energies

MDPI

Article

Short-Term Electrical Load Forecasting Based on IDBO-PTCN-GRU Model

Renxi Gong *, Zhihuan Wei, Yan Qin, Tao Liu and Jiawei Xu

School of Electrical Engineering, Guangxi University, Nanning 530004, China; 2212391058@st.gxu.edu.cn (Z.W.); 2212392098@st.gxu.edu.cn (Y.Q.); 2212391030@st.gxu.edu.cn (T.L.); 2212392123@st.gxu.edu.cn (J.X.)
* Correspondence: rxgong@gxu.edu.cn

Abstract: Accurate electrical load forecasting is crucial for the stable operation of power systems. However, existing forecasting models face limitations when handling multidimensional features and feature interactions. Additionally, traditional metaheuristic algorithms tend to become trapped in local optima during the optimization process, negatively impacting model performance and prediction accuracy. To address these challenges, this paper proposes a short-term electrical load forecasting method based on a parallel Temporal Convolutional Network–Gated Recurrent Unit (PTCN-GRU) model, optimized by an improved Dung Beetle Optimization algorithm (IDBO). This method employs a parallel TCN structure, using TCNs with different kernel sizes to extract and integrate multi-scale temporal features, thereby overcoming the limitations of traditional TCNs in processing multidimensional input data. Furthermore, this paper enhances the optimization performance and global search capability of the traditional Dung Beetle Optimization algorithm through several key improvements. Firstly, Latin hypercube sampling is introduced to increase the diversity of the initial population. Next, the Golden Sine Algorithm is integrated to refine the search behavior. Finally, a Cauchy–Gaussian mutation strategy is incorporated in the later stages of iteration to further strengthen the global search capability. Extensive experimental results demonstrate that the proposed IDBO-PTCN-GRU model significantly outperforms comparison models across all evaluation metrics. Specifically, the mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE) were reduced by 15.01%, 14.44%, and 14.42%, respectively, while the coefficient of determination (R^2) increased by 2.13%. This research provides a novel approach to enhancing the accuracy of electrical load forecasting.

Keywords: short-term electrical load forecasting; improved dung beetle optimization algorithm; parallel temporal convolutional network; gated recurrent unit

check for updates

Citation: Gong, R.; Wei, Z.; Qin, Y.; Liu, T.; Xu, J. Short-Term Electrical Load Forecasting Based on IDBO-PTCN-GRU Model. *Energies* **2024**, *17*, 4667. https://doi.org/10.3390/en17184667

Academic Editor: Andrea Mariscotti

Received: 29 July 2024
Revised: 12 September 2024
Accepted: 15 September 2024
Published: 19 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Electrical load forecasting plays a pivotal role in the management, scheduling, and operation of power systems [1]. Accurate load forecasting can optimize the dispatch of generating units, reduce the operational costs, and ensure the balance between power supply and demand, which is essential for maintaining system stability [2,3]. Moreover, improving load forecasting accuracy can significantly reduce the risk of system overload and economic losses. According to the literature [4] and [5], even a 1% reduction in forecast error can save the UK power industry about GBP 10 million in the operational costs. Therefore, enhancing the accuracy of load forecasting has become a crucial research topic for reducing costs in the power system and power market [6].

Electrical load forecasting can be categorized into three types based on the prediction time horizon: short-term (1 h to 1 week), medium-term (1 week to 1 year), and long-term (1 year to 20 years) [7,8]. Among these, short-term load forecasting is vital for power grid planning and smart grid construction. It has a profound impact on the stability, reliability, and economic operation of the system. Thus, it has become a focal point for

research and technological development in the power industry [9,10]. However, short-term load forecasting faces numerous challenges, such as handling high-dimensional data with significant volatility and accounting for various influencing factors, including weather, economic activities, and holidays [11,12]. The interactions among these factors are complex, thus making accurate short-term load forecasting an extremely challenging task.

To address these challenges, many scholars have conducted extensive research and proposed various effective prediction methods for short-term load forecasting. These methods can be broadly classified into three categories: classical statistical prediction methods, machine learning prediction methods, and modern hybrid prediction methods [13].

Classical statistical prediction methods include the Kalman Filter (KF) [14], Linear Regression (LR) [15], Exponential Smoothing (ES) [16], Grey Model (GM) [17], and Autoregressive Moving Average (ARMA) [18]. These methods predict electrical loads by building and analyzing mathematical relationships between data points, involving pattern identification, data characterization, and the application of time-series methods. While classical statistical models are simple, easy to understand, and offer rapid calculations, they fall short when dealing with nonlinear data and complex features, especially the high-frequency, non-smooth characteristics of power loads. As the complexity of power load forecasting requirements increases, the limitations of these traditional methods become more apparent, underscoring the need for more effective methods to improve forecast accuracy.

With the rapid development of artificial intelligence technology, machine learning has effectively overcome the limitations of classical statistical models. It excels in nonlinear learning and has a lower dependence on complex mathematical modeling. Machine learning methods can be divided into two categories: traditional machine learning and deep learning. Traditional methods include Support Vector Machines (SVMs) [19], Random Forests (RFs) [20], Decision Trees (DTs) [21], Extreme Learning Machines (ELMs) [22], and Feed-Forward Neural Networks (FFNNs) [23]. These methods, based on probability theory and statistics, utilize powerful computing capabilities to extract features from historical data and build models that achieve accurate predictions of future loads. Traditional machine learning models have a high processing speed and a strong ability to capture nonlinear relationships in power load forecasting. However, they have limitations when it comes to handling changes in data distribution and time-series characteristics.

Recurrent Neural Networks (RNNs) [24], Long Short-Term Memory networks (LSTMs) [25], and Gated Recurrent Units (GRUs) [26] are deep learning models widely applied in modern power system load forecasting. These models are favored for their unique memory capabilities. Through recursive and gating mechanisms, they effectively capture temporal dependencies and accommodate the time-series and nonlinear characteristics of load data. However, despite their strong performance in load forecasting, each standalone method has inherent limitations. For instance, complex deep learning architectures like deep RNNs or LSTMs may suffer from overfitting. This overfitting reduces their generalization ability. Additionally, these standalone models often exhibit poor robustness to outliers or noisy data. As a result, they become susceptible to external disturbances, leading to unstable prediction outcomes.

In recent years, with the diversification of forecasting models, new architectures based on hybrid forecasting methods have gradually emerged. These methods combine the strengths of different standalone models, demonstrating excellent performance in improving prediction accuracy and robustness. For example, the CNN-LSTM hybrid model is first based on Convolutional Neural Networks (CNNs) to extract the local features from the input data, and then, on Long Short-Term Memory networks (LSTMs) to capture the temporal dependencies. The model has been successfully applied to short-term load forecasting for individual households and regional power systems [27,28]. Experimental results show that this hybrid model achieves a significantly higher prediction accuracy compared to standalone LSTM models. Similarly, the GRU-CNN hybrid model extracts time-series features using Gated Recurrent Units (GRUs) and processes high-dimensional data through CNNs. Research indicates that this model outperforms traditional Backpropa-

gation Neural Networks (BPNNs) as well as standalone GRU or CNN models in terms of error metrics [29].

While CNNs have significant advantages in extracting local features, such as daily or hourly load patterns, their limited receptive field makes it difficult to effectively capture long-term seasonal trends and time-series characteristics across time steps. To address this issue, Temporal Convolutional Networks (TCNs) utilize causal dilation convolutions and residual connections [30]. These techniques markedly enhance TCNs' ability to capture long-term temporal features in power load data. The TCN-GRU hybrid model described in the literature [31] leverages a TCN's capability to extract temporal features, combined with a GRU's short-term load forecasting functionality. Research indicates that a TCN outperforms the traditional CNNs in handling one-dimensional features. Moreover, the combination of a TCN and GRU shows significant advantages in capturing complex temporal relationships, thereby further improving the prediction accuracy [32–34].

However, the application of the conventional TCN-GRU model in power load forecasting still has certain limitations. One limitation is that the stacked structure of TCN layers—where the output of one layer serves as the input to the next—performs poorly when handling multidimensional input data, such as historical load, meteorological, economic, and date features. Integrating the interactive relationships among these different features is challenging. For example, the impact of weather changes and economic fluctuations on load is both nonlinear and interrelated. The TCN-GRU model, when dealing with these complex features, shows limited performance, which may potentially affect prediction accuracy. Another challenge is that a TCN is primarily designed for one-dimensional time-series data. This design makes it difficult to distinguish and process temporal-dependent features from non-temporal-dependent features, such as holidays or sudden events. This limitation could further weaken the accuracy of the predictions.

When designing effective hybrid forecasting models, it is crucial to focus not only on the model's architecture and its ability to extract temporal features but also on the optimization of model parameters. Although the aforementioned hybrid models have made significant progress in structural design, optimizing their parameters is equally critical to fully realizing their potential. To address this issue, researchers have combined metaheuristic algorithms with hybrid forecasting models. This combination allows for the automated and efficient optimization of model parameters, thereby significantly enhancing model performance. For example, the literature [35] employs the Particle Swarm Optimization (PSO) algorithm to optimize the hyperparameters of the CNN-BiGRU model. This approach significantly improves the accuracy of remaining life prediction under complex operating conditions. Similarly, the literature [36] integrates the Grey Wolf Optimization (GWO) algorithm to optimize the CNN-BiLSTM model, which has been successfully applied to building energy consumption forecasting. This integration effectively enhances the model's generalization ability and accuracy. Furthermore, the literature [37] developed a hybrid Genetic Whale Optimization Algorithm (GCWOA) to optimize the hyperparameters of the CNN-GRU-AM model, leading to improvements in both the accuracy and robustness of ship motion prediction. Despite these effective optimization strategies, metaheuristic algorithms still have limitations, such as a tendency to fall into local optima or a failure to find the global optimum, which can adversely affect the model's overall performance.

Based on the analysis and discussion above, the existing TCN-GRU model exhibits limitations in handling multidimensional features, especially when it comes to integrating the interactive relationships among different features. Additionally, traditional metaheuristic algorithms are prone to falling into local optima during the optimization process, which can negatively impact model performance. To address these challenges, this paper proposes an IDBO-PTCN-GRU short-term power load forecasting method. By introducing a parallel TCN structure, this method utilizes TCNs with different kernel sizes to comprehensively extract temporal features from multiple scales. These features are then effectively fused, overcoming the limitations of conventional TCNs in handling multidimensional input data (such as historical load, meteorological conditions, economic indicators, and

date features). This approach particularly addresses the challenge of fully capturing and integrating complex interactions among different features. To enhance the optimization process, this paper incorporates Latin hypercube sampling to increase the diversity of the initial population. It also combines the Golden Sine Algorithm to optimize the rolling behavior and introduces a Cauchy–Gaussian mutation strategy in the later stages of iteration. These improvements effectively enhance the optimization performance and global search capability of the traditional Dung Beetle Optimization (DBO) algorithm.

To better understand the method proposed in this paper and its advantages, the following sections provide a detailed overview of the content's organization. The remainder of this paper is organized as follows. Section 2 introduces the PTCN-GRU model proposed in this study. Section 3 elaborates on the improved Dung Beetle Optimization (IDBO) algorithm and tests of its optimization capability. Section 4 presents the structure and principles of the PTCN-GRU model, which is optimized by the IDBO algorithm. Section 5 outlines the experimental setup, results, and comparative analysis. Finally, Section 6 summarizes the research findings and discusses future research directions.

2. PTCN-GRU Model

The structure of the PTCN-GRU hybrid network model proposed in this paper is illustrated in Figure 1. It primarily consists of five key modules: the input layer, PTCN layer, GRU layer, fully connected (Dense) layer, and output layer. This architectural design aims to fully leverage the advantages of each layer, achieving efficient feature extraction and nonlinear relationship modeling.

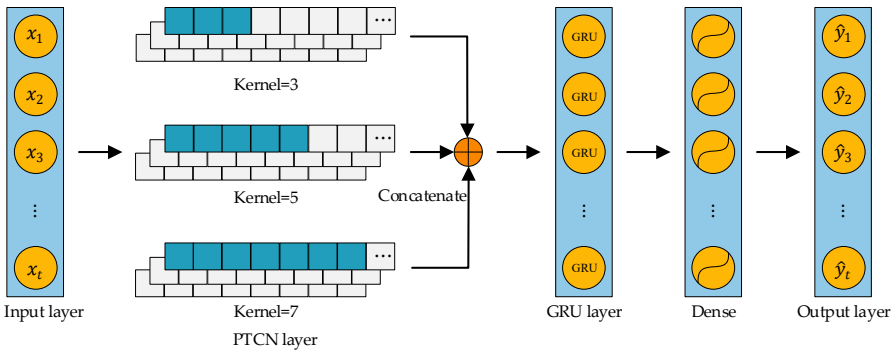


Figure 1. Model structure of PTCN-GRU.

2.1. Multi-Scale Temporal Feature Extraction Based on PTCN

The input layer of the model integrates multidimensional features, including historical load data, holiday information, and meteorological conditions. Unlike the classic layered stack hybrid model, where the output of one layer serves as the input to the next (commonly referred to as the traditional TCN-GRU model), the PTCN layer introduces a parallel TCN structure. This structure employs convolution kernels of different sizes, such as 3, 5, and 7. These varying kernels comprehensively extract temporal features across multiple scales. Feature fusion and output are achieved through parallel concatenation.

Specifically, the parallel structure of the PTCN applies differentiated temporal convolution kernels based on the specific characteristics of each feature. This approach effectively avoids the common issues in the traditional TCN-GRU model, such as feature confusion and the inadequate capture of complex inter-feature relationships when processing multidimensional data. Additionally, the PTCN utilizes a multi-scale convolution strategy. By applying convolution kernels of varying sizes, it extracts features across multiple temporal scales. This allows the model to simultaneously capture short-term fluctuations, medium-term variations, and long-term trends. As a result, it overcomes the limitations of single-scale feature extraction seen in traditional TCN models. Lastly, the feature fusion mechanism in the PTCN effectively integrates features from different temporal scales. It also captures complex feature interactions, enhancing the model's ability to represent mul-

tidimensional features. This mechanism ensures the full utilization of information across different temporal scales, addressing the shortcomings of the traditional TCN-GRU model in feature integration and representation.

To better understand the operations of the PTCN layer, the following mathematical expressions can be used to represent it:

$$S_1 = C_3(x) \tag{1}$$

$$S_2 = C_5(x) \tag{2}$$

$$S_3 = C_7(x) \tag{3}$$

$$S = \text{Concat}(S_1, S_2, S_3) \tag{4}$$

Here, C_k represents the convolution operation with a kernel size of k, where k takes the values 3, 5, and 7, respectively. The variable x denotes the input data to the model, and S represents the output of the PTCN layer. The function $\text{Concat}(\cdot)$ is used to concatenate the convolution results across different scales.

2.2. Capturing Complex Nonlinear Changes Based on GRU

The fused features are transmitted to the GRU layer for processing. Through its unique update gate and reset gate mechanisms, the GRU can simultaneously consider multiple influencing factors and their nonlinear interactions. By dynamically adjusting the state of memory units, the GRU adaptively reflects these complex relationships. This mechanism enables the GRU to effectively learn and simulate complex nonlinear patterns in load variations, providing a robust foundation for accurate prediction.

The computational process of the GRU is as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{5}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{6}$$

$$\widetilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t]) \tag{7}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \tag{8}$$

where $\sigma(\cdot)$ represents the sigmoid activation function; W_z, W_r, and W denote the weight matrices; $\odot$ indicates element-wise multiplication; z_t and r_t are the outputs of the update gate and reset gate, respectively; x_t is the input at the current time step; and h_{t-1} is the hidden state from the previous time step. The candidate hidden state $\widetilde{h}_t$ is computed based on the current input and the adjusted hidden state from the previous time step, reflecting the information at the current time step. The updated hidden state h_t is derived by combining the candidate hidden state with the update gate z_t, thereby balancing the influence of current information and past information [38].

2.3. Fully Connected Layer and Final Output

The output from the GRU layer is processed through a fully connected layer. This layer employs the sigmoid function as its activation function, compressing multidimensional features into a one-dimensional output to generate the final predicted value $\hat{y}_t$. This process can be represented by the following equation:

$$\hat{y}_t = \sigma(W_a \cdot A + b_a) \tag{9}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function; A represents the output from the GRU layer; W_a is the weight matrix; and b_a represents the bias vector.

3. Improved Dung Beetle Optimizer and Its Optimization Capability Test

3.1. Improved Beetle Optimization Algorithm

The Dung Beetle Optimizer (DBO) is a novel swarm intelligence algorithm proposed by Xue et al. in 2022 to address complex optimization problems [39]. This algorithm draws inspiration from the unique survival strategies of dung beetles. It achieves optimization objectives by simulating their five behaviors: ball rolling, dancing, reproduction, foraging, and stealing. Compared to traditional intelligent optimization algorithms such as the Grey Wolf Optimizer (GWO) and the Whale Optimization Algorithm (WOA), the DBO achieves a better balance between local exploitation and global exploration. However, in practical applications, the DBO still faces several challenges. These include low population diversity, insufficient global search capability, and a tendency to become trapped in local optima. To address these limitations, this paper proposes improvements to the traditional DBO in the following three aspects.

3.1.1. Population Initialization Based on Latin Hypercube Sampling

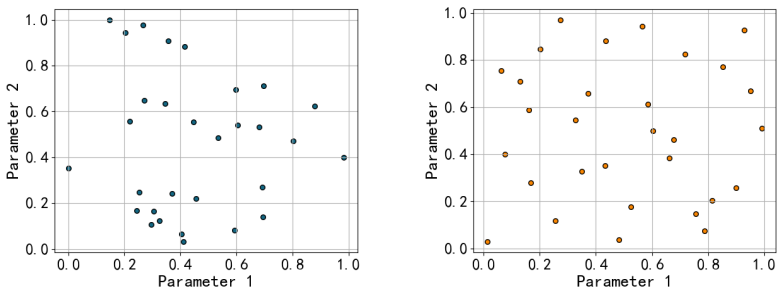
To address the issue of reduced diversity caused by unevenly distributed initial populations in the original DBO algorithm, this paper proposes using Latin hypercube sampling (LHS) for population initialization. LHS ensures comprehensive coverage of the variable range through stratified sampling. It guarantees that each region contains sampling points, enhancing population randomness and uniformity. This approach significantly improves population diversity.

The detailed operation principle of this method is as follows:

1. Assume the sampling space is d-dimensional with a sampling size of N.
2. Divide the definition interval $[x_l, x_u]$ of each dimensional variable x_i into N equal parts, resulting in $N \times d$ intervals.
3. Randomly generate a number within each divided interval, forming an $N \times d$ sampling matrix S.
4. In matrix S, randomly sort each column, then sequentially select one number from each column to form a sampling vector.
5. Repeat step 4 until N sampling vectors are generated, constituting the final population.

Through this approach, LHS ensures a uniform distribution of sampling points within each variable range. At the same time, it increases the randomness of population initialization. This not only significantly enhances population diversity but also allows the algorithm to explore the search space more comprehensively. As a result, the probability of finding the global optimum is increased.

Assuming a population size of 30 and a two-dimensional search space with each dimension ranging from [0, 1], Figure 2 illustrates two scenarios. These are (a) random population initialization and (b) Latin hypercube sampling initialization. As shown, Latin hypercube sampling avoids the concentration or sparsity of samples observed in random initialization. This leads to a more representative and diverse initial population, thereby improving the algorithm's search efficiency and optimization performance.



(**a**) (**b**)

Figure 2. (**a**) Random population initialization; (**b**) LHS population initialization.

3.1.2. Golden Sine Strategy

The original DBO algorithm has a limitation due to the linear rolling behavior of dung beetles. This behavior restricts both the local and global search capabilities of the algorithm. To address this issue, this paper proposes an improvement using the Golden Sine Algorithm (Golden-SA). The Golden-SA leverages the sine function to traverse the unit circle. It then combines this with the golden section method to narrow the solution space. This optimization improves the balance between exploration and exploitation. Consequently, it enhances the performance of rolling dung beetles in both global exploration and local search. The improved position update formula for the rolling dung beetle is as follows:

$$x_i(t+1) = x_i(t) \times |\sin(r_1)| - r_2 \times \sin(r_1) \times \left| c_1 \times X^b - c_2 \times x_i(\mathrm{t}) \right| \tag{10}$$

where X^b represents the global best position; r_1 is a random number within the interval $[0, 2\pi]$; r_2 is a random number within the interval $[0, \pi]$; c_1 and c_2 are the golden section coefficients, calculated as follows:

$$c_1 = a \times (1 - g) + b \times g \tag{11}$$

$$c_2 = a \times g + b \times (1 - g) \tag{12}$$

where $g = 0.618$ is the golden ratio, and a and b are the initial values for the golden section ratio, typically set to $a = -\pi$ and $b = \pi$. This improvement effectively enhances the algorithm's global search capability and local search precision, significantly increasing the probability of finding the global optimum.

3.1.3. Introduction of Cauchy–Gaussian Mutation Strategy

The original DBO algorithm tends to fall into local optima in the later stages of iteration. To solve this issue, this paper proposes an improvement using the Cauchy–Gaussian mutation strategy. This strategy first selects the individual with the best current fitness for mutation. Then, it compares the positions before and after mutation. The superior position is chosen for the next iteration. By doing so, this approach enhances population diversity and improves global search capability in the later stages of iteration. The specific formula is as follows:

$$X_{\text{new}} = X^b \times [1 + \lambda_1 \text{Cauchy}(0,1) + \lambda_2 \text{Gaussian}(0,1)] \tag{13}$$

where X_{new} is the position after mutation; X^b is the current global best position; $\text{Cauchy}(0,1)$ is a random number following the Cauchy distribution; and $\text{Gaussian}(0,1)$ is a random number following the Gaussian distribution. λ_1 and λ_2 are dynamic parameters defined as

$$\lambda_1 = 1 - t^2 / T_{\max}^2 \tag{14}$$

$$\lambda_2 = t^2 / T_{\max}^2 \tag{15}$$

where t is the current iteration number and $T_{\max}$ is the maximum number of iterations. As the iteration progresses, λ_1 gradually decreases while λ_2 increases. This mechanism enables the algorithm to escape local optima in the later stages, balancing local exploitation and global exploration capabilities.

3.2. Function Testing

3.2.1. Experimental Environment

The simulation experiments in this study were conducted on a 64-bit Windows 10 operating system, with hardware configurations including an Intel® Core™ i7-10700 CPU (manufactured by Intel Corporation, Santa Clara, CA, USA), an NVIDIA RTX 2070 graphics card (manufactured by NVIDIA Corporation, Santa Clara, CA, USA), and 16 GB of

RAM. The programming environment used was PyCharm 2022, with Python 3.8 as the programming language.

3.2.2. Test Functions

To validate the performance of the improved Dung Beetle Optimizer (IDBO), 23 classic test functions were selected. Functions F1 to F7 are unimodal test functions, used to evaluate the convergence speed and precision of the algorithm. Functions F8 to F13 are high-dimensional multimodal test functions, while F14 to F23 are fixed-dimension multimodal test functions, both designed to examine the algorithm's global search capability and its ability to escape local optima.

3.2.3. Comparative Algorithms' Parameters and Experimental Setup

To comprehensively evaluate the performance of the IDBO algorithm, this study selected four widely applied optimization algorithms for comparison: Particle Swarm Optimization (PSO) [40], Whale Optimization Algorithm (WOA) [41], Grey Wolf Optimizer (GWO) [42], and Butterfly Optimization Algorithm (BOA) [43]. These algorithms encompass both advanced techniques and classical methods. All comparative algorithms' parameters were set according to their original literature to ensure experimental fairness and accuracy. Table 1 lists the specific parameter settings for each algorithm.

Table 1. Parameter values of the algorithm.

Algorithm	Parameters	Parameter Settings
PSO	Inertia weight w	0.9
	Individual learning factor c_1	2
	Social learning factor c_2	2
WOA	Convergence factor a	Linearly decreases from 2 to 0
GWO	Convergence factor a	Linearly decreases from 2 to 0
BOA	Power exponent a	0.1
	Sensory modality c	0.1
DBO	Obstacle encounter probability λ	0.1
	Deflection coefficient k	0.1
	Constants b and S	0.3, 0.5
IDBO	Obstacle encounter probability λ	0.1
	Constant S	0.5

In the experimental setup, all algorithms were executed for 500 iterations with a population size of 30. Each algorithm was run independently 30 times. Mean values and standard deviations were used to assess algorithm performance. A smaller mean value indicates higher convergence precision, while a smaller standard deviation suggests stronger optimization stability. To mitigate the impact of randomness in single runs, the average fitness variation curve of 30 independent runs was taken to reflect the convergence speed and overall optimization performance of each algorithm. This method allows for a more objective comparison of the algorithms' performance in terms of convergence and stability.

3.2.4. Experimental Analysis

The experimental results are presented in Table 2. For unimodal test functions, the IDBO algorithm demonstrates average fitness values closer to the theoretical optimum in F1 to F6. However, it shows only slightly inferior performance in F7. This indicates the algorithm's superior local exploitation capability when dealing with such problems.

Table 2. Function test results of IDBO, DBO, BOA, GWO, WOA, and PSO (bold indicates the optimal result).

Functions	Metrics	PSO	WOA	GWO	BOA	DBO	IDBO
F1	Mean	27.7	2.70×10^{-20}	1.61×10^{-27}	5.94×10^{-10}	4.53×10^{-43}	$\mathbf{1.02 \times 10^{-256}}$
	Std	8.62	4.95×10^{-20}	2.41×10^{-27}	3.32×10^{-11}	2.44×10^{-42}	0.00
F2	Mean	12.0	4.32×10^{-14}	1.34×10^{-16}	1.33×10^{-6}	2.04×10^{-22}	$\mathbf{1.03 \times 10^{-140}}$
	Std	7.45	6.83×10^{-14}	9.85×10^{-17}	3.95×10^{-7}	8.25×10^{-22}	5.55×10^{-140}
F3	Mean	986	5.49×10^{-3}	2.18×10^{-5}	5.65×10^{-10}	7.35×10^{-10}	$\mathbf{3.00 \times 10^{-222}}$
	Std	420	8.76×10^{-3}	6.01×10^{-5}	4.03×10^{-11}	3.96×10^{-9}	0.00
F4	Mean	0.419	1.91×10^{-5}	5.65×10^{-8}	1.60×10^{-7}	5.21×10^{-11}	$\mathbf{3.04 \times 10^{-172}}$
	Std	0.338	2.41×10^{-5}	4.30×10^{-8}	8.62×10^{-9}	2.38×10^{-10}	0.00
F5	Mean	1.52×10^{4}	27.4	27.2	28.9	28.1	**24.6**
	Std	1.03×10^{4}	0.76	0.743	3.10×10^{-2}	0.68	7.43
F6	Mean	26.7	1.24	0.77	5.89	2.34	$\mathbf{1.16 \times 10^{-3}}$
	Std	6.26	0.451	0.432	0.65	1.54	1.48×10^{-3}
F7	Mean	0.751	8.13×10^{-4}	1.85×10^{-3}	$\mathbf{9.61 \times 10^{-5}}$	6.35×10^{-4}	9.87×10^{-4}
	Std	0.55	1.06×10^{-3}	8.88×10^{-4}	8.84×10^{-5}	5.21×10^{-4}	1.11×10^{-3}
F8	Mean	-5.27×10^{3}	-7.13×10^{3}	-5.86×10^{3}	-2.30×10^{3}	$\mathbf{-1.26 \times 10^{4}}$	-1.22×10^{4}
	Std	1.52×10^{3}	3.38×10^{2}	9.12×10^{2}	4.68×10^{2}	0.963	613
F9	Mean	59.4	4.55×10^{-14}	2.94	147	**0.00**	**0.00**
	Std	32.7	1.06×10^{-13}	3.05	66.5	0.00	0.00
F10	Mean	5.23	5.10×10^{-11}	5.72×10^{-14}	1.91×10^{-7}	$\mathbf{4.44 \times 10^{-16}}$	$\mathbf{4.44 \times 10^{-16}}$
	Std	0.532	8.00×10^{-11}	8.96×10^{-15}	9.94×10^{-9}	0.00	0.00
F11	Mean	0.832	4.68×10^{-12}	3.63×10^{-3}	1.18×10^{-9}	**0.00**	**0.00**
	Std	9.65×10^{-2}	2.52×10^{-11}	7.82×10^{-3}	1.85×10^{-10}	0.00	0.00
F12	Mean	8.01	0.117	4.64×10^{-2}	0.755	0.174	$\mathbf{1.47 \times 10^{-4}}$
	Std	4.60	0.132	2.83×10^{-2}	0.229	0.172	2.88×10^{-4}
F13	Mean	11.8	1.48	0.662	2.91	2.52	$\mathbf{1.54 \times 10^{-3}}$
	Std	11.2	0.313	0.225	0.165	0.351	4.24×10^{-3}
F14	Mean	2.88	1.50	4.10	**1.41**	2.67	1.91
	Std	1.87	0.717	4.09	0.694	0.925	0.886
F15	Mean	5.29×10^{-3}	$\mathbf{3.44 \times 10^{-4}}$	5.21×10^{-3}	3.96×10^{-4}	6.44×10^{-4}	4.48×10^{-4}
	Std	7.55×10^{-3}	5.01×10^{-5}	8.37×10^{-3}	7.20×10^{-5}	4.03×10^{-4}	3.06×10^{-4}
F16	Mean	**−1.03**	**−1.03**	**−1.03**	−0.572	**−1.03**	**−1.03**
	Std	1.80×10^{-3}	1.19×10^{-7}	1.23×10^{-8}	0.199	8.30×10^{-4}	9.58×10^{-4}
F17	Mean	**0.398**	**0.398**	**0.398**	0.541	**0.398**	**0.398**
	Std	3.97×10^{-4}	3.86×10^{-6}	2.78×10^{-7}	0.286	3.11×10^{-5}	3.11×10^{-4}
F18	Mean	3.02	**3.00**	**3.00**	3.15	**3.00**	**3.00**
	Std	2.17×10^{-2}	1.29×10^{-5}	3.29×10^{-5}	0.233	1.18×10^{-6}	1.10×10^{-14}
F19	Mean	**−0.3**	**−0.3**	**−0.3**	-1.32×10^{-2}	**−0.3**	**−0.3**
	Std	1.11×10^{-16}	1.11×10^{-16}	1.11×10^{-16}	2.56×10^{-2}	1.11×10^{-16}	1.11×10^{-16}
F20	Mean	−3.16	−2.74	**−3.25**	−1.50	−3.01	−3.06
	Std	0.151	0.313	9.64×10^{-2}	0.49	0.424	0.298
F21	Mean	**−9.71**	−2.25	−9.40	−0.485	−5.37	−7.27
	Std	1.65	1.52	1.97	0.246	3.45	3.40

Table 2. *Cont.*

Functions	Metrics	PSO	WOA	GWO	BOA	DBO	IDBO
F22	Mean	**−10.2**	−2.78	−10	−0.686	−5.38	−8.93
	Std	1.04	1.54	1.33	0.301	2.92	2.99
F23	Mean	**−10.5**	−2.78	−9.99	−0.826	−5.94	−7.52
	Std	2.47×10^{-6}	1.53	2.02	0.302	2.52	3.53
+/−/=		16/4/3	16/3/4	15/4/4	20/3/0	14/2/7	~
Mean		4.3478	3.1739	2.7826	4.7391	2.5652	**1.6957**
Rank		5	4	3	6	2	1

In the high-dimensional multimodal function tests, the IDBO algorithm exhibits significant superiority in F9 to F13. Notably, for F9 to F11, the IDBO algorithm successfully locates the global optimal solution. This highlights its excellent global search performance. Although the IDBO algorithm does not achieve optimal performance in handling the high-dimensional multimodal function F8, its results are only marginally inferior to those of the DBO algorithm.

For fixed-dimension multimodal function tests, the IDBO algorithm also demonstrates the ability to find the global optimal solution in F16 to F19. Despite not achieving optimal results in the tests for F14, F15, and F20 to F23, the IDBO algorithm consistently maintains top-tier performance. This fully reflects its optimization stability and reliability.

These results collectively underscore the IDBO algorithm's robust performance across various function types. They showcase its balanced capability in both exploitation and exploration. The algorithm's consistent performance across the different problem landscapes demonstrates its versatility and effectiveness as an optimization tool.

Figure 3 illustrates the average fitness variation curves for selected test functions. It can be observed that the IDBO algorithm demonstrates the fastest convergence rate and highest precision when handling unimodal test functions F1 to F3. Similarly, it shows outstanding performance with high-dimensional multimodal test functions F9 to F11. This indicates that the IDBO combines both efficiency and accuracy in solving these problems. For fixed-dimension multimodal functions F16 to F18, the IDBO's initial convergence speed is not the fastest. However, it effectively avoids being trapped in local optima and ultimately converges to the global optimal solution. This demonstrates that the IDBO excels not only in convergence speed but also in maintaining efficient global search capabilities in complex optimization environments. These results reflect its excellent overall optimization performance.

In summary, the improved Dung Beetle Optimizer (IDBO) exhibits exceptional adaptability and stability across various types of test functions. Its outstanding performance in optimization efficiency and broad applicability suggests that the IDBO has high potential for use in a wide range of optimization scenarios and practical applications.

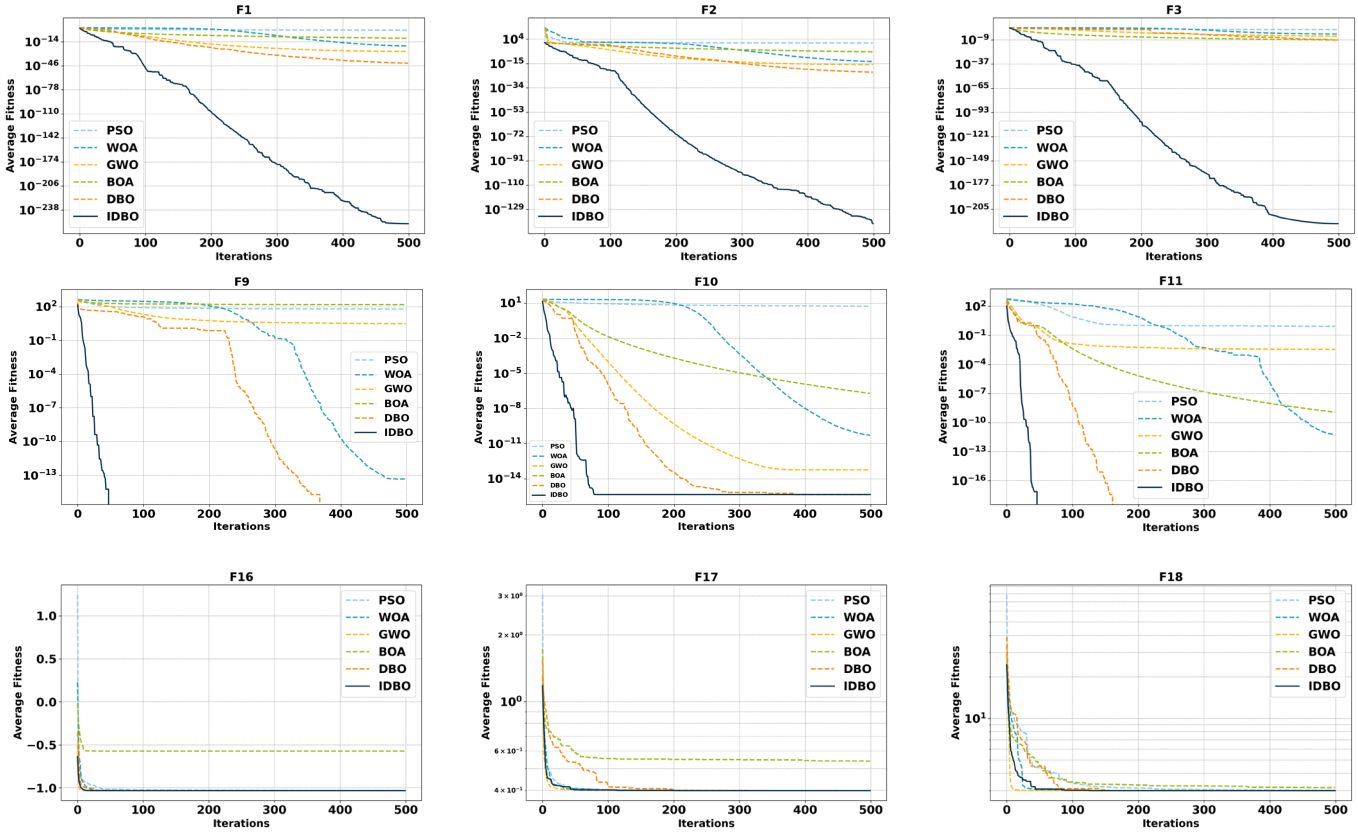


Figure 3. Convergence curves of IDBO algorithm and other optimization algorithms for test functions.

3.2.5. Statistical Analysis of IDBO

To further validate the significant differences between the IDBO and other algorithms, we conducted Wilcoxon rank-sum tests and calculated *p*-values. A *p*-value less than 0.05 indicates a significant difference between the IDBO and the compared algorithm. If the *p*-value is greater than 0.05, it suggests no significant difference. When the *p*-value is NaN, it means the results of the two algorithms are highly similar, making it impossible to determine significance. As shown in Table 3, for F9–F11, F17, and F19, the results of the IDBO and DBO are highly similar, making it difficult to determine significance. Additionally, for F19, the IDBO shows high similarity with four other algorithms, except the BOA. However, for most test functions, the *p*-values are less than 0.05. This indicates significant differences in overall performance between the IDBO and other comparative algorithms.

To more intuitively compare the optimization performance of the IDBO with other algorithms, we calculated the optimization precision of the IDBO relative to other algorithms for each test function. We used the symbols "+/−/=" to represent "better than", "worse than", and "equal to", respectively. Furthermore, we conducted a comprehensive assessment of the results for each test function. We ranked each optimization algorithm based on mean values, calculated the average ranking (Mean), and finally determined the overall algorithm ranking (Rank). As shown in Table 2, the IDBO algorithm achieves an average ranking value (Mean) of 1.6957, which is significantly lower than other optimization methods. This highlights its superior optimization performance and demonstrates strong stability and robustness.

Table 3. Wilcoxon rank-sum test *p*-values of IDBO algorithm and other optimization algorithms (bold indicates results with no significant difference).

Functions	PSO	WOA	GWO	BOA	DBO
F1	3.16×10^{-12}	3.16×10^{-12}	3.16×10^{-12}	3.16×10^{-12}	3.16×10^{-12}
F2	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F3	5.22×10^{-12}	5.22×10^{-12}	5.22×10^{-12}	5.22×10^{-12}	5.22×10^{-12}
F4	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F5	3.02×10^{-11}	2.92×10^{-2}	**0.122**	3.02×10^{-11}	1.47×10^{-7}
F6	3.02×10^{-11}	3.02×10^{-11}	4.20×10^{-10}	3.02×10^{-11}	3.02×10^{-11}
F7	3.02×10^{-11}	**0.297**	6.77×10^{-5}	3.08×10^{-8}	**0.246**
F8	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	2.22×10^{-2}
F9	1.21×10^{-11}	3.06×10^{-4}	1.20×10^{-12}	1.21×10^{-12}	NaN
F10	1.21×10^{-12}	1.21×10^{-12}	1.03×10^{-12}	1.21×10^{-12}	NaN
F11	1.21×10^{-12}	$\mathbf{8.15 \times 10^{-2}}$	1.10×10^{-2}	1.21×10^{-12}	NaN
F12	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F13	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}	3.02×10^{-11}
F14	6.49×10^{-4}	**0.97**	1.94×10^{-3}	**0.673**	5.82×10^{-3}
F15	3.47×10^{-10}	**0.228**	2.39×10^{-4}	6.55×10^{-4}	9.52×10^{-4}
F16	2.11×10^{-10}	2.56×10^{-10}	2.56×10^{-10}	1.24×10^{-11}	**0.248**
F17	1.44×10^{-10}	1.44×10^{-10}	1.44×10^{-10}	1.22×10^{-11}	NaN
F18	3.01×10^{-11}	3.01×10^{-11}	3.01×10^{-11}	3.01×10^{-11}	**0.717**
F19	NaN	NaN	NaN	1.21×10^{-12}	NaN
F20	**0.935**	3.59×10^{-5}	**0.234**	7.39×10^{-11}	**0.796**
F21	**0.494**	6.51×10^{-7}	**0.52**	3.68×10^{-11}	**0.176**
F22	2.99×10^{-4}	9.60×10^{-8}	3.52×10^{-4}	4.34×10^{-11}	5.64×10^{-4}
F23	**0.658**	3.09×10^{-6}	**0.264**	4.48×10^{-11}	**0.594**

4. IDBO-Optimized PTCN-GRU Model

Through the aforementioned research, we conducted a comprehensive performance evaluation of various algorithms. The results demonstrate that the proposed IDBO algorithm exhibits superior performance across multiple key indicators. The critical hyperparameters of the PTCN-GRU model, such as the number of convolutional filters, the number of GRU hidden layer units, and the learning rate, significantly influence the model's network structure and prediction accuracy.

However, in practical applications, selecting these parameters often relies on experience, which introduces considerable uncertainty and may significantly affect the model's predictive performance. To address this issue and enhance the model's prediction accuracy, we employed the IDBO algorithm to systematically optimize the key parameters of the PTCN-GRU model. As a result, we constructed an IDBO-optimized PTCN-GRU model for electrical load forecasting.

The entire process of IDBO optimization is illustrated in Figure 4. The specific steps for hyperparameter optimization are as follows:

1. Determine the structure of the PTCN-GRU network and specify the hyperparameters to be optimized;
2. Based on the network's hyperparameter settings, define the IDBO algorithm's population size, search space dimension, search range, and maximum number of iterations, then initialize the population;

3. Input the initialized hyperparameters into the PTCN-GRU network for training;
4. Use the trained network to make predictions on the validation set, and employ the mean squared error between the predicted results and actual values on the validation set as the fitness function to evaluate model performance;
5. Rank the obtained fitness values and adjust individual positions according to the corresponding update formulas;
6. Determine whether the preset maximum number of iterations has been reached. If so, terminate the optimization process and output the optimal network parameters; otherwise, transmit the position information of the new generation of individuals back to the PTCN-GRU network for continued training until the stopping criterion is met.

In conclusion, the method combining the PTCN-GRU network with the IDBO algorithm, through automated hyperparameter optimization and an efficient search mechanism, not only significantly enhances the model's predictive performance and training efficiency but also ensures that the model possesses strong stability and good generalization capabilities.

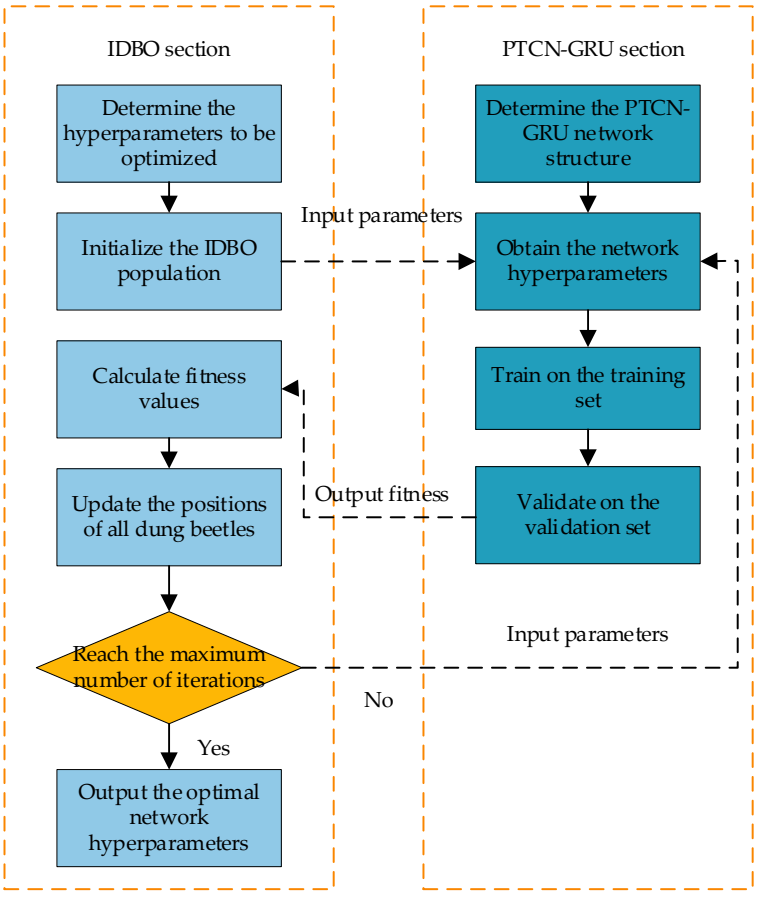


Figure 4. Optimized PTCN-GRU model structure based on IDBO.

5. Experiments

To validate the feasibility and superiority of the proposed model, this study selected electrical load and weather data from a specific region in Australia. The data were collected between January 2006 and December 2010 for short-term electrical load forecasting research. The sampling interval was 30 min, resulting in 48 data points per day. The first 80% of the dataset was used for model training, while the remaining 20% was used for model validation. Data from 20 to 26 December 2010 were chosen as the test sample.

This study employed a direct multi-output prediction method. It used the historical load data from the previous day, combined with weather and date features of the forecast day, as inputs to directly predict the electrical load for that day.

The hardware configuration for this experiment was identical to that described in Section 3. The deep learning architecture was implemented based on the TensorFlow framework, using TensorFlow GPU version 2.10.0.

5.1. Feature Engineering

5.1.1. Pearson Correlation Analysis

The initial feature set comprised 13 dimensions of data. These included meteorological features (such as dry-bulb temperature and dew point temperature), economic features (e.g., electricity price), and date-related features (such as year and season). To mitigate the impact of redundant features on prediction accuracy, we used Pearson correlation coefficients to calculate the correlation between each feature and the electrical load. This analysis formed the basis for selecting the model's input variables. The results of the correlation analysis are presented in Figure 5.

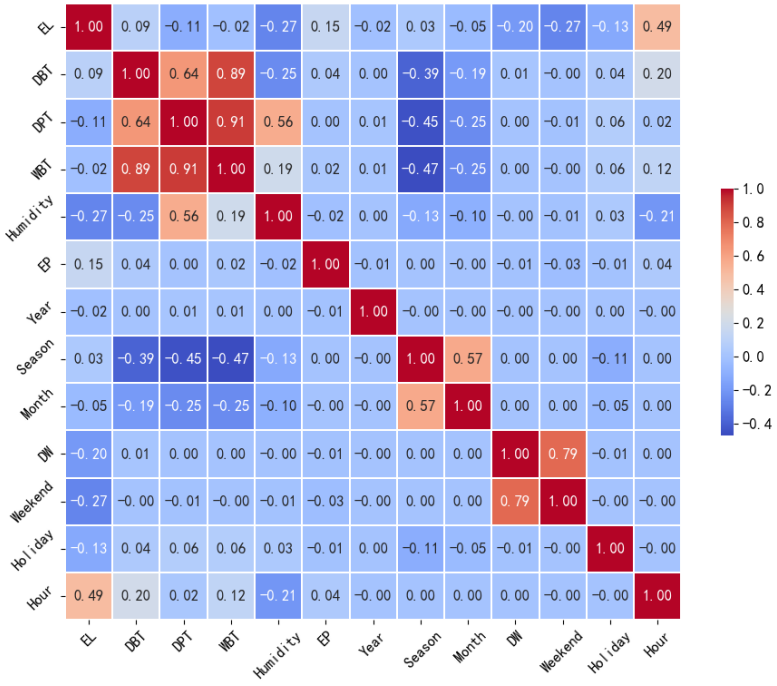


Figure 5. Pearson correlation coefficient graph of various features and power load.

Based on these results, we identified and eliminated features with low correlation to the electrical load. After carefully considering feature importance and model complexity, we selected features with absolute correlation coefficients greater than 0.1 as input variables for the model. The final selected input variables included 8 dimensions of data features, such as time of day, weekend indicator, and day of the week.

5.1.2. K-Means Clustering

The K-means clustering algorithm is a widely used unsupervised learning algorithm. Its core objective is to iteratively optimize the assignment of samples into K clusters to minimize the sum of squared errors within each cluster. By performing clustering analysis on the selected feature data and using the clustering results as inputs to the model, the algorithm can effectively reduce noise and outliers in the data. This process also helps to identify underlying patterns and structures, thereby enhancing the model's stability and predictive accuracy.

To determine the optimal number of clusters, this study employed the Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC). These methods help balance model complexity and goodness of fit. The analysis results indicated that the optimal number of clusters is 5. To visually demonstrate the clustering effect, we used Principal Component Analysis (PCA) to reduce the high-dimensional data to two dimensions and plotted the clustering results, as shown in Figure 6. For a more detailed understanding of the dataset, examples are provided in Appendix A.

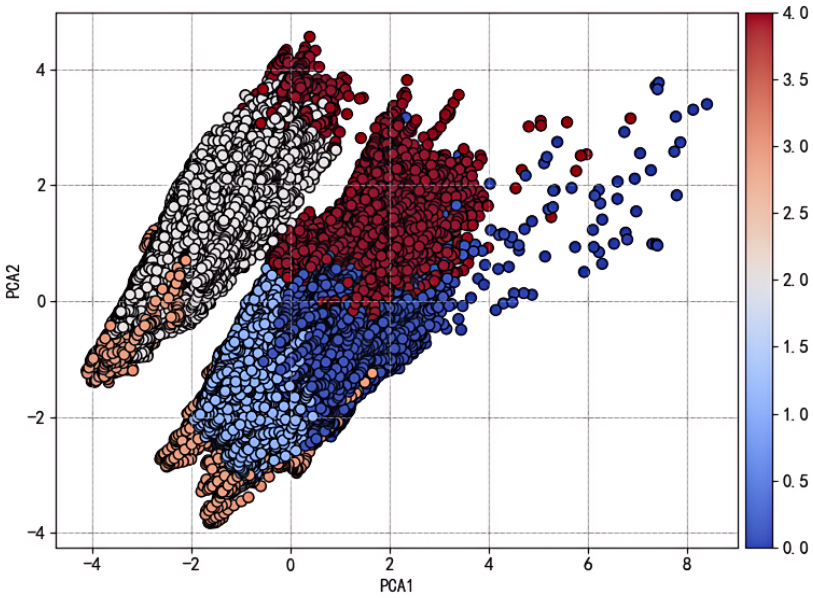


Figure 6. PCA two-dimensional projection of clustering results.

5.2. Feature Preprocessing

The classification and processing methods for input features are presented in Table 4. Firstly, the Min–Max normalization method was applied to scale the historical load from the previous day, electricity price, dew point temperature, humidity, and forecast time data to the interval [0, 1]. This ensures that numerical values of different features are on the same scale, avoiding errors caused by disparate magnitudes.

Table 4. Classification and processing methods of input features.

Feature Category	Feature Description	Processing Method
Historical load	Previous day's historical load	Normalized to 0–1
Economic feature	Electricity price	Normalized to 0–1
Meteorological features	Dew point temperature	Normalized to 0–1
	Humidity	Normalized to 0–1
Date features	Day of the week	Encoded as 1–7
	Weekend indicator	Encoded as 1 for weekend, 0 for weekday
	Holiday indicator	Encoded as 1 for holiday, 0 for non-holiday
	Forecast time	Normalized to 0–1
Other feature	Cluster category	Encoded as 0–4

Secondly, days of the week were represented using numbers 1 to 7 (with 1 representing Monday and 7 representing Sunday). Weekends and holidays were encoded using binary representation: 1 for weekends and 0 for weekdays; 1 for holidays and 0 for non-holidays.

Lastly, the clustering results were encoded as integers from 0 to 4 to distinguish different categories. This systematic feature processing approach ensures consistency and interpretability of the data throughout the model training process.

5.3. Evaluation Metrics

This study employs mean absolute percentage error (MAPE), root mean square error (RMSE), mean absolute error (MAE), and coefficient of determination (R^2) as performance evaluation metrics for the load forecasting model. The specific formulas are as follows:

$$X_{\mathrm{MAPE}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \times 100\% \tag{16}$$

$$X_{\mathrm{RMSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{17}$$

$$X_{\mathrm{MAE}} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{18}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{19}$$

where $\hat{y}_i$ represents the predicted load value at time i, y_i is the actual load value at the corresponding time, $\overline{y}$ is the mean of the load values, and n denotes the total number of test samples.

5.4. Experiment 1: Performance Comparison between PTCN-GRU and Other Deep Learning Models

In Experiment One, we provide a detailed list of hyperparameters for various deep learning models, as shown in Table 5. For models such as LSTM and GRU, we initially evaluated their performance based on the relevant literature. After that, we conducted extensive experiments through hyperparameter tuning and determined that the optimal number of layers is two.

Table 5. Configuration of deep learning model parameters.

Model	Parameter Description	Value
LSTM	Number of units in the first LSTM layer	128
	Number of units in the second LSTM layer	64
	Activation function	Tanh
GRU	Number of units in the first GRU layer	128
	Number of units in the second GRU layer	64
	Activation function	Tanh
CNN-GRU	Number of filters in the first CNN layer	64
	Kernel size	5
	Number of filters in the second CNN layer	32
	Kernel size	3
	Activation function	Relu
TCN-GRU	Number of filters in the first TCN layer	64
	Kernel size	5
	Number of filters in the second TCN layer	32
	Kernel size	3
	Activation function	Relu
	Dilation factor	[1, 2, 4]
PTCN-GRU	Number of filters in the first TCN	64
	Kernel size	5
	Number of filters in the second TCN	32
	Kernel size	3
	Activation function	Relu
	Dilation factor	[1, 2, 4]

For specific configurations, the learning rate for all models was uniformly set to 0.001 to ensure robust convergence. The dropout parameter was set to 0.2, effectively preventing overfitting and improving generalization capability. We set the batch size to 128, balancing the training speed and memory usage. The number of training epochs was set to 100, and we used an early stopping mechanism to enhance the training efficiency and avoid resource waste.

We chose mean squared error (MSE) as the loss function due to its effectiveness in regression tasks. The Adam optimizer was used to adjust the model parameters on the validation set, as it can adapt the learning rate during training.

Figure 7 illustrates the results of different models in the task of electric load forecasting. To provide a more intuitive comparison of the predictive performance of these models, Table 6 presents the MAE, MAPE, RMSE, and R^2 for each model. Additionally, Figure 8 offers a visual representation of these performance metrics, facilitating a clearer understanding of each model's performance.

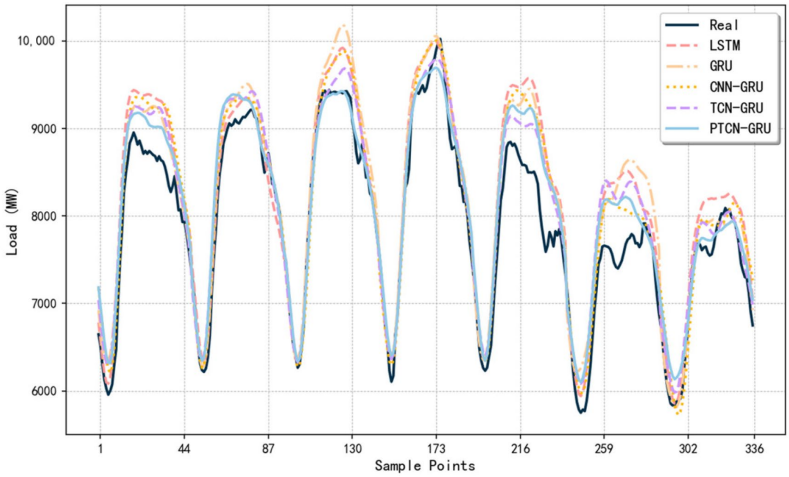


Figure 7. Power load prediction results of various models.

Table 6. Comparison of power load prediction results of various models (bold indicates the optimal result).

Model	MAE (MW)	MAPE (%)	RMSE (MW)	R^2
LSTM	346.8436	4.3748	438.7202	0.8266
GRU	341.5841	4.3148	424.5021	0.8376
CNN-GRU	297.5707	3.7730	369.4001	0.8770
TCN-GRU	273.5477	3.5415	343.8332	0.8935
PTCN-GRU	**255.2373**	**3.3499**	**322.7208**	**0.9062**

Based on the data from Figure 7 and Table 6, we conducted a detailed analysis of the performance of each model in electric load forecasting. After an in-depth examination of these performance metrics, we draw the following conclusions:

1. The PTCN-GRU model achieves a mean absolute error (MAE) of 255.2373 MW. This is significantly lower than all other models, indicating that the PTCN-GRU can more accurately approximate actual values when predicting electric load, minimizing the mean absolute error.
2. In terms of the mean absolute percentage error (MAPE), the PTCN-GRU model outperforms other models with a result of 3.3499%. This demonstrates its excellence in controlling relative errors and providing more reliable and stable predictions.
3. The root mean square error (RMSE) of the PTCN-GRU model is 322.7208 MW. This is 6.14% (21.1124 MW) lower than the second-best TCN-GRU and 26.44% (115.9994 MW)

lower than the poorest-performing LSTM. These results indicate that the PTCN-GRU model effectively reduces fluctuations in prediction errors, providing more stable forecasting results.

4. Regarding the R^2 metric, the PTCN-GRU achieves a score of 0.9062. This is an improvement of 1.27 percentage points over the second-best TCN-GRU and 7.96 percentage points over the poorest-performing LSTM. This implies that the PTCN-GRU model better explains data variance, showing the strongest correlation between predicted and actual values.

In conclusion, the PTCN-GRU model demonstrates superior performance in electric load forecasting. It outperforms other comparative models, including LSTM, GRU, CNN-GRU, and TCN-GRU, across key metrics such as the MAE, MAPE, RMSE, and R^2. The superior performance of the PTCN-GRU model stems from its combination of the TCN's long-term dependency modeling and GRU's nonlinear feature extraction. Its parallel structure and multi-scale convolution strategy prevent feature confusion, thereby enhancing the model's expressive power. The feature fusion mechanism integrates information across the different temporal scales and captures the complex feature interactions. These improvements enable the PTCN-GRU model to excel in electric load forecasting. It offers higher prediction accuracy and reliability, making it highly valuable for power system planning and operation.

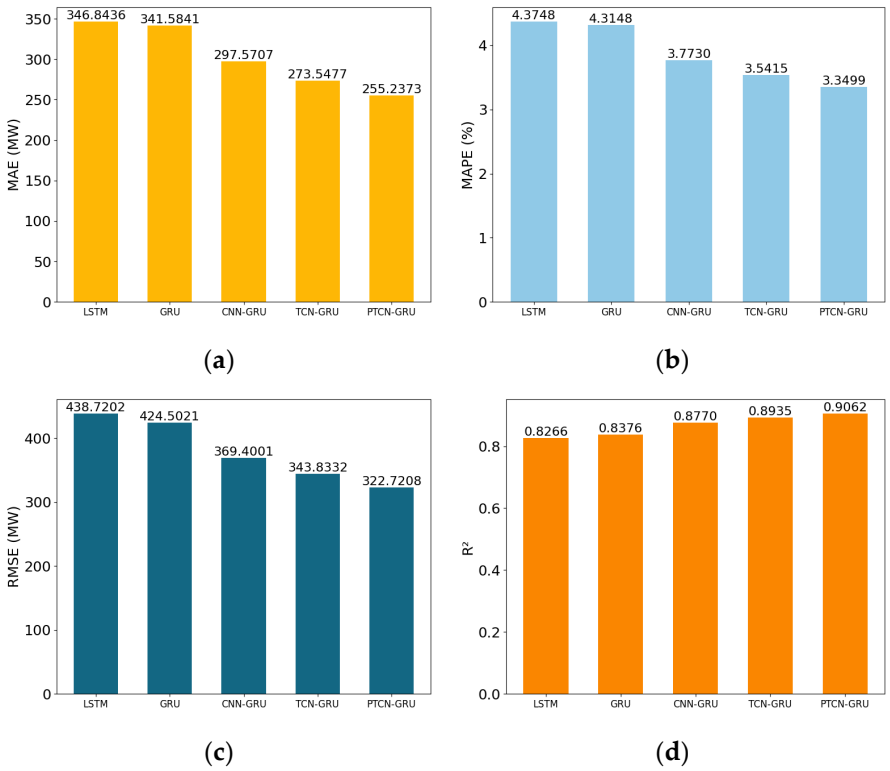


Figure 8. Prediction performance indicators obtained by various models: (**a**) MAE; (**b**) MAPE; (**c**) RMSE; (**d**) R^2.

5.5. Experiment 2: Performance Comparison of PTCN-GRU Model under Different Optimization Algorithms

In this study, we fixed the temporal dimension kernel sizes of the first and second TCN layers in the PTCN-GRU model to 5 and 3, respectively. Subsequently, we applied five optimization algorithms—PSO, WOA, GWO, DBO, and IDBO—separately to optimize six hyperparameters. The iteration count for each algorithm was set to 6. Table 7 presents the optimization results for each algorithm and the time taken for optimization.

Table 7. Hyperparameter optimization results.

Optimization Parameters	Range	PSO	WOA	GWO	DBO	IDBO
Number of filters in the first TCN	[8, 64]	28	12	18	8	64
Number of filters in the second TCN	[8, 64]	61	21	19	64	11
Number of units in the first GRU layer	[8, 128]	95	60	69	41	128
Number of units in the second GRU layer	[8, 128]	79	126	97	128	128
Batch size	[64, 256]	93	187	105	158	194
Learning rate	[0.0001, 0.01]	0.0016	0.007	0.0015	0.0025	0.0019
Time (s)	~	76,647	64,188	80,949	92,982	78,467

To evaluate the effectiveness of different optimization algorithms in tuning the hyperparameters of the PTCN-GRU model, we applied the optimal hyperparameters obtained by each algorithm to train the model and conducted predictions on the test set. Figure 9 illustrates the prediction results under different optimization algorithms, Table 8 lists the specific values of various performance metrics, and Figure 10 provides a graphical comparison of these metrics.

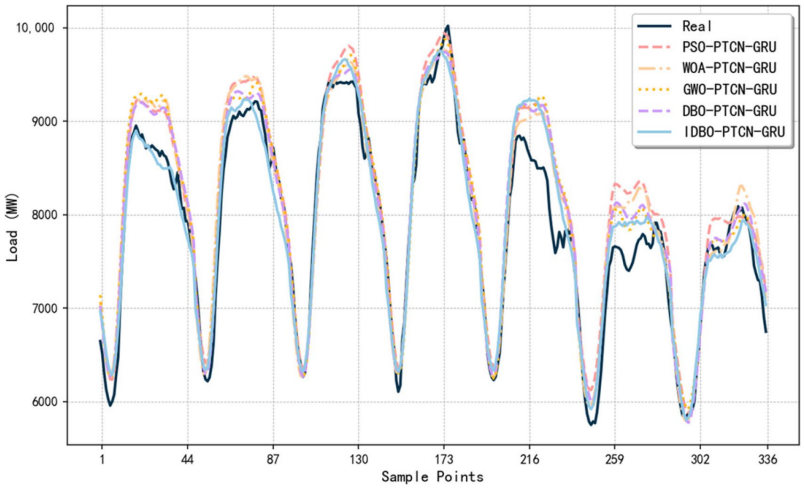


Figure 9. Power load prediction results of PTCN-GRU model under different optimization algorithms.

Table 8. Power load prediction results of PTCN-GRU model under different optimization algorithms (bold indicates the optimal result).

Model	MAE (MW)	MAPE (%)	RMSE (MW)	R^2
PSO-PTCN-GRU	269.6601	3.4632	333.2106	0.9000
WOA-PTCN-GRU	241.5022	3.0879	308.7808	0.9141
GWO-PTCN-GRU	233.4118	2.9734	298.4715	0.9197
DBO-PTCN-GRU	224.9236	2.8861	285.8699	0.9264
IDBO-PTCN-GRU	**191.1733**	**2.4694**	**244.6529**	**0.9461**

Through an in-depth analysis of the prediction results, we found that the PTCN-GRU model optimized by the IDBO achieved significant improvements across all evaluation metrics. Specifically, the MAE decreased from 255.2373 to 191.1733, representing a reduction of 25.1%. The MAPE reduced from 3.3499% to 2.4694%, marking a decrease of 26.3%. The RMSE declined from 322.7208 to 244.6529, a reduction of 24.2%. Additionally, the R^2 value increased from 0.9062 to 0.9461, an improvement of 4.4%. These data convincingly

demonstrate the superior effect of the IDBO algorithm in enhancing the performance of the PTCN-GRU model.

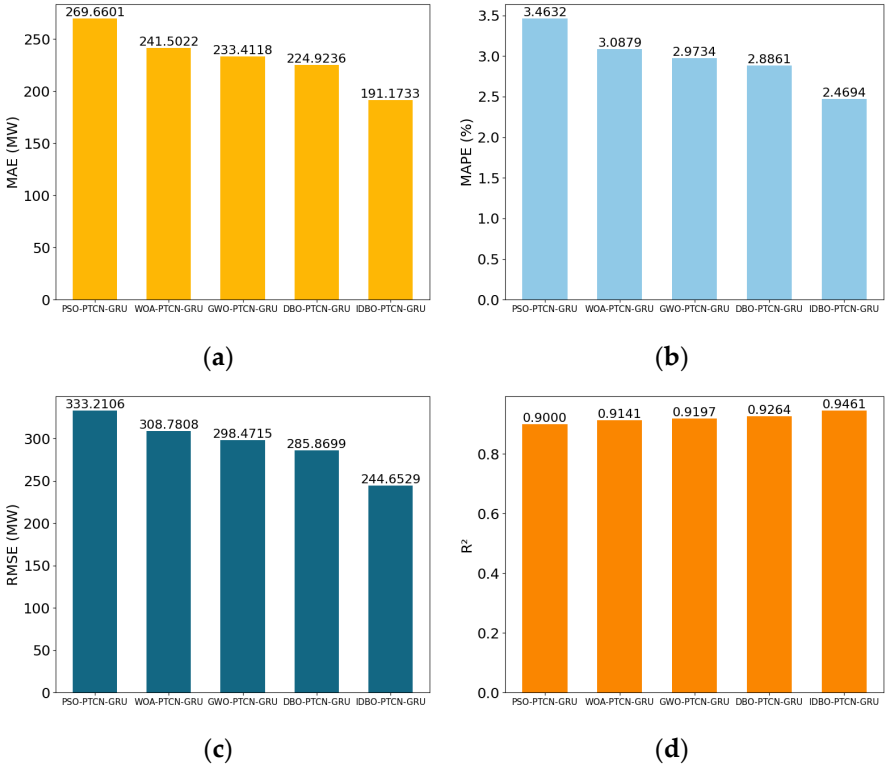


Figure 10. Prediction performance indicators of PTCN-GRU model under different optimization algorithms: (**a**) MAE; (**b**) MAPE; (**c**) RMSE; (**d**) R^2.

When comparing the performance of the IDBO-PTCN-GRU with the second-best model, DBO-PTCN-GRU, we observed significant advantages across all metrics. The MAE, MAPE, and RMSE of the IDBO-PTCN-GRU were lower than those of the DBO-PTCN-GRU by 15.01%, 14.44%, and 14.42%, respectively. This highlights the outstanding capability of the IDBO algorithm in reducing prediction errors. Meanwhile, the R^2 value of the IDBO-PTCN-GRU improved by approximately 2.13%. This further enhances the model's goodness of fit and demonstrates the effectiveness of the IDBO algorithm in optimizing the overall model performance.

In conclusion, the IDBO algorithm leverages its improved exploration and exploitation capabilities to demonstrate exceptional performance in optimizing the PTCN-GRU model. It not only significantly enhances the model's predictive accuracy but also shows distinct advantages over other optimization algorithms. This indicates that the IDBO algorithm possesses outstanding adaptability and efficiency in optimizing complex nonlinear systems such as electric load forecasting.

6. Conclusions and Future Prospects

To address the limitations of current short-term electric load forecasting models in capturing multi-scale features and handling complex influencing factors, such as holiday effects and weather conditions, this study proposes an IDBO-optimized PTCN-GRU electric load forecasting model. We conducted an empirical analysis using an electric load dataset from a region in Australia. The proposed model was then compared with advanced deep learning algorithms, including LSTM and GRU. The experimental results show that our model achieves superior predictive accuracy across key performance metrics, such as the MAPE, RMSE, MAE, and R^2. These findings confirm the model's effectiveness in improving electric load forecasting accuracy, as well as its strong adaptability and stability.

Although the IDBO-PTCN-GRU model demonstrates significant improvements in prediction accuracy, some limitations of this study must be acknowledged. Firstly, the performance of the model is highly dependent on the quality and quantity of the input data. If the data are sparse or contain high levels of noise, the accuracy of the model may decrease. In such cases, more advanced data preprocessing techniques may be required to improve the results. Secondly, the computational complexity of the model increases with the introduction of the IDBO algorithm. This could result in longer training times, making it less suitable for certain real-time applications. Additionally, while the IDBO algorithm enhances the global search capabilities and helps avoid the local optima, it may still struggle in extremely complex or high-dimensional optimization scenarios. Finally, the performance of the model has been evaluated on a specific dataset. Its generalizability to other datasets or different forecasting tasks may need further validation and adjustment.

To address the aforementioned limitations, future research can focus on the following areas:

1. Improving Data Quality and Preprocessing: develop more advanced data augmentation and noise-filtering techniques to enhance the model's performance in sparse or noisy data environments and explore methods for integrating multi-source data.
2. Optimizing Computational Efficiency: reduce computational complexity and improve the model's real-time application capabilities through efficient algorithms, distributed computing, or new model architectures.
3. Enhancing the Robustness of the IDBO Algorithm: improve parameter adjustment mechanisms and incorporate other optimization techniques to enhance the algorithm's performance in complex and high-dimensional scenarios.
4. Validating and Adjusting Model Generalizability: test the model across different datasets and application scenarios and optimize as necessary to ensure its broad applicability.

Author Contributions: Conceptualization, Z.W. and R.G.; methodology, Z.W.; validation, Z.W., Y.Q. and T.L.; writing—original draft preparation, Z.W.; writing—review and editing, Z.W., R.G. and J.X.; project administration, R.G.; funding acquisition, R.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 61561007, and the Natural Science Foundation of Guangxi Province, China, grant number 2017GXNSFAA198168.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

Appendix A

Table A1. The table of some examples from the load datasets.

Electricity Load	Hour	Dew Point Temperature	Humidity	Electricity Price	Weekday	Weekend	Holiday	Clustering Result
7168.7	0	15.6	83	18.9	7	1	0	2
6966.2	0.5	15.6	85	16.54	7	1	0	2
6783.41	1	15.6	87	16.79	7	1	0	2
6517.77	1.5	15.55	88	16.64	7	1	0	2
6260.54	2	15.5	89	15.48	7	1	0	2
6120.26	2.5	15.4	89.5	11.99	7	1	0	2
6050.79	3	15.3	90	8.99	7	1	0	2

Table A1. *Cont.*

Electricity Load	Hour	Dew Point Temperature	Humidity	Electricity Price	Weekday	Weekend	Holiday	Clustering Result
6016.9	3.5	15.15	90	8.34	7	1	0	2
6024.09	4	15	90	7.43	7	1	0	2
6057.58	4.5	14.9	90	9.7	7	1	0	2
6123.96	5	14.8	90	10.29	7	1	0	2
6233.32	5.5	14.5	89	13.11	7	1	0	2
6368.18	6	14.2	88	16.92	7	1	0	2
6500.9	6.5	14.6	88.5	16.46	7	1	0	2
6805.55	7	15	89	17.76	7	1	0	2
7212.22	7.5	15.05	83.5	18.33	7	1	0	2
7516.03	8	15.1	78	21.44	7	1	0	2
7868.87	8.5	15.15	73.5	22.02	7	1	0	2
8058.02	9	15.2	69	22.21	7	1	0	2
8207	9.5	15.6	66.5	22.5	7	1	0	2
8277.08	10	16	64	23.23	7	1	0	2
8313.29	10.5	15.2	60.5	25.65	7	1	0	2
8359.25	11	14.4	57	28.36	7	1	0	2
8367.19	11.5	14.5	57.5	29.76	7	1	0	2
8369.91	12	14.6	58	29.3	7	1	0	2
8304.84	12.5	15.05	60	26.77	7	1	0	2
8216.69	13	15.5	62	22.94	7	1	0	2
8175.09	13.5	15.9	64	23.9	7	1	0	2
8146.17	14	16.3	66	22.67	7	1	0	2
8190.38	14.5	16.4	66.5	23.02	7	1	0	2
8241.69	15	16.5	67	24.17	7	1	0	2
8314.83	15.5	16.8	69	27.13	7	1	0	2
8317.66	16	17.1	71	28.13	7	1	0	2
8277.12	16.5	16.65	69.5	26.08	7	1	0	2
8330.46	17	16.2	68	25.84	7	1	0	2
8385.39	17.5	16.5	70.5	25.83	7	1	0	2
8443.86	18	16.8	73	25.66	7	1	0	2
8676.79	18.5	16.8	73.5	32.18	7	1	0	2
8692.05	19	16.8	74	32.16	7	1	0	2
8541.22	19.5	16.85	74	29.18	7	1	0	2
8363.99	20	16.9	74	22.76	7	1	0	2
8097.05	20.5	17.15	75.5	21.55	7	1	0	2
7807.56	21	17.4	77	20.86	7	1	0	2

Table A1. *Cont.*

Electricity Load	Hour	Dew Point Temperature	Humidity	Electricity Price	Weekday	Weekend	Holiday	Clustering Result
7699.76	21.5	17.5	77.5	21.82	7	1	0	2
7528.29	22	17.6	78	19.29	7	1	0	2
7463.16	22.5	17.45	77.5	21.64	7	1	0	2
7283.74	23	17.3	77	19.75	7	1	0	2
7095.54	23.5	17.25	77	19.22	7	1	0	2

References

1. Hua, H.; Liu, M.P.; Li, Y.Q.; Deng, S.H.; Wang, Q.N. An Ensemble Framework for Short-Term Load Forecasting Based on Parallel CNN and GRU with Improved ResNet. *Electr. Power Syst. Res.* **2023**, *216*, 109057. [CrossRef]
2. Hu, H.X.; Zheng, B.Y. Short-term electricity load forecasting based on CEEMDAN-FE-BiGRU-Attention model. *Int. J. Low-Carbon Technol.* **2024**, *19*, 988–995. [CrossRef]
3. Guo, X.F.; Zhao, Q.N.; 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]
4. Xiao, L.Y.; Shao, W.; Liang, T.L.; Wang, C. A combined model based on multiple seasonal patterns and modified firefly algorithm for electrical load forecasting. *Appl. Energy* **2016**, *167*, 135–153. [CrossRef]
5. Cerne, G.; Dovzan, D.; Skrjanc, I. Short-Term Load Forecasting by Separating Daily Profiles and Using a Single Fuzzy Model Across the Entire Domain. *IEEE Trans. Ind. Electron.* **2018**, *65*, 7406–7415. [CrossRef]
6. Jalali, S.M.J.; Ahmadian, S.; Khosravi, A.; Shafie-khah, M.; Nahavandi, S.; Catalao, J.P.S. A Novel Evolutionary-Based Deep Convolutional Neural Network Model for Intelligent Load Forecasting. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8243–8253. [CrossRef]
7. Xiong, J.; Zhang, Y. A Unifying Framework of Attention-Based Neural Load Forecasting. *IEEE Access* **2023**, *11*, 51606–51616. [CrossRef]
8. Pavlatos, C.; Makris, E.; Fotis, G.; Vita, V.; Mladenov, V. Enhancing Electrical Load Prediction Using a Bidirectional LSTM Neural Network. *Electronics* **2023**, *12*, 4652. [CrossRef]
9. Zhu, X. Research on short-term power load forecasting method based on IFOA-GRNN. *Power Syst. Prot. Control* **2020**, *48*, 121–127. [CrossRef]
10. Ren, L.Z.L.; Wang, H.; Guo, Q. Short-term load interval prediction based on IPSO-GPR. *Comput. Eng. Des.* **2019**, *40*, 3002–3008. [CrossRef]
11. Gao, X.; Li, X.B.; Zhao, B.; Ji, W.J.; Jing, X.; He, Y. Short-Term Electricity Load Forecasting Model Based on EMD-GRU with Feature Selection. *Energies* **2019**, *12*, 1140. [CrossRef]
12. Deng, D.; Li, J.; Zhang, Z.; Teng, Y.; Huang, Q. Short-term Electric Load Forecasting Based on EEMD-GRU-MLR. *Power Syst. Technol.* **2020**, *44*, 593–602. [CrossRef]
13. Hafeez, G.; Khan, I.; Jan, S.; Shah, I.A.; Khan, F.A.; Derhab, A. A novel hybrid load forecasting framework with intelligent feature engineering and optimization algorithm in smart grid. *Appl. Energy* **2021**, *299*, 117178. [CrossRef]
14. Pan, Z.; Han, X. A Multi-dimensional Method of Nodal Load Forecasting in Power Grid. *Autom. Electr. Power Syst.* **2012**, *36*, 47–52.
15. Guo, Z.Z.G.; Zhou, X.; Hu, S.; Ma, G.; He, C. Research on electric heating load forecasting considering meteorological factors. *Electr. Meas. Instrum.* **2022**, *59*, 154–158. [CrossRef]
16. Göb, R.; Lurz, K.; Pievatolo, A. Electrical load forecasting by exponential smoothing with covariates. *Appl. Stoch. Models Bus. Ind.* **2013**, *29*, 629–645. [CrossRef]
17. Wei, Q.Z.M.; Cai, S.; Jiang, L.; Lu, L.; Zhang, Z.; Zhou, B. Medium and long-term load forecasting based on fractional order grey model optimized by BFGS-FA. *J. Guangxi Univ. Nat. Sci. Ed.* **2020**, *45*, 270–276. [CrossRef]
18. Su, Z.; Long, Y.; Zhao, L. Study on the Monthly Power Load Forecasting Performance Based on regARIMA Model. *Electr. Power* **2018**, *51*, 166–171.
19. Zhao, P.; Dai, Y. Power Load Forecasting of SVM Based on Real-time Price and Weighted Grey Relational Projection Algorithm. *Power Syst. Technol.* **2020**, *44*, 1325–1332. [CrossRef]
20. Li, Y.; Jia, Y.; Li, L.; Hao, J.; Zhang, X. Short term power load forecasting based on a stochastic forest algorithm. *Power Syst. Prot. Control* **2020**, *48*, 117–124. [CrossRef]
21. Gu, Y.; Ma, D.; Cheng, H. Power Load Forecasting Based on Similar-data Selection and Improved Gradient Boosting Decision Tree. *Proc. CSU-EPSA* **2019**, *31*, 64–69.
22. Li, S.; Goel, L.; Wang, P. An ensemble approach for short-term load forecasting by extreme learning machine. *Appl. Energy* **2016**, *170*, 22–29. [CrossRef]
23. Massaoudi, M.; Refaat, S.S.; Chihi, I.; Trabelsi, M.; Oueslati, F.S.; Abu-Rub, H. A novel stacked generalization ensemble-based hybrid LGBM-XGB-MLP model for Short-Term Load Forecasting. *Energy* **2021**, *214*, 118874. [CrossRef]

24. Eskandari, H.; Imani, M.; Moghaddam, M.P. Convolutional and recurrent neural network based model for short-term load forecasting. *Electr. Power Syst. Res.* **2021**, *195*, 107173. [CrossRef]
25. Kong, W.C.; Dong, Z.Y.; Jia, Y.W.; Hill, D.J.; Xu, Y.; Zhang, Y. Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Trans. Smart Grid* **2019**, *10*, 841–851. [CrossRef]
26. Wang, Z.; Zhao, B.; Ji, W.; Gao, X.; Li, X. Short-term Load Forecasting Method Based on GRU-NN Model. *Autom. Electr. Power Syst.* **2019**, *43*, 53–58.
27. Alhussein, M.; Aurangzeb, K.; Haider, S.I. Hybrid CNN-LSTM Model for Short-Term Individual Household Load Forecasting. *IEEE Access* **2020**, *8*, 180544–180557. [CrossRef]
28. Lu, J.X.; Zhang, Q.P.; Yang, Z.H.; Tu, M.; Lu, J.; Peng, H. A short-term load forecasting method based on CNN-LSTM hybrid neural network model. *Autom. Electr. Power Syst.* **2019**, *43*, 131–137.
29. Wu, L.Z.; Kong, C.; Hao, X.H.; Chen, W. A Short-Term Load Forecasting Method Based on GRU-CNN Hybrid Neural Network Model. *Math. Probl. Eng.* **2020**, *2020*, 1428104. [CrossRef]
30. Bai, S.; Kolter, J.Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271. [CrossRef]
31. Guo, L.; Xu, Q.S.; Zheng, L. A short-term load forecasting method based on TCN-GRU model. *Electr. Power Eng. Technol.* **2021**, *40*, 66–71.
32. Shi, H.; Wang, L.; Scherer, R.; Woźniak, M.; Zhang, P.; Wei, W. Short-term load forecasting based on Adabelief optimized temporal convolutional network and gated recurrent unit hybrid neural network. *IEEE Access* **2021**, *9*, 66965–66981. [CrossRef]
33. Cai, C.; Li, Y.; Su, Z.; Zhu, T.; He, Y. Short-term electrical load forecasting based on VMD and GRU-TCN hybrid network. *Appl. Sci.* **2022**, *12*, 6647. [CrossRef]
34. Hong, Y.; Wang, D.; Su, J.; Ren, M.; Xu, W.; Wei, Y.; Yang, Z. Short-term power load forecasting in three stages based on CEEMDAN-TGA model. *Sustainability* **2023**, *15*, 11123. [CrossRef]
35. Wang, X.P.; Wang, L.; Han, X.W.; Zhang, P.C.; Xu, H.R. Remaining useful life prediction method of engine based on k-means clustering and particle swarm optimization CNN-BiGRU-HAM. *Mach. Tool Hydraul.* **2024**, *52*, 1–16. Available online: http://kns.cnki.net/kcms/detail/44.1259.th.20231225.1719.002.html (accessed on 25 December 2023).
36. Sekhar, C.; Dahiya, R. Robust framework based on hybrid deep learning approach for short term load forecasting of building electricity demand. *Energy* **2023**, *268*, 126660. [CrossRef]
37. Li, M.W.; Xu, D.Y.; Geng, J.; Hong, W.C. A hybrid approach for forecasting ship motion using CNN–GRU–AM and GCWOA. *Appl. Soft Comput.* **2022**, *114*, 108084. [CrossRef]
38. Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014.
39. Xue, J.; Shen, B. Dung Beetle Optimizer: A New Meta-heuristic Algorithm for Global Optimization. *J. Supercomput.* **2022**, *79*, 7305–7336. [CrossRef]
40. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948. [CrossRef]
41. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
42. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
43. Arora, S.; Singh, S. Butterfly Optimization Algorithm: A Novel Approach for Global Optimization. *Soft Comput.* **2019**, *23*, 715–734. [CrossRef]

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