Article

Short-Term Electric Load Forecasting Based on Signal Decomposition and Improved TCN Algorithm

Xinjian Xiang [1,*], Tianshun Yuan [1], Guangke Cao [2] and Yongping Zheng [1]

[1] School of Automation and Electrical Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China
[2] Key Laboratory of Intelligent Operation and Maintenance Robot of Zhejiang Province, Hangzhou Shenhao Technology, Hangzhou 311121, China
* Correspondence: 188002@zust.edu.cn; Tel.: +86-15868153622

Abstract: In the realm of power systems, short-term electric load forecasting is pivotal for ensuring supply–demand balance, optimizing generation planning, reducing operational costs, and maintaining grid stability. Short-term load curves are characteristically coarse, revealing high-frequency data upon decomposition that exhibit pronounced non-linearity and significant noise, complicating efforts to enhance forecasting precision. To address these challenges, this study introduces an innovative model. This model employs complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN) to bifurcate the original load data into low- and high-frequency components. For the smoother low-frequency data, a temporal convolutional network (TCN) is utilized, whereas the high-frequency components, which encapsulate detailed load history information yet suffer from a lower fitting accuracy, are processed using an enhanced soft thresholding TCN (SF-TCN) optimized with the slime mould algorithm (SMA). Experimental tests of this methodology on load forecasts for the forthcoming 24 h across all seasons have demonstrated its superior forecasting accuracy compared to that of non-decomposed models, such as support vector regression (SVR), recurrent neural network (RNN), gated recurrent unit (GRU), long short-term memory (LSTM), convolutional neural network-LSTM (CNN-LSTM), TCN, Informer, and decomposed models, including CEEMDAN-TCN and CEEMDAN-TCN-SMA.

Keywords: electric load forecasting; forecasting; complete ensemble empirical mode decomposition with adaptive noise; temporal convolutional network; soft thresholding temporal convolutional network; slime mould algorithm

check for updates

Citation: Xiang, X.; Yuan, T.; Cao, G.; Zheng, Y. Short-Term Electric Load Forecasting Based on Signal Decomposition and Improved TCN Algorithm. *Energies* **2024**, *17*, 1815. https://doi.org/10.3390/en17081815

Academic Editors: Fernando Sánchez Lasheras and Daniel Pérez-López

Received: 12 March 2024
Revised: 3 April 2024
Accepted: 6 April 2024
Published: 10 April 2024

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

The escalating demand for electricity in contemporary society, along with the increasing complexity of power load variations, has heightened requirements for the reliability, economic efficiency, and sustainability of our electricity supply [1]. As a vital component of the electric power industry, load forecasting is categorized into short-term, medium-term, and long-term forecasts, depending on the forecast horizon [2]. Short-term electric load forecasting enables the estimation of electricity demand for the forthcoming hours or days, playing a crucial role in power system operations, generation scheduling, energy market transactions, and strategic system planning.

Short-term electric load forecasting can be divided into two main types: those based on statistical methods and those based on machine learning approaches [3]. Statistical methods include techniques such as moving averages, exponential smoothing, and the autoregressive integrated moving average (ARIMA) model. Lee et al. [4] conducted short-term load forecasting by embedding a lifting scheme into ARIMA models, utilizing the Coeflet 12 wavelet for decomposing and reconstructing power load series to enhance forecasting accuracy. Taylor [5] compared the performance of various univariate exponential smoothing methods in short-term electric load forecasting and discovered that the exponential smoothing method based on singular value decomposition sometimes outperforms weather-based approaches. However, statistical methods tend to perform poorly on nonlinear and complex load variation patterns, struggle with seasonal and holiday effects, and typically require data to be stationary. When the load changes rapidly, statistical methods often remain conservative and fail to keep pace with swift load variations, prompting the emergence of machine learning techniques.

Representative algorithms of traditional machine learning methods include support vector machines (SVMs), decision trees, etc., which predict future load demands by training models on historical load data and other relevant features. Xia et al. [6] proposed an improved fuzzy support vector regression method for power load forecasting, enhancing prediction accuracy through a membership function design and model parameter optimization. X Dong et al. [7] used an SVM method based on K-means clustering for short-term power load forecasting, exploring the impact of temperature and holidays on load, thereby enhancing the accuracy of predictions. Compared to statistical methods, traditional machine learning approaches possess nonlinear modeling capabilities and adapt better to non-stationary data. However, compared to deep learning algorithms, their capacity to model complex nonlinear relationships remains limited, and they typically cannot effectively capture temporal dependencies and sequential relationships in time series data.

In the field of electric load forecasting, widely utilized deep learning algorithms include recurrent neural networks (RNNs), gated recurrent units (GRUs), and long short-term memory (LSTM) networks. Abumohsen et al. [8] employed LSTM, GRU, and RNN deep learning models for electric load forecasting and concluded that the GRU model offered the best performance. GRUs and LSTM networks, as variants of RNNs, have addressed the issues of vanishing and exploding gradients in traditional RNNs by incorporating gating mechanisms, allowing for the better capture and management of long-term dependencies with high levels of representational power and adaptability [9]. However, their disadvantages include a lower computational efficiency, larger parameter sizes, higher level of complexity, and limited modeling capacity for long-term dependencies. Consequently, Liu et al. [10] introduced the temporal convolutional network (TCN), which overcomes these shortcomings through parallel computing and an extended receptive field, offering an efficient, lightweight solution for sequence modeling that is particularly adept at handling long-term dependencies. To further enhance the precision of time series forecasting, Yao et al. [11] proposed a CNN–LSTM hybrid model initially used for traffic flow forecasting. Due to its superior performance, it was extensively applied in load forecasting, as demonstrated by Guo et al. [12], who confirmed its superiority in electric load forecasting. The introduction of numerous hybrid models has continuously improved the accuracy of electric load forecasting. Geng et al. [13] proposed a new forecasting framework that integrates particle swarm optimization (PSO) and variational mode decomposition (VMD) with a TCN equipped with an attention mechanism. The forecasting after the VMD decomposition is better at handling the high levels of randomness and uncertainty typically present in electric loads, especially in capturing minor fluctuations that deep learning algorithms alone may not effectively detect. However, VMD might perform poorly when dealing with noise and non-stationary signals, and integrating algorithms can complicate the deep learning network, making its hyperparameters challenging to adjust. Additionally, Smyl et al. [14] employed a context-augmented, mixed, and hierarchical architecture that integrates exponential smoothing (ES) and an RNN to achieve superior results in short-term load forecasting. This innovative model for short-term load prediction introduces additional information through a contextual track, which is dynamically modulated to adapt to the individual sequences of the main track prediction. Yang [15] et al. introduced a framework combining enhanced K-means and feature selection to tackle limited historical electric load data. Using XGBoost and Bayesian optimization, it achieves precise short-term load forecasting in data-scarce areas through model transfer and parameter adjustments. Nguyen et al. [16] proposed a new model called online SARIMA, specifically designed for short-term load forecasting in power systems. It employs online machine learning techniques to update forecasting parameters in real time, adapting to the seasonal changes and real-time data flows of power loads. Therefore, the model demonstrates significant advantages in enhancing the accuracy of power load forecasting. Liu et al. [17] initially enhanced the nonlinear representation of load data using a feedforward network (FFN), followed by iteratively extracting and exchanging load data information across multiple time resolutions through SCINet to capture deep long-term dependencies. Finally, the extraction of temporal dependencies is further strengthened using an LSTM network. Tarmanini et al. [18] explored short-term electric power load forecasting methods based on an autoregressive integrated moving average (ARIMA) and an artificial neural network (ANN), indicating that ANNs, with their high accuracy in handling nonlinear data and ability to conduct parallel processing for input data, serve as an effective tool in power load forecasting and are superior to traditional statistical methods. In recent years, attention mechanisms have also been widely applied to electric load forecasting. For example, Xu et al. [19] proposed a multi-step power load forecasting method based on the Informer model, which, through a comparative analysis, demonstrates a higher prediction accuracy and efficiency in handling long sequence data than traditional recurrent neural network approaches. However, these methods face multiple challenges when processing power load data, including an insufficient ability to handle complex load variation patterns, limitations in capturing long-term dependencies, low processing efficiency for data with high noise, and high computational complexity and lack of robustness in models. While these methods excel at making overall predictions, they sometimes struggle to capture the nuanced changes within historical data. This oversight can be critical, as the subtle details in historical information often contain key clues about future patterns and anomalies. Therefore, predictive models need not only to absorb large-scale trends but also to be sensitive to the subtleties within the data. By enhancing the model's ability to recognize these fine details, the accuracy of predictions can be significantly improved, especially in complex systems in which even minor variations can have far-reaching effects.

Addressing issues such as the inability of conventional deep learning algorithms to capture subtle changes, the impact of noise in signal decomposition, poor fitting accuracy for high-frequency components, and the difficulty of adjusting deep learning models' hyperparameters, this paper introduces an electric load forecasting method utilizing complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN), a soft thresholding temporal convolutional network (SF-TCN), and the slime mould algorithm (SMA). Initially, the load is decomposed into high- and low-frequency components using CEEMDAN. Subsequently, a TCN is used to forecast both the high- and low-frequency components separately. For the high-frequency components, predictions are obtained through the SF-TCN model, and the SMA is utilized to optimize the network's hyperparameters for these components. For the low-frequency components, predictions are made using the original TCN to obtain the low-frequency forecasting results. Finally, the forecasting results for both the high- and low-frequency components are reconstructed to produce the ultimate prediction outcome.

2. Methodology

2.1. CEEMDAN

CEEMDAN represents an advancement over empirical mode decomposition (EMD) and ensemble empirical mode decomposition (EEMD) [20,21]. EMD is a method for extracting singular and symmetric components from nonlinear and non-stationary signals, dividing the signal into a set of intrinsic mode functions (IMFs), each representing the signal's oscillation patterns at different frequencies [22]. Wu et al. [23] proposed the EEMD decomposition method to address these issues. By adding different types of noise to the original signal for multiple decompositions and then averaging the results, the impact of mode mixing is reduced, enhancing the stability and reliability of the decomposition. However, EEMD may introduce additional noise components during decomposition, leading to less accurate reconstruction. Furthermore, EEMD requires parameters related to the added noise to be set, necessitating some prior knowledge. Torres et al. [19] introduced the CEEMDAN decomposition method, employing an adaptive noise technique that better recovers the signal patterns. Its ability to automatically adjust the noise level also reduces the need for parameter setting.

The CEEMDAN algorithm can be described through the following steps:

1. Add white noise to the original signal $x(t)$ to obtain $x_i(t)$:

$$x_i(t) = x(t) + \beta_0 \omega_i(t) \tag{1}$$

where β_0 is the amplitude coefficient of white noise, and $\omega_i(t)$ is standard white noise.

2. Use EMD to decompose the first set of $x_i(t)$ and obtain the residual $r_1(t)$:

$$IMF_1(t) = \frac{1}{I} \sum_{i=1} IMF_1^i(t) \tag{2}$$

$$r_1(t) = x(t) - IMF_1(t) \tag{3}$$

where $IMF_1(t)$ is the first-order intrinsic mode component, and I is the number of trials.

3. Use EMD on $r_1(t)$ after adding adaptive noise to obtain the second-order modal component $IMF_2(t)$ and the residual $r_2(t)$:

$$r_1(t) + \beta_1 M_1(\omega_i(t)) = IMF_2(t) + r_2(t) \tag{4}$$

where M_1 is the operator of the first-order intrinsic mode component.

4. Repeat step (3), calculating the (k + 1)th order modal component and residual, until the residual becomes a monotonic function and can no longer be further decomposed into IMFs.

2.2. Principles of TCN and the Improved SF-TCN

Unlike typical recurrent neural networks (RNNs), such as GRUs and LSTM networks, TCN is a model based on convolutional neural networks (CNNs) designed for processing time series data. To ensure the output's length mirrors that of the input and to adhere to the principle of preventing future information leakage, TCN employs a 1D fully convolutional structure. This structure incorporates zero padding with a length that is one less than the kernel size and utilizes both causal and dilated convolutions. For a one-dimensional sequence input $\overrightarrow{X} \in \mathbb{R}^n$ and a convolution kernel $f : \{0, \ldots, k-1\} \to \mathbb{R}$, the dilated convolution operation F for the sequence element s is defined as follows [24]:

$$F(s) = (\overrightarrow{X} *_d f)(s) = \sum_{i=0}^{\mathrm{k}-1} f(i) \cdot X_{s-d \cdot i} \tag{5}$$

where $*$ indicates the convolution operation, $\overrightarrow{X}$ represents the input sequence, f represents the convolutional kernel, d represents the dilation rate, and $s - d \cdot i$ accounts for the direction of the past.

In Formula (5), the output at each temporal step s, denoted by $F(s)$, is computed through a dilated convolution operation applied to the input sequence $\overrightarrow{X}$ with the convolution kernel f. This operation integrates information from the current time step and its antecedents. Each element of the convolution kernel $f(i)$ is multiplied by the corresponding element in the input sequence at $X_{s-d \cdot i}$, where d signifies the dilation rate. This rate adjusts the intervals at which the convolution kernel spans the input sequence, thereby enabling the model to capture dependencies across an extended range of data. Consequently, the output $F(s)$ represents the sum of these products, ensuring that each step of the model output is influenced solely by the data preceding its own temporal position. This characteristic is fundamental to time series analysis, as it guarantees the integrity of the predictions by precluding the undue influence of future data.

Causal convolution ensures that the output depends only on the current and previous time points, avoiding leakage of future information. Dilated convolution allows TCN to capture longer historical dependencies by expanding the convolutional operation with a larger receptive field between hidden layers, as shown in Figure 1. In the provided figure, blue circles signify the network's input layer, capturing elements $x(t)$ from sequential data; and hollow circles illustrate the hidden layers, distinguished by depths $d = 1$ and $d = 2$, which are interconnected by horizontal arrows that convey the cyclical propagation of data through time and hierarchical processing stages. The red circles depict the output layer, which yields the predicted outcomes $\hat{y}_t$ at each discrete temporal juncture. Arrows represent the vectors of informational flux and the schematic architecture of the network's connections. This structural design facilitates the maintenance of historical information continuity without preemptive disclosure of future data points, integral to the model's predictive integrity.

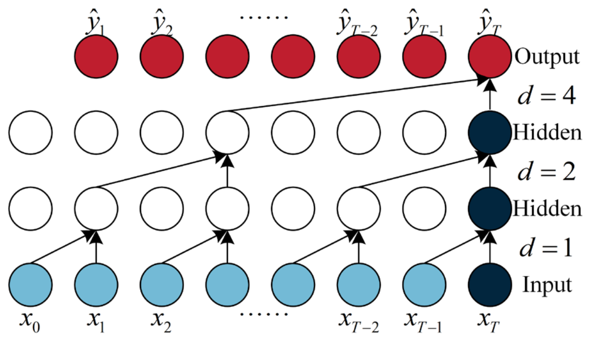

Figure 1. Causal convolution and dilated convolution.

The temporal convolutional network's residual block is ingeniously crafted, comprising convolutional layers, weight normalization layers, ReLU activation functions, and Dropout layers, all meticulously sequenced twice. At the heart of its architecture, a residual connection ingeniously bridges the gap between the input of the first convolutional layer and the output of the second Dropout layer. This innovative linkage not only safeguards the essential information embedded within the input but also facilitates the seamless flow of gradients, effectively circumventing the peril of gradient explosion.

Delving into the realm of high-frequency components, replete with abundant noise, the TCN model exhibits a heightened sensitivity. This susceptibility to noise within the input data can significantly muddle the model's learning trajectory, precipitating inaccuracies in its predictive prowess. The absence of a dedicated noise suppression mechanism within the TCN framework renders it less adept at managing the intricacies of noisy input data, thereby somewhat diminishing its robustness. Notably, the convolutional layers of the TCN model are characterized by a substantial parameter count, particularly when tasked with unraveling the complexities of elongated time series, which, in turn, increases the model's storage requisites and computational intricacy.

In response to the challenges posed by high-frequency components, this discourse introduces an innovative adaptation—the SF-TCN. This refinement integrates threshold filtering parameters designed to meticulously sift through features prior to their emergence from the residual block. Specifically, the soft thresholding operation engages in a meticulous comparison of each feature's magnitude against a pre-established threshold, consequentially nullifying features beneath this threshold while preserving those above. As depicted in Figure 2, the SF-TCN model processes the input sequence $\hat{\mathbf{Z}}^{(i-1)} = (\hat{z}_1^{(i-1)}, \ldots, \hat{z}_T^{(i-1)})$ through a series of operations within the residual blocks, resulting in an output sequence $\hat{\mathbf{Z}}^{(i)} = (\hat{z}_1^{(i)}, \ldots, \hat{z}_T^{(i)})$ that has been refined by the soft thresholding mechanism. This paper applies soft thresholding after the passage through two residual blocks and immediately before engaging the residual connection. The incorporation of threshold filtering parameters in the SF-TCN aims to refine network performance by diligently curating the noise and superfluous information, thereby elevating its proficiency in modeling and processing time series data. This strategic enhancement not only stabilizes the network but also amplifies its generalization capabilities, proving particularly invaluable when navigating the turbulent waters of high-frequency data. Without the mitigating influence of soft thresholding, the TCN model may teeter on the brink of overfitting, a scenario accentuated in conditions of scant training data or an overabundance of model parameters. Lacking a robust feature selection and sparsity mechanism, the model risks becoming ensnared in the intricacies of the training data, culminating in diminished performance on novel test datasets.

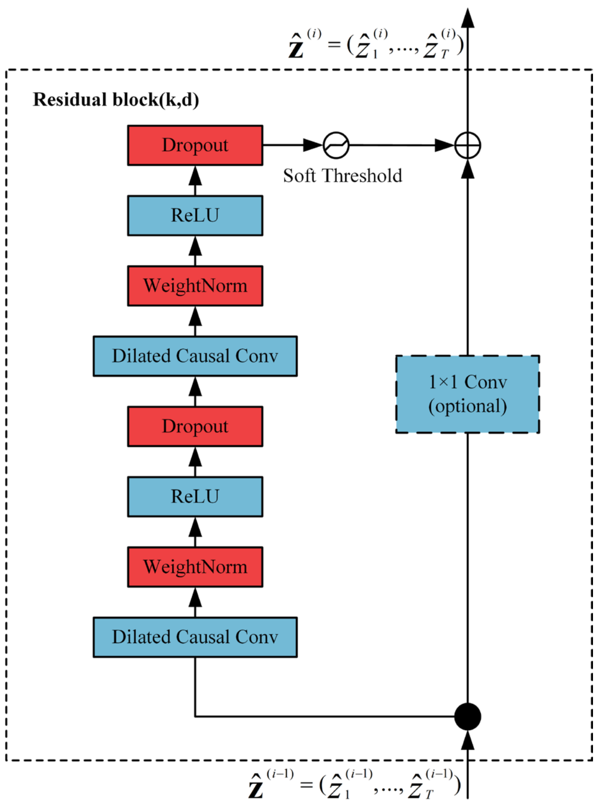
$\hat{\mathbf{z}}^{(i)} = (\hat{z}_1^{(i)}, \ldots, \hat{z}_T^{(i)})$
Residual block(k,d)
Dropout
Soft Threshold
ReLU
WeightNorm
Dilated Causal Conv
1×1 Conv (optional)
Dropout
ReLU
WeightNorm
Dilated Causal Conv
$\hat{\mathbf{z}}^{(i-1)} = (\hat{z}_1^{(i-1)}, \ldots, \hat{z}_T^{(i-1)})$

Figure 2. Soft thresholding residual connection.

Low-frequency component data typically manifest as relatively smooth trends, exhibiting smaller fluctuations and slower changes compared to high-frequency components. Therefore, for low-frequency components, there is no need to employ techniques such as soft thresholding that emphasize noise filtering. Their inherent stability and gradual change characteristics make them easier to handle during analysis and modeling processes, eliminating the need for additional signal processing steps.

2.3. Principles of SMA

Slime mold can select foraging strategies based on the quality of food sources and can utilize multiple food sources simultaneously. Applying this adaptive search behavior, through which the mold dynamically adjusts its search patterns based on current information, to optimization algorithms can achieve strong global search capabilities, adaptability, flexibility, a balance between exploration and exploitation, and multi-objective optimization capabilities. Compared to other commonly used optimization algorithms, such as the marine predators algorithm (MPA) and the salp swarm algorithm (SSA), the SMA algorithm exhibits superior levels of precision and stability, hence its selection for use in this paper [25].

The growth and behavior of slime molds can begin to be understood with the formation of plasmodia following mitosis, entering a mature phase in which slime molds start their trophic growth stage, exhibiting a network-like morphology. This morphological transformation is primarily aimed at maximizing the absorption of essential nutrients such as food, water, and oxygen by adjusting their surface area to adapt to the distribution of nutrients in the environment. In slime mold algorithms, the behavior of these organisms is utilized to simulate an intelligent optimization process, in which the slime mold adjusts the positional weight of each individual based on the environmental conditions (i.e., the quality of the fitness function) at its location, thereby determining its new direction of movement.

As slime molds approach a food source, biological oscillators are activated, generating waves through their venous system to increase the flow of substances within cells. The intensity of these waves correlates positively with food concentration; that is, the higher the concentration of food, the stronger the waves, leading to faster internal substance flow. By this means, the slime mold algorithm mimics the predatory behavior of slime molds, achieving an intelligent function for finding optimal solutions.

From the natural behavior of slime molds, three basic rules can be abstracted to simulate their foraging process: firstly, slime molds can move towards a food source by sensing odors in the air, forming circular or fan-shaped movement structures; secondly, when the venous system of a slime mold contacts a high concentration of food, its internal biological oscillator generates stronger waves, thereby accelerating the flow of cytoplasm to quickly enclose the food; and lastly, slime molds adjust their movement speed according to the concentration of food, approaching low-concentration food slowly, while accelerating towards high-quality food sources. These rules together guide the behavioral patterns of slime molds in nature and provide a theoretical basis for the slime mold algorithm.

The implementation process of the slime mold algorithm is as follows [26]:

1. Approaching food

$$\overrightarrow{X(t+1)} = \begin{cases} \overrightarrow{X_b(t)} + \overrightarrow{vb} \cdot (\overrightarrow{W} \cdot \overrightarrow{X_A(t)} - \overrightarrow{X_B(t)}), r < p \\ \overrightarrow{vc} \cdot \overrightarrow{X(t)}, r \geq p \end{cases} \tag{6}$$

where $\overrightarrow{vb} \in [-a, a]$, $\overrightarrow{vc}$ is a parameter that gradually decreases from 1 to 0; t represents the current iteration number; $\overrightarrow{X_b}$ indicates the position of the individual with the highest odor concentration, i.e., the position of the individual with the optimal fitness; $\overrightarrow{X}$ represents the current position of the slime mold individual; $\overrightarrow{X_A}$ and $\overrightarrow{X_B}$ represent the positions of two randomly chosen slime mold individuals; and $\overrightarrow{W}$ is a weight coefficient. The updated formula for p is as follows:

$$p = \tanh|S(i) - DF| \tag{7}$$

where $i \in 1, 2, 3, \cdots n$ represents the individual index in the slime mold population; $S(i)$ represents the fitness of individual $\overrightarrow{X}$; and DF indicates the best fitness obtained across all iterations. The formula of $\overrightarrow{vb}$ is $\overrightarrow{vb} = [-a, a]$, and its updated formula is as follows:

$$a = \operatorname{arctan}\mathrm{h}(-(\frac{t}{\max\_t}) + 1) \tag{8}$$

The updated formula for $\overrightarrow{W}$ is as follows:

$$\overrightarrow{W(SmellIndex(i))} = \begin{cases} 1 + r \cdot \log(\frac{bF - S(i)}{bF - wF} + 1), \text{condition} \\ 1 - r \cdot \log(\frac{bF - S(i)}{bF - wF} + 1), \text{others} \end{cases} \tag{9}$$

$$SmellIndex = sort(S) \tag{10}$$

where $r \in [0,1]$, bF represents the best fitness achieved during the current iteration process; wF denotes the worst fitness achieved during the current iteration process; condition indicates that $S(i)$ ranks in the top half of the group; and $SmellIndex$ is a sequence of fitness values arranged in ascending order, used when addressing minimization problems.

2. Encircling Food Depending on the quality of the food, slime mold can adjust its search patterns. When the food concentration is high, it places more emphasis on that area; conversely, when the food concentration is low, it reduces the weight of that area and turns to explore other regions. The mathematical formula for updating the position of the slime mold is as follows:

$$\overrightarrow{X^*} = \begin{cases} rand\cdot(UB - LB) + LB, rand < z \\ \overrightarrow{X_b(t)} + \overrightarrow{vb} \cdot (W\cdot\overrightarrow{X_A(t)} - \overrightarrow{X_B(t)}), r < p \\ \overrightarrow{vc} \cdot \overrightarrow{X(t)}, r \geq p \end{cases} \tag{11}$$

where UB and LB are the upper and lower bounds of the hyperparameter search, respectively; $rand, r \in [0,1]$,z acts as a control factor, which adjusts the balance between global search and local exploitation.

3. Capturing Food Slime mold employs biological oscillators to generate propagation waves that alter the flow of its cytoplasm, enabling it to search for and select food resources within its environment. By adjusting its oscillation frequency and engaging in random exploratory behavior, the slime mold adapts to varying concentrations of food, allowing the cells to more swiftly converge upon sources of high-quality food, while allocating a portion of its resources to the exploration of additional areas. The oscillation of parameters and their synergistic effect simulate the selective behavior of slime mold, empowering it to discover superior food sources and circumvent local optima. Despite facing numerous constraints during propagation, these very limitations afford the slime mold opportunities, enhancing its likelihood of locating high-quality food sources.

The optimization of hyperparameters represents a complex global optimization challenge, for which the slime mold algorithm offers an effective solution for the hyperparameter optimization of high-frequency models. It not only assists in identifying the optimal combination of hyperparameters, enhancing model performance, but also boosts the model's robustness and generalization capabilities.

2.4. CEEMDAN-SF-TCN-SMA

The model proposed in this paper utilizes CEEMDAN to decompose the load signal as shown in Figure 3, extracting and reconstructing the first two IMFs as high-frequency signals and reconstructing the remaining IMFs as low-frequency signals, as depicted in Figure 4. These figures only display the electric load data for the spring month of April. Given the high complexity and low accuracy of high-frequency signal prediction, and considering that low-frequency signal curves are relatively smooth, without the need to address high noise and volatility, soft thresholding does not significantly impact them. Therefore, the SF-TCN model will be applied solely to high-frequency signals, while the SMA algorithm will be used for hyperparameter optimization of both high- and low-frequency models. Figure 5 illustrates the specific process of the model method proposed in this paper.

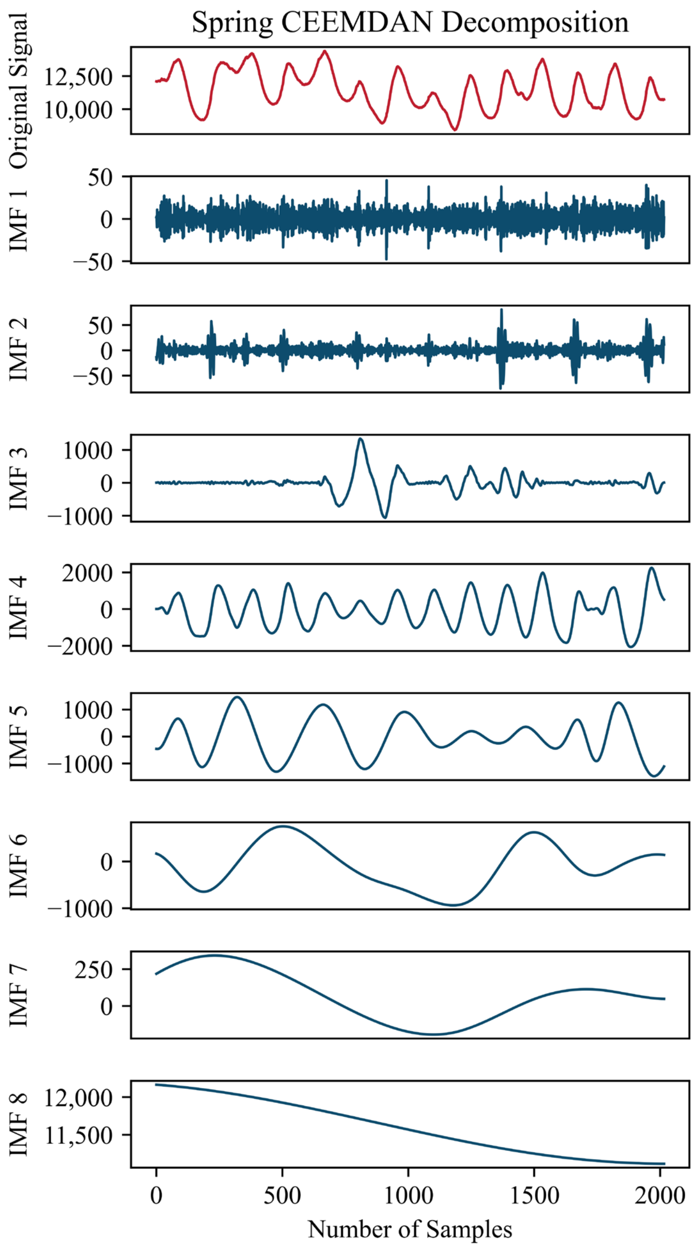

Figure 3. Spring electricity load CEEMDAN decomposition results.

By decomposing time series data into high-frequency and low-frequency signals for reconstruction, this strategy offers multifaceted advantages in prediction, significantly enhancing our ability to handle complex data. Firstly, it markedly reduces the adverse impact of noise on forecast accuracy by specifically addressing the noise and fluctuations in high-frequency signals, while choosing the most suitable model for the unique characteristics of different signals, thereby improving overall prediction precision. Moreover, processing high-frequency and low-frequency signals separately reduces the risk of model overfitting, enhancing the model's generalization ability when faced with new, unseen data. In terms of computational efficiency, this method allows for the allocation of different computational resources and parallel processing strategies for different signals, effectively saving computation time and resources. Additionally, by clearly distinguishing between the high-frequency and low-frequency components of the data, it becomes easier to understand and interpret the model's prediction results, facilitating adjustments and optimizations to the model as necessary. Lastly, this approach offers a high degree of flexibility and adaptability, enabling the model to adjust its focus on high-frequency or low-frequency signals based on the specific requirements of the application scenario, effectively coping with changes in data characteristics over time and the predictive needs of different scenarios. In summary, the method of dividing data into high-frequency and low-frequency signals for reconstruction ensures prediction accuracy, enhances efficiency, improves interpretability, and provides a high level of adaptability to complex data situations.

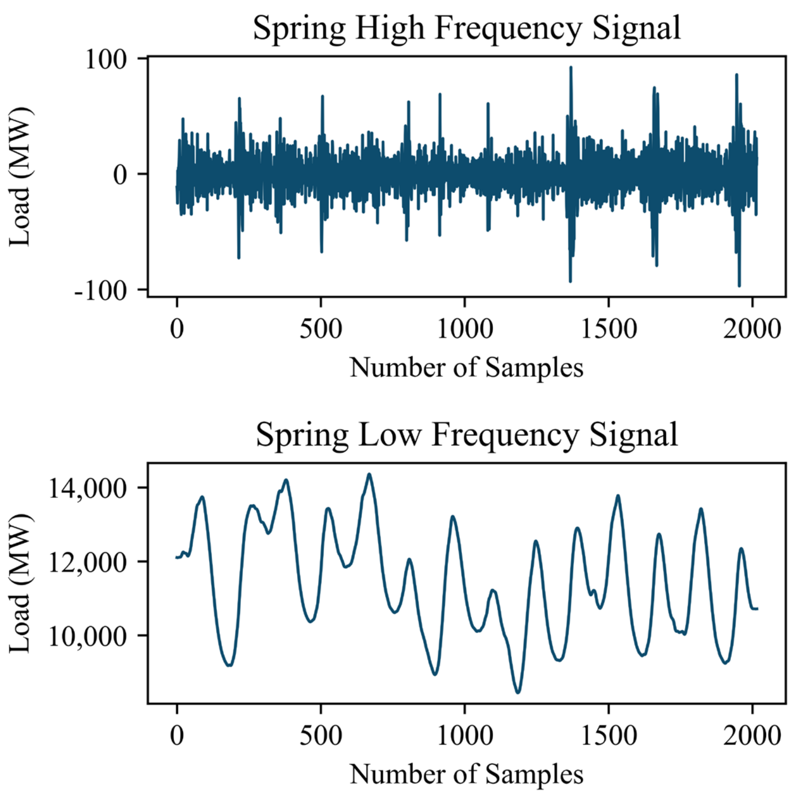

Figure 4. Spring high- and low-frequency signal classification results.

For the load signal, CEEMDAN is utilized to decompose it into multiple IMFs. The results of CEEMDAN decomposition automatically arrange the IMFs in descending order of frequency. Hence, the first two IMFs are reconstructed into a high-frequency signal, while the remaining IMFs are reconstructed into a low-frequency signal. The issues of diminished precision in handling high-frequency components are primarily attributed to the structural characteristics of TCNs and their sensitivity to noise. High-frequency data features rapid changes along with higher levels of noise, which complicates the TCN models' ability to learn effective features from it. Without specialized noise filtering or feature discrimination mechanisms, TCNs might struggle to differentiate valuable information from noise effectively. This leads to the model mistaking noise for valuable signals and learning from it, thereby affecting the predictive performance. This suggests that TCNs are inherently more suited to processing low-frequency data with more stable feature variations. In contrast, their ability to handle high-frequency components with drastic feature changes is limited, as reflected in their modeling and predictive accuracy for such data. For predicting high-frequency components, the SF-TCN-SMA model is employed to obtain the high-frequency prediction results. In predicting the low-frequency signal, TCN-SMA is directly used to achieve the low-frequency prediction results. Finally, the high-frequency and low-frequency prediction results are reconstructed to obtain the final prediction outcome.

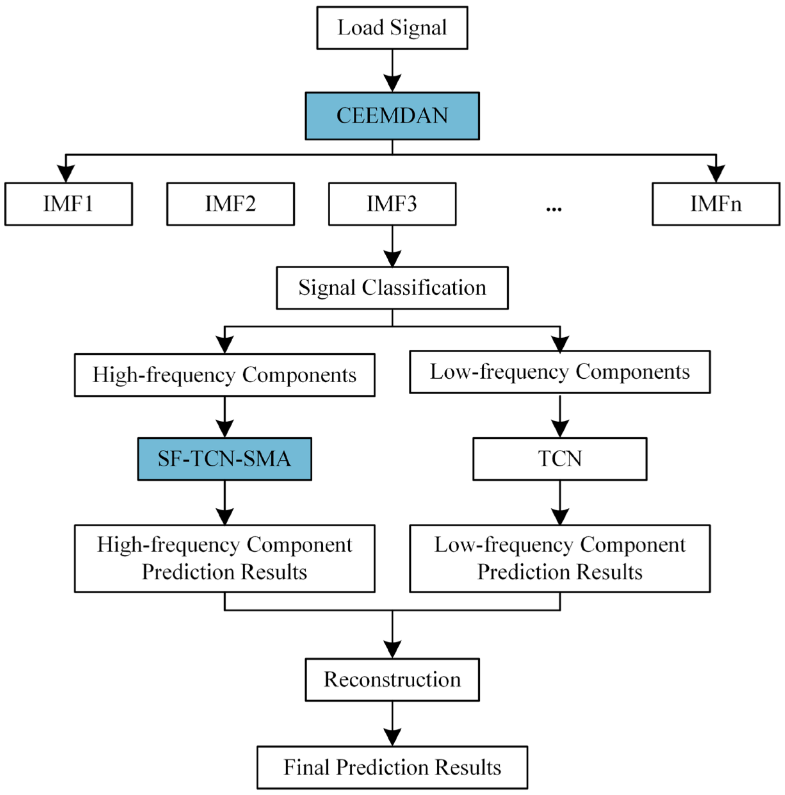

Figure 5. CEEMDAN-SF-TCN-SMA model.

3. Data Sources and Preprocessing

3.1. Data Sources

This study utilizes the electric load data of the New England area in the United States for the year 2021, which are available from the website ISO New England, an independent organization responsible for managing the power system in the New England region. The dataset has a time step of 5 min and is used for forecasting one day ahead. To verify the model's robustness and stability, data from the first 7 days of representative months for each of the four seasons were selected for the New England area, with April representing the spring, July representing the summer, October representing the autumn, and January representing the winter. For each month, the first 5 days were used as the training set, the 6th day as the validation set, and the 7th day as the test set.

3.2. Data Preprocessing

To ensure the robustness and stability of the forecast results, missing values and outliers exceeding a difference of 600 in the dataset were addressed using the random forest algorithm [27]. The processed dataset and its segmentation are depicted in Figure 6, with green representing the training set, blue representing the validation set, and red representing the test set. To enhance the neural network's performance, the data underwent extreme value normalization, with the formula as follows:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{12}$$

where x' is the value after normalization, x are the current data to be normalized, and $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of all the data that need to be normalized, respectively.

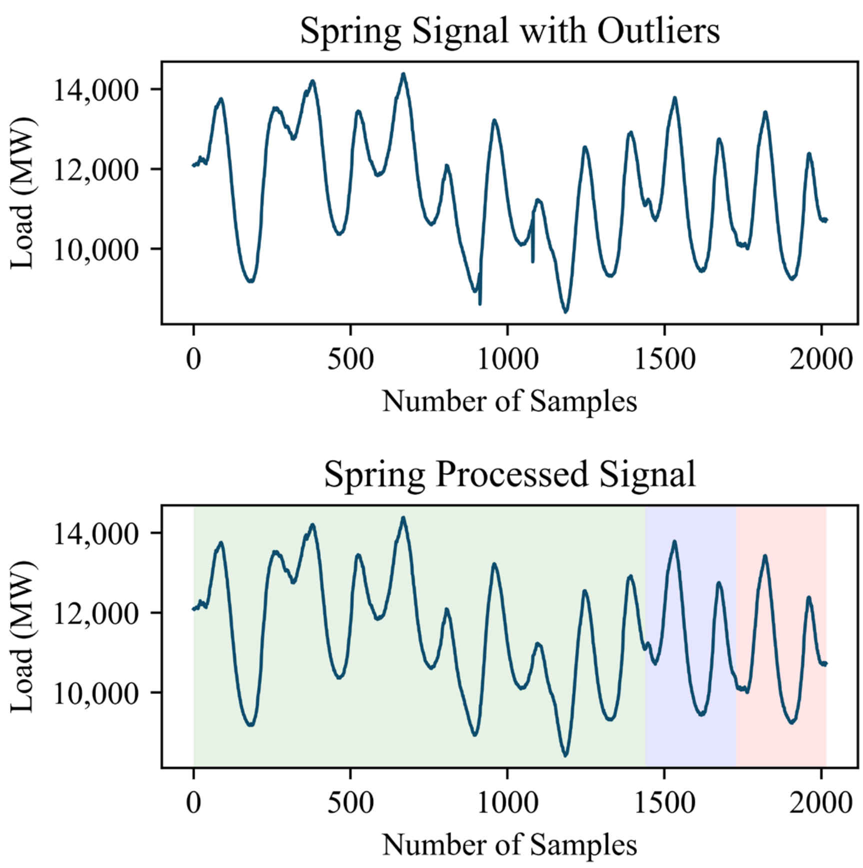

Figure 6. Error point handling and data set splitting.

4. Experiment and Results Analysis

4.1. Model Configuration and Evaluation Metrics

To validate the effectiveness and superiority of the proposed CEEMDAN-SF-TCN-SMA model, comparative experiments and ablation studies were conducted. The comparative experiments employed SVR, RNN, GRU, LSTM, and CNN-LSTM models, while the ablation studies utilized TCN, CEEMDAN-TCN, and CEEMDAN-TCN-SMA models to verify the effectiveness of CEEMDAN, SMA, and SF-TCN, respectively.

The proposed model's high- and low-frequency components each comprise two stacked TCN layers, followed by a Flatten layer and two Dense layers. For the low-frequency component, the SMA optimization algorithm optimizes six hyperparameters: the number of filters in the two TCN layers (nb_filters), the kernel size (kernel_size), the number of neurons in the first Dense layer (dense_units), and the batch size (batch_size). The high-frequency component additionally includes optimization for the soft threshold. The dilations and epochs are not subject to optimization and are set at [1,2,4,8] and 100, respectively. The population size (pop) of the SMA optimization algorithm is set to 15, with a maximum iteration number (MaxIter) of 10. The hyperparameters and optimization results for the high- and low-frequency components are presented in Table 1. The optimization results for the low-frequency component are identical across all four seasons, with the nb_filters and kernel_size for both of the TCN layers being 16 and three, respectively, and the dense_units and batch_size being 16. The optimization results for the high-frequency component vary across seasons, as shown in Table 2.

Table 1. CEEMDAN-SF-TCN-SMA model parameters.

Network Layer	Network Parameters	Low Frequency	High Frequency
First Layer TCN	nb_filters	16	Refer to Table 2
	kernel_size	3	Refer to Table 2
	dilations	[1,2,4,8]	[1,2,4,8]
	activation	ReLU	ReLU
	threshold	No Parameter	Refer to Table 2
Second Layer TCN	nb_filters	16	Refer to Table 2
	kernel_size	3	Refer to Table 2
	dilations	[1,2,4,8]	[1,2,4,8]
	activation	ReLU	ReLU
	threshold	No Parameter	Refer to Table 2
First Layer Dense	dense_units	16	Refer to Table 2
	activation	ReLU	ReLU
Second Layer Dense	dense_units	1	1
	activation	Linear	Linear
Training Parameters	batch_size	16	Refer to Table 2
	epochs	100	100

Table 2. High-frequency parameters.

Hyperparameters	Spring	Summer	Autumn	Winter
nb_filters_1	220	223	218	27
kernel_size_1	4	7	8	6
threshold_1	0.043	0.969	0.221	0.008
nb_filters_2	81	250	150	187
kernel_size_2	5	6	9	6
threshold_2	0.224	0.964	0.189	0.556
dense_units_1	44	21	40	33
batch_size	39	104	111	52

The parameter settings for each model in the comparative experiments are shown in Table 3. The SVR model uses the radial basis function (RBF) kernel, with a regularization coefficient (C) of 1.5, a kernel coefficient (gamma) of 0.45, and a tolerance coefficient (epsilon) of 0.01. The RNN, GRU, and LSTM models each feature two stacked layers, followed by a Flatten layer and two Dense layers. The CNN-LSTM model comprises both CNN and LSTM components: the CNN part includes a Reshape layer, Conv2D layer, MaxPooling2D layer, Dropout layer, and another Reshape layer, while the LSTM part consists of two LSTM layers, a Flatten layer, and two Dense layers. For the Informer model, which is tailored for enhanced long-sequence time series forecasting, the precise configuration of its parameters plays a vital role. The model dimension is set to 512 (d_model), providing a substantial representation capacity. It utilizes eight heads (n_heads) in its multi-head attention mechanism, facilitating the parallel processing of sequence information. The architecture comprises two encoder layers (e_layers) and one decoder layer (d_layers), establishing a balanced depth for processing and prediction. A dropout rate of 0.05 (dropout) is chosen to mitigate overfitting by randomly omitting features during training. The attention mechanism specified is the probabilistic sparse attention mechanism (attn='prob'), which is optimized for handling long sequences, while the time features are encoded using a time feature encoding method (embed='timeF'). These parameter settings collectively define the Informer model's structure, equipping it to process long time series data efficiently and with a high level of predictive accuracy.

Table 3. Comparison experiment's model parameters.

Model	Network Parameters
SVR	kernel='rbf', C=1, gamma=0.5, epsilon=0.01

RNN	hidden_units_1=16, hidden_activation_1='relu'
	hidden_units_2=16, hidden_activation_2='relu'
	dense_units_1=16, dense_activation_1='relu',
	dense_units_2=1, dense_activation_2='linear',
	batch_size=16, epochs=100

GRU	hidden_units_1=16, hidden_activation_1='relu'
	hidden_units_2=16, hidden_activation_2='relu'
	dense_units_1=16, dense_activation_1='relu',
	dense_units_2=1, dense_activation_2='linear',
	batch_size=16, epochs=100

LSTM	hidden_units_1=140, hidden_activation_1='relu'
	hidden_units_2=60, hidden_activation_2='relu'
	dense_units_1=16, dense_activation_1='relu',
	dense_units_2=1, dense_activation_2='linear',
	batch_size=64, epochs=100

CNN-LSTM	filters=64, kernel_size=3, strides=1,
	pool_size=2,
	dropout=0.3,
	hidden_units_1=140, hidden_activation_1='relu',
	hidden_units_2=60, hidden_activation_2='relu',
	dense_units_1=16, dense_activation_1='relu',
	dense_units_2=1, dense_activation_2='linear',
	batch_size=64, epochs=100

Informer	seq_len=12, label_len=6, pred_len=1,
	enc_in=1, dec_in=1, c_out=1, d_model=512,
	n_heads=8, e_layers=2, d_layers=2, s_layers='3, 2, 1',
	d_ff=2048, fator=5, padding=0, distill='store_false',
	dropout=0.05, attn='prob', embed='timeF', activation='gelu',
	output_attention='store_true', do_predict='store_true',
	mix='store_false', cols='+', num_workers=0, itr='2',
	train_epochs=6, batch_size=32, patience=3, learning_rate=0.001,
	des='test', loss='mse', lradj='type1', use_amp='store_true', inverse=True

In the ablation experiments, the TCN model consists of two stacked TCN layers, followed by a Flatten layer and two Dense layers. The network structure of the low-frequency part and the hyperparameters optimized by the SMA for CEEMDAN-TCN and CEEMDAN-TCN-SMA are consistent with those proposed in this paper. The parameters for the high-frequency component, including the number of filters (nb_filters), kernel size (kernel_size), number of neurons in the first Dense layer (dense_units), and batch size (batch_size), are presented in Table 4. All experiments in this paper employ a sliding window size of 12 to predict the next data point. In the tables of this paper, "_1" denotes the network's first layer, and "_2" indicates the second layer.

Model performance can be evaluated using the following metrics: the mean square error (MSE), mean absolute percentage error (MAPE), and mean absolute deviation (AbsDEV), with their formulas as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{13}$$

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{14}$$

$$AbsDEV = \frac{1}{\sum_{i=1}^{n} y_i} \cdot \sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{15}$$

where y_i represents the actual value, $\hat{y}_i$ represents the predicted value, and n is the number of samples.

Table 4. Ablation experiment parameters.

Model	Network Parameters

CEEMDAN-TCN High-frequency Component	nb_filters_1=32, kernel_size_1=3,
	hidden_activation_1='relu',
	dilations_1=[1,2,4,8],
	nb_filters_2=32, kernel_size2=3,
	hidden_activation2='relu',
	dilations2=[1,2,4,8],
	dense_units1=16, dense_activation1='relu',
	dense_units2=1, dense_activation2='linear',
	batch_size=16, epochs=100

CEEMDAN-TCN-SMA High-frequency Component	nb_filters_1=Optimization, kernel_size_1=Optimization,
	hidden_activation_1='relu',
	dilations_1=[1,2,4,8],
	nb_filters_2=Optimization, kernel_size_2=Optimization,
	hidden_activation_2='relu',
	dilations_2=[1,2,4,8],
	dense_units_1=Optimization, dense_activation_1='relu',
	dense_units_2=1, dense_activation_2='linear',
	batch_size=Optimization, epochs=100

TCN	nb_filters_1=64, kernel_size_1=3,
	hidden_activation_1='relu',
	dilations_1=[1,2,4,8],
	nb_filters_2=64, kernel_size_2=3,
	hidden_activation_2='relu',
	dilations_2=[1,2,4,8],
	dense_units_1=16, dense_activation_1='relu',
	dense_units_2=1, dense_activation_2='linear',
	batch_size=16, epochs=100

4.2. CEEMDAN-SF-TCN-SMA Forecasting Analysis

Using CEEMDAN-SF-TCN-SMA to predict short-term electric loads one day in advance resulted in a commendable level of accuracy. The overall evaluation metrics before decomposition are shown in Table 5, while the overall, high-frequency, and low-frequency evaluation metrics after decomposition are presented in Table 6, Table 7, and Table 8, respectively. In all tables, dark blue indicates the best evaluation metrics, followed by light blue.

In Table 6, experiments show that the CEEMDAN-SF-TCN-SMA model has a better prediction accuracy in the spring and autumn than in the summer and winter. As depicted in Figure 7, both the summer and winter exhibit a higher level of electricity consumption, especially the summer. Higher electricity usage often accompanies more complex changing trends, resulting in increased data noise and a decrease in prediction accuracy. Overall, the model proposed in this study achieves a coefficient of determination (R^2) greater than 0.999 across all seasons, as shown in Figure 7, indicating an excellent load curve fitting accuracy and reflecting its good prediction accuracy and robustness.

Table 5. Overall evaluation metrics before CEEMDAN.

Season	Metric	SVR	RNN	GRU	LSTM	CNN-LSTM	Informer	TCN
Spring	MSE	1088.32	1770.20	3959.82	2138.16	1807.81	6264.42	1690.01
	MAPE (%)	0.23	0.30	0.43	0.32	0.33	0.57	0.30
	AbsDEV	25.59	33.51	48.67	35.82	34.97	62.91	32.53
Summer	MSE	8212.09	8194.42	8385.86	8278.83	7661.2	16,603.23	5412.53
	MAPE (%)	0.45	0.38	0.39	0.41	0.49	0.65	0.32
	AbsDEV	73.62	65.11	65.14	68.15	77.45	108.38	54.63
Autumn	MSE	1198.91	1336.82	1351.08	1659.37	1408.35	6762.06	1142.42
	MAPE (%)	0.22	0.24	0.24	0.27	0.25	0.52	0.22
	AbsDEV	26.17	28.33	28.24	32.16	29.51	62.20	25.97
Winter	MSE	2295.14	4623.00	3023.68	2910.03	2449.34	7700.20	2109.48
	MAPE (%)	0.22	0.39	0.26	0.25	0.27	0.45	0.23
	AbsDEV	31.87	55.90	37.25	34.84	37.59	63.60	33.88

Table 6. Overall evaluation metrics after CEEMDAN.

Season	Metric	CEEMDAN-TCN	CEEMDAN-TCN-SMA	CEEMDAN-SF-TCN-SMA
Spring	MSE	971.27	862.96	628.42
	MAPE (%)	0.23	0.21	0.18
	AbsDEV	25.51	23.81	19.67
Summer	MSE	3698.86	3609.73	3442.13
	MAPE (%)	0.28	0.27	0.27
	AbsDEV	46.03	45.36	44.39
Autumn	MSE	1003.2	745.43	555.45
	MAPE (%)	0.20	0.17	0.15
	AbsDEV	24.23	20.99	18.17
Winter	MSE	1167.29	1081.12	1014.45
	MAPE (%)	0.18	0.17	0.16
	AbsDEV	24.84	23.41	22.05

Table 7. Low-frequency component evaluation metrics.

Hyperparameters	Metric	Value
Spring	MSE	421.00
	MAPE (%)	0.15
	AbsDEV	16.47
Summer	MSE	371.47
	MAPE (%)	0.26
	AbsDEV	43.77
Autumn	MSE	343.39
	MAPE (%)	0.11
	AbsDEV	13.60
Winter	MSE	202.10
	MAPE (%)	0.09
	AbsDEV	11.43

Table 8. High-frequency component evaluation metrics.

Season	Metric	CEEMDAN-TCN	CEEMDAN-TCN-SMA	CEEMDAN-SF-TCN-SMA
Spring	MSE	374.18	357.31	342.88
	MAPE (%)	470.49	394.08	256.40
	AbsDEV	15.06	14.73	14.49
Summer	MSE	477.5	454.64	331.19
	MAPE (%)	337.54	282.115	250.49
	AbsDEV	16.46	16.0916	13.87
Autumn	MSE	428.82	330.91	269.38
	MAPE (%)	297.26	272.27	205.91
	AbsDEV	16.16	14.29	12.89
Winter	MSE	1012.09	923.01	788.92
	MAPE (%)	753.28	483.9	311.43
	AbsDEV	22.84	20.15	18.30

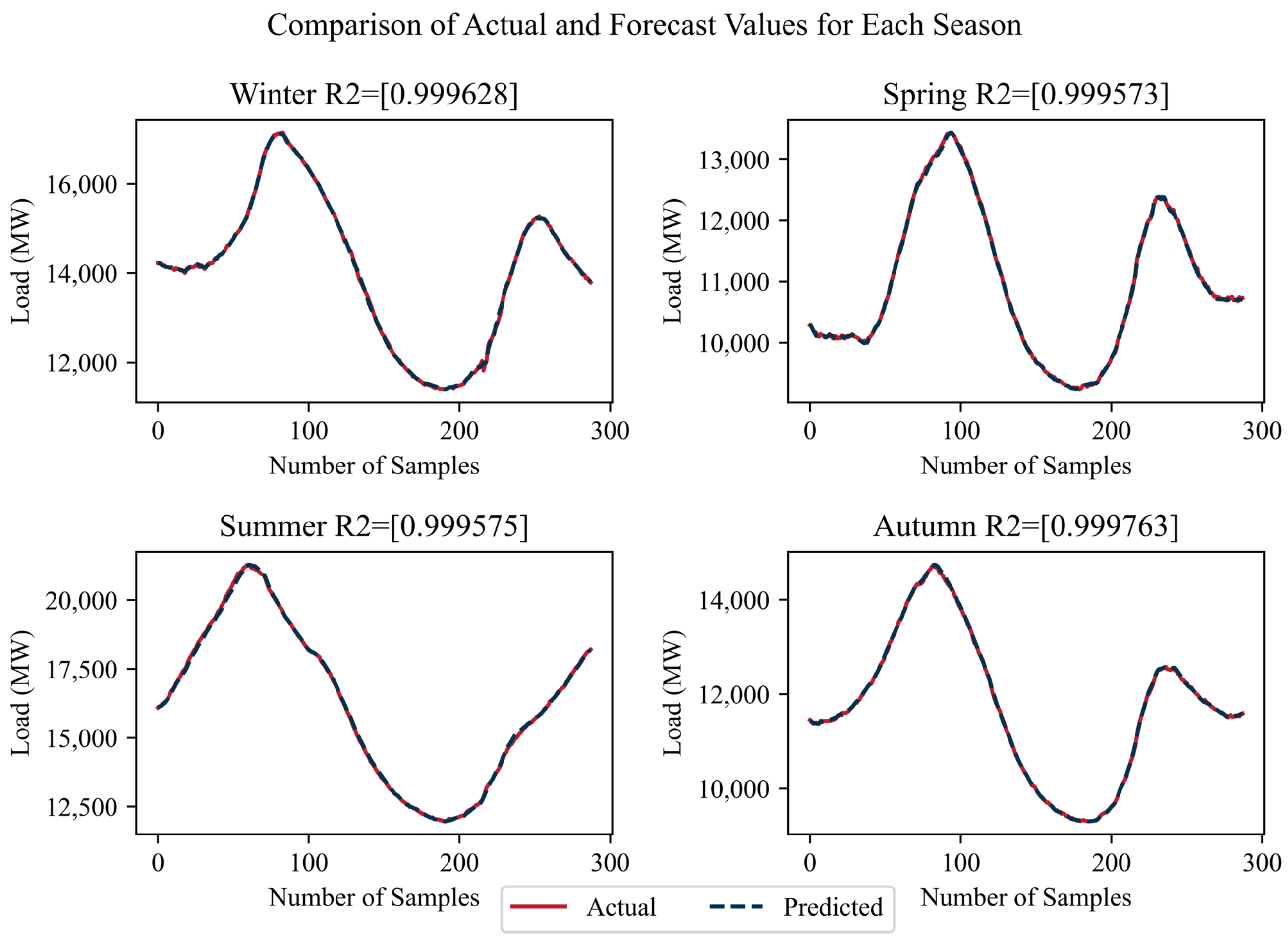

Figure 7. Comparison of actual and forecast values for each season.

4.3. Comparative Analysis

Table 5 compares the prediction effects before the CEEMDAN model's decomposition. All models were evaluated using the MSE, MAPE, and AbsDEV. In Tables 5, 6 and 8. Among them, the TCN model performed better in the summer, autumn, and winter seasons, showing a superior overall performance. The prediction results of the CNN-LSTM and SVR models also showed a good performance. The SVR model performed best in spring, but its prediction accuracy in summer was significantly lower than that of the TCN and CNN-LSTM models. The Informer, based on the attention mechanism, performs the worst in all seasons and across all parameters.

The performance differences among various time series models when dealing with electricity load data characterized by volatility and cyclicality can be attributed to their unique information processing methods and structural designs. The SVR, as a statistical-based machine learning model, performs well with electricity load data that have certain linear relationships. However, it may underperform when facing complex nonlinear fluctuations and cyclic patterns. RNNs, along with its variants GRUs and LSTM networks, are designed to capture time dependencies in data. RNNs are limited by the vanishing and exploding gradient problems, making them less effective in capturing long-term dependencies. GRUs and LSTM networks introduce gating mechanisms to address this issue, enhancing efficiency in learning from long-sequence data. Despite this, they might still struggle with particularly complex cyclic and volatile patterns due to constraints in parameter configurations and network structures. The CNN-LSTM combines the spatial feature extraction capabilities of CNNs with the long-term dependency handling ability of the LSTM, adapting well to fluctuations and cyclicality. The Informer, utilizing the self-attention mechanism based on the Transformer architecture, theoretically captures dependencies across the entire sequence, making it suitable for long-term series prediction. However, its performance may vary with the specificity of electricity load data and the scale of training data. The TCN effectively captures long-distance dependencies in time series through stacked convolutional layers and is especially adept at recognizing and leveraging cyclic patterns for accurate predictions in data with significant fluctuations and cyclicality.

The experiments in Table 6 decomposed the original data using CEEMDAN. Table 7 uses the TCN model for low-frequency predictions. The hyperparameters of different models in different seasons remain the same before and after the SMA optimization, resulting in consistent outcomes. Separately predicting high- and low-frequency data after decomposition and then reconstructing them significantly improved the prediction accuracy. Compared to the best-performing undecomposed TCN model, the decomposed CEEMDAN-TCN model saw average reductions in the MSE, MAPE, and AbsDEV of 878.455, 0.045, and 6.600, respectively, with particularly noticeable reductions in the summer, when these three evaluation metrics decreased by 1713.670, 0.040, and 8.600, respectively. Decomposition, prediction, and then reconstruction allow for layered modeling methods to independently process features within different frequency ranges, more accurately capturing rapid changes, stable trends, and periodicity. Predicting high- and low-frequency parts separately allows for the selection of suitable prediction models and hyperparameters according to their different characteristics, improving the prediction accuracy and stability and effectively reducing the impact of noise on the prediction results.

In contrast, the results for the low-frequency component, in which the hyperparameters of each model in different seasons remained the same before and after the SMA optimization, differed for the high-frequency component after the SMA optimization, changing the hyperparameters for different seasons to adapt to varying demands. After optimizing the high-frequency hyperparameters using the SMA algorithm to simulate the slime mold foraging process, the MSE, MAPE, and AbsDEV for the high-frequency component had average reductions of 56.573, 106.550, and 1.313, respectively, and the overall prediction values after reconstruction saw reductions of 135.345, 0.018, and 1.760, respectively, in these three evaluation metrics.

The high non-linearity and strong noise of the high-frequency component made fitting with the TCN less accurate, whereas the SF-TCN could suppress noise components in the high-frequency part, retain important detail information, enhance model robustness, and smooth data fluctuations. The application of the SF-TCN improved the fitting accuracy of the high-frequency component, enabling the model to better capture the true patterns of change in the original signal, thereby improving the overall prediction accuracy and stability. Table 8 shows the prediction results for the high-frequency component, revealing that after using the SF-TCN-SMA, the MSE, MAPE, and AbsDEV decreased on average by 76.603, 101.338, and 1.270, respectively, compared to the TCN-SMA, and the overall prediction values after reconstruction saw decreases of 164.698, 0.015, and 2.322, respectively, in these three evaluation metrics, reflecting the effectiveness of the improved SF-TCN.

In conclusion, the methodologies proposed in this paper, encompassing decomposition, prediction followed by reconstruction, and the utilization of the enhanced SF-TCN and the slime mold optimization algorithm, represent innovative and effective approaches.

The CEEMDAN-SF-TCN-SMA model proposed in this paper performed the best among the compared models. Decomposing the original data and using the SF-TCN for the high-frequency component were effective, as was the application of the SMA optimization algorithm.

5. Conclusions

This study introduces a novel approach to short-term electric power load forecasting, which integrates the CEEMDAN, an improved SF-TCN, and the SMA for hyperparameter optimization. Compared to traditional models, this method demonstrates superior predictive accuracy, marking progress in addressing the complexity of short-term electric power load forecasting.

This paper's primary contributions are as follows:

1. By utilizing the CEEMDAN, our study effectively decomposes electric power load data into high-frequency and low-frequency components. This enables a more detailed analysis, capturing subtle fluctuations in the load curve that traditional methods may overlook.
2. The introduction of an improved SF-TCN addresses the challenges in predicting high-frequency components. This model enhancement not only reduces the impact of noise but also improves the accuracy of short-term forecasts.
3. The application of the shuffled memetic algorithm (SMA) for adjusting the neural network's hyperparameters and soft thresholding enhances the neural network's adaptability and forecasting ability.
4. Our experimental results demonstrate that, compared to un-decomposed SVR, RNN, GRU, LSTM, CNN-LSTM, and TCN models as well as decomposed CEEMDAN-TCN and CEEMDAN-SF-TCN models, our method possesses superior forecasting capabilities.

This study primarily focuses on the New England region, validating the exceptional performance of the proposed electricity load forecasting model across the four seasons: spring, summer, autumn, and winter. By observing the diversity of consumption patterns in these seasons, the robustness and adaptability of the model are confirmed, and it is capable of handling different electricity usage characteristics. Nevertheless, this research has not yet thoroughly investigated the model's effectiveness in regions with consumption patterns significantly different from those of New England. It is necessary to explore further in this area, and plans are underway to extend our research to various geographical locations to validate the model's universality and applicability.

The model introduced in this paper has demonstrated an outstanding performance across the four quarters, with an R^2 value exceeding 0.999, indicating that the model can highly accurately fit the existing data. Theoretically, introducing more relevant variables, such as weather conditions or socio-economic indicators, could potentially improve the model's forecasting accuracy since these factors might have a significant impact on the electricity load. However, incorporating more variables would also increase the model's complexity, which could lead to more challenging data processing, a more cumbersome model structure, and possibly overfitting, in which the model becomes overly sensitive to the training data, reducing its generalization ability for unknown data. Therefore, while adding additional variables might help enhance its forecasting accuracy, this also brings a series of challenges that need further exploration and investigation in future research.

Typically, as the prediction horizon extends, its accuracy decreases due to insufficient data; thus, this study mainly focuses on short-term electricity load forecasting within a 24 h period. Hence, its predictive performance over longer time spans has not been further investigated, which requires continued exploration in future works. Additionally, considering the model's adaptable framework, it is not only applicable to electricity load forecasting but may also be suitable for other time series forecasting applications, although this necessitates further experimental validation.

Author Contributions: Conceptualization, X.X. and T.Y.; methodology, X.X. and T.Y.; software, T.Y.; validation, T.Y.; formal analysis, X.X. and T.Y.; investigation, T.Y.; resources, X.X.; data curation, T.Y.; writing—original draft preparation, T.Y.; writing—review and editing, X.X. and T.Y.; visualization, T.Y.; supervision, Y.Z.; project administration, Y.Z. and G.C.; funding acquisition, X.X. and T.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Zhejiang University of Science and Technology 2023 Postgraduate Research Innovation Fund Projects (2023yjskc05) and Zhejiang Provincial Natural Science Foundation (LY19F030004).

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: At this point, we extend our heartfelt appreciation to Cao Guangke for his invaluable contributions to our team. His role in launching the project and securing financial support has been fundamental to our progress. Throughout the duration of this project, Cao Guangke has been a beacon of guidance, providing us with essential data platform support and steering us towards our objectives. His commitment and leadership have been pivotal in driving our project forward. We wish to express our profound gratitude for his unwavering dedication and support.

Conflicts of Interest: Author Guangke Cao was employed by the company Hangzhou Shenhao Technology. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Chan, K.Y.; Yiu, K.F.C.; Kim, D.; Abu-Siada, A. Fuzzy Clustering-Based Deep Learning for Short-Term Load Forecasting in Power Grid Systems Using Time-Varying and Time-Invariant Features. *Sensors* **2024**, *24*, 1391. [CrossRef] [PubMed]
2. Yin, C.; Wei, N.; Wu, J.; Ruan, C.; Luo, X.; Zeng, F. An Empirical Mode Decomposition-Based Hybrid Model for Sub-Hourly Load Forecasting. *Energies* **2024**, *17*, 307. [CrossRef]
3. Kwon, B.S.; Park, R.J.; Song, K.B. Short-term load forecasting based on deep neural networks using LSTM layer. *J. Electr. Eng. Technol.* **2020**, *15*, 1501–1509. [CrossRef]
4. Lee, C.M.; Ko, C.N. Short-term load forecasting using lifting scheme and ARIMA models. *Expert Syst. Appl.* **2011**, *38*, 5902–5911. [CrossRef]
5. Taylor, J.W. Short-term load forecasting with exponentially weighted methods. *IEEE Trans. Power Syst.* **2011**, *27*, 458–464. [CrossRef]
6. Xia, Y.; Yu, S.; Jiang, L.; Wang, L.; Lv, H.; Shen, Q. Application of fuzzy support vector regression machine in power load prediction. *J. Intell. Fuzzy Syst.* **2023**, *45*, 8027–8048. [CrossRef]
7. Dong, X.; Deng, S.; Wang, D. A short-term power load forecasting method based on k-means and SVM. *J. Ambient Intell. Humaniz. Comput.* **2022**, *13*, 5253–5267. [CrossRef]
8. Abumohsen, M.; Owda, A.Y.; Owda, M. Electrical load forecasting using LSTM, GRU, and RNN algorithms. *Energies* **2023**, *16*, 2283. [CrossRef]
9. Zhang, J.; Huang, Y.; Pi, Y.; Sun, C.; Cai, W.; Huang, Y. Research on a Service Load Prediction Method Based on VMD-GLRT. *Appl. Sci.* **2023**, *13*, 3315. [CrossRef]
10. Liu, M.; Qin, H.; Cao, R.; Deng, S. Short-Term Load Forecasting Based on Improved TCN and DenseNet. *IEEE Access* **2022**, *10*, 115945–115957. [CrossRef]
11. Yao, H.; Tang, X.; Wei, H.; Zheng, G.; Yu, Y.; Li, Z. Modeling spatial-temporal dynamics for traffic prediction. *arXiv* **2018**, arXiv:1803.01254.
12. Guo, X.; Zhao, Q.; Zheng, D.; Ning, Y.; Gao, Y. A short-term load forecasting model of multi-scale CNN-LSTM hybrid neural network considering the real-time electricity price. *Energy Rep.* **2020**, *6*, 1046–1053. [CrossRef]
13. Geng, G.; He, Y.; Zhang, J.; Qin, T.; Yang, B. Short-Term Power Load Forecasting Based on PSO-Optimized VMD-TCN-Attention Mechanism. *Energies* **2023**, *16*, 4616. [CrossRef]
14. Smyl, S.; Dudek, G.; Pełka, P. Contextually enhanced ES-dRNN with dynamic attention for short-term load forecasting. *Neural Netw.* **2024**, *169*, 660–672. [CrossRef]
15. Yang, Q.; Lin, Y.; Kuang, S.; Wang, D. A novel short-term load forecasting approach for data-poor areas based on K-MIFS-XGBoost and transfer-learning. *Electr. Power Syst. Res.* **2024**, *229*, 110151. [CrossRef]

16. Nguyen, Q.D.; Nguyen, N.A.; Tran, N.T.; Solanki, V.K.; Crespo, R.G.; Nguyen, T.N.A. Online SARIMA applied for short-term electricity load forecasting. *Preprint* **2021**. [CrossRef]
17. Liu, M.; Li, Y.; Hu, J.; Wu, X.; Deng, S.; Li, H. A New Hybrid Model Based on SCINet and LSTM for Short-Term Power Load Forecasting. *Energies* **2024**, *17*, 95. [CrossRef]
18. Tarmanini, C.; Sarma, N.; Gezegin, C.; Ozgonenel, O. Short term load forecasting based on ARIMA and ANN approaches. *Energy Rep.* **2023**, *9*, 550–557. [CrossRef]
19. Xu, H.; Peng, Q.; Wang, Y.; Zhan, Z. Power-Load Forecasting Model Based on Informer and Its Application. *Energies* **2023**, *16*, 3086. [CrossRef]
20. Torres, M.E.; Colominas, M.A.; Schlotthauer, G.; Flandrin, P. A complete ensemble empirical mode decomposition with adaptive noise. In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 22–27 May 2011; IEEE: New York, NY, USA, 2011; pp. 4144–4147.
21. Yang, D. Short-term load monitoring of a power system based on neural network. *Int. Trans. Electr. Energy Syst.* **2023**, *2023*, 4581408. [CrossRef]
22. Huang, N.E.; Shen, Z.; Long, S.R.; Wu, M.C.; Shih, H.H.; Zheng, Q.; Yen, N.-C.; Tung, C.C.; Liu, H.H. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proc. R. Soc. Lond. A Math. Phys. Eng. Sci.* **1998**, *454*, 903–995. [CrossRef]
23. Wu, Z.; Huang, N.E. Ensemble empirical mode decomposition: A noise-assisted data analysis method. *Adv. Adapt. Data Anal.* **2009**, *1*, 1–41. [CrossRef]
24. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271.
25. Yang, T.; Hu, D.; Tang, C. Prediction of Dissolved Gas Content in Transformer Oil Based on SMA-VMD-GRU Model. *Trans. China Electrotech. Soc.* **2023**, *38*, 117–130.
26. Li, S.; Chen, H.; Wang, M.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323. [CrossRef]
27. Ho, T.K. Random decision forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; IEEE: New York, NY, USA, 1995; Volume 1, pp. 278–282.