16**, 1480 (2023).
2. Badr, M. M. et al. Detection of false-reading attacks in smart grid net-metering system. *IEEE Internet Things J.* **9**, 1386–1401 (2021).
3. Lago, J., De Ridder, F., Vrancx, P. & De Schutter, B. Forecasting day-ahead electricity prices in Europe: The importance of considering market integration. *Appl. Energy* **211**, 890–903 (2018).
4. Arif, A. et al. in Web, Artificial intelligence and network applications. in *Proceedings of the Workshops of the 34th International Conference on Advanced Information Networking and Applications (WAINA-2020)*, Springer. 471–483.
5. Ahmad, A., Xiao, X., Mo, H. & Dong, D. TFTformer: A novel transformer based model for short-term load forecasting. *Int. J. Electr. Power Energy Syst.* **166**, 110549 (2025).
6. Rahmatinia, S. M., & Seno, S. A. H. Enhancing NTMA with simultaneous multi-QoS parameterprediction using transformer-based deep learning. In *2024 11th International Symposium on Telecommunications (IST)* 53–58 (IEEE, 2024).
7. Akhtar, S. et al. Short-term load forecasting models: A review of challenges, progress, and the road ahead. *Energies* **16**, 4060 (2023).
8. Boopathy, P. et al. Deep learning for intelligent demand response and smart grids: A comprehensive survey. *Comput. Sci. Rev.* **51**, 100617 (2024).
9. Khan, M. A., Saleh, A. M., Waseem, M. & Sajjad, I. A. Artificial intelligence enabled demand response: Prospects and challenges in smart grid environment. *Ieee Access* **11**, 1477–1505 (2022).
10. Graves, A. & Graves, A. Long short-term memory. Supervised sequence labelling with recurrent neural networks, 37–45 (2012).
11. Cho, K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078 (2014).
12. Vaswani, A. et al. Attention is all you need. Advances in neural information processing systems 30 (2017).
13. Kitaev, N., Kaiser, Ł. & Levskaya, A. Reformer: The efficient transformer. arXiv:2001.04451 (2020).
14. Li, S. et al. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Advances in neural information processing systems 32 (2019).
15. Zhou, H. et al. in *Proceedings of the AAAI conference on artificial intelligence*. pp. 11106–11115.
16. Wu, H., Xu, J., Wang, J. & Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Adv. Neural. Inf. Process. Syst.* **34**, 22419–22430 (2021).
17. Alrasheedi, A. & Almalaq, A. Hybrid deep learning applied on Saudi smart grids for short-term load forecasting. *Mathematics* **10**, 2666 (2022).
18. Kim, T.-Y. & Cho, S.-B. Predicting residential energy consumption using CNN-LSTM neural networks. *Energy* **182**, 72–81 (2019).
19. Lai, G., Chang, W. C., Yang, Y., & Liu, H. Modeling long-and short-term temporal patterns withdeep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* 95–104 (2018).
20. Bang, S., Xie, P., Lee, H., Wu, W. & Xing, E. in *Proceedings of the AAAI conference on artificial intelligence*. pp. 11396–11404.

21. Zhou, T. et al. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning* 27268–27286 (PMLR, 2022).
22. Rahmatinia, S. -M., & Seno, S. -A. Prediction of load, price, and wind power generation in smart grids. In *Proc. 2025 29th Int. Electrical Power Distribution Conf. (EPDC)*. https://doi.org/10.1109/EPDC67173.2025.11278281 (2025).
23. Zhang, Y., & Yan, J. Crossformer: Transformer utilizing cross-dimension dependency formultivariate time series forecasting. In *The Eleventh International Conference on Learning Representations* (2023).
24. Sun, B., Chen, X., Shen, T. & Ma, L. Enhancing long-term load forecasting with convolutional informer-based hybrid model. *Eng. Appl. Artif. Intell.* **161**, 112051 (2025).
25. Li, M., Tian, H., Chen, Q., Zhou, M. & Li, G. A hybrid prediction method for short-term load based on temporal convolutional networks and attentional mechanisms. *IET Gener. Transm. Distrib.* **18**, 885–898 (2024).
26. Tang, Z., Ji, T., Kang, J., Huang, Y. & Tang, W. Learning global and local features of power load series through transformer and 2D-CNN: An image-based multi-step forecasting approach incorporating phase space reconstruction. *Appl. Energy* **378**, 124786 (2025).
27. Xu, C. & Chen, G. Interpretable transformer-based model for probabilistic short-term forecasting of residential net load. *Int. J. Electr. Power Energy Syst.* **155**, 109515 (2024).
28. Hong, Q., Meng, F. & Maldonado, F. Advancing long-term multi-energy load forecasting with patchformer: A patch and transformer-based approach. arXiv:2404.10458 (2024).
29. Yanmei, J. et al. Enhanced neighborhood node graph neural networks for load forecasting in smart grid. *Int. J. Mach. Learn. Cybern.* **15**, 129–148 (2024).
30. Lv, Y., Wang, L., Long, D., Hu, Q. & Hu, Z. Multi-area short-term load forecasting based on spatiotemporal graph neural network. *Eng. Appl. Artif. Intell.* **138**, 109398 (2024).
31. Wang, H., Lei, Z., Zhang, X., Zhou, B. & Peng, J. A review of deep learning for renewable energy forecasting. *Energy Convers. Manage.* **198**, 111799 (2019).
32. Hyndman, R. J. & Athanasopoulos, G. Forecasting: Principles and practice. (OTexts, 2018).
33. Jiang, Y. et al. Very short-term residential load forecasting based on deep-autoformer. *Appl. Energy* **328**, 120120 (2022).
34. Papoulis, A. *Random variables and stochastic processes* (McGraw Hill, 1965).
35. Hutter, F., Kotthoff, L. & Vanschoren, J. *Automated machine learning: Methods, systems, challenges* (Springer Nature, 2019).
36. Boullé, M. A parameter-free classification method for large scale learning. *J. Mach. Learn. Res.* **10**, 1367–1385 (2009).
37. Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv:1312.6034.

Author contributions

S.M.R and S.M.H jointly conceptualized the study, developed the methodology, implemented the model, and performed the validation and experimental analysis. S.M.R wrote the original draft of the manuscript. S.A.H.S provided supervision and conceptual guidance for the project. All authors contributed to the review and editing of the final manuscript and have approved the submitted version.

Funding

The authors received no specific funding for this work.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to S.-A.H.-S.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.