

Article

Assessing Residential Building Energy Efficiency Using Evolutionary Dendritic Neural Regression

Zhenyu Song ^{1,*} , Yajiao Tang ² , Shuangbao Song ³ , Bin Zhang ¹ and Cheng Tang ^{4,*} ¹ College of Information Engineering, Taizhou University, Taizhou 225300, China² College of Economics, Central South University of Forestry and Technology, Changsha 410004, China; t20060857@csuft.edu.cn³ School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou 213164, China; leadingsong@cczu.edu.cn⁴ Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819-0395, Japan

* Correspondence: songzhenyu@tzu.edu.cn (Z.S.); tang@ait.kyushu-u.ac.jp (C.T.)

Abstract: Assessing building energy consumption is of paramount significance in sustainability and energy efficiency (EE) studies. The development of an accurate EE prediction model is pivotal for optimizing energy resources and facilitating effective building planning. Traditional physical modeling approaches are encumbered by high complexity and protracted modeling cycles. In this paper, we introduce a novel evolutionary dendritic neural regression (EDNR) model tailored to forecasting residential building EE. Acknowledging the vast landscape and complexity of the EDNR weight space, coupled with the inherent susceptibility of traditional optimization algorithms to local optima, we propose a complex network-guided strategy-based differential evolution algorithm for training the EDNR model. This strategy adeptly strikes a balance between exploration and exploitation during the search process, significantly enhancing the predictive and generalization capacities of EDNR. To our knowledge, this study represents the inaugural application of dendritic neural regression in real-world prediction scenarios. Extensive experimental findings demonstrate the efficacy of EDNR in accurately predicting building EE with commendable performance. Furthermore, the results of two nonparametric statistical tests affirm the validity and stability of EDNR. Consequently, our proposed methodology exhibits high potential and competitiveness in machine learning applications within the energy domain.

Keywords: energy efficiency; dendritic neural regression; buildings; differential evolution



Citation: Song, Z.; Tang, Y.; Song, S.; Zhang, B.; Tang, C. Assessing Residential Building Energy Efficiency Using Evolutionary Dendritic Neural Regression. *Electronics* **2024**, *13*, 1803. <https://doi.org/10.3390/electronics13101803>

Academic Editor: Ping-Feng Pai

Received: 2 April 2024

Revised: 3 May 2024

Accepted: 4 May 2024

Published: 7 May 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

With the advancement of urbanization and the continuous improvement in people's quality of life, the proportion of urban building energy consumption (EC) is increasing annually. EC is one of the main causes of the increase in greenhouse gas emissions and the increased inhalation amount, and building EC accounts for a large proportion of the total EC [1,2]. Building energy efficiency (EE) and EC are interdependent. Enhancing building EE is key to achieving sustainable building development, which can be realized through reducing building EC. Therefore, improving building EE can decrease building EC, which in turn enhances building EE. As an effective means to reflect the operational characteristics of buildings, building EE prediction is an important tool for assessing the energy-saving measures and energy-portal parts of smart buildings and can provide effective support for the intelligent management of buildings [3]. Therefore, it is important to explore the mechanism and patterns of building EE and to establish an accurate and effective model for predicting EE in buildings.

In recent years, numerous scholars have presented various techniques for predicting building EC and EE [4,5]. Currently, the techniques utilized for building EC prediction consist mainly of engineering methods, statistical methods, and artificial intelligence

methods. The engineering approach involves utilizing substantial professional knowledge of building physics during the modeling phase to obtain building EC data for analysis and research. This is achieved by developing an accurate building physics model and combining it with a building operation simulation. Shabunko et al. [6] employed the EnergyPlus V2018 energy simulation software to simulate the EC of three different residential buildings with a total of four hundred units. They subsequently compared the intensity of the building EC with real EC data and found that the simulation results were consistent with the actual data. The authors concluded that EnergyPlus software is an efficient tool for simulating building EC. Asadi et al. [7] altered the design variables of building materials, including thickness, appearance, and internal personnel scheduling. To assess the EC of buildings under each design scheme, tens of thousands of simulations were run using the eQUEST V3.65 and DOE-2 V.2 building simulation software. This study employed multiple linear regression equations to predict building EC. The study results indicated that the multiple linear regression model's EC predictions deviated less than 5% from the simulation software's results. The outcome precision of these approaches depends heavily on the utilization of specific knowledge in building physics, which is intricate due to the complex system setup and numerous design parameters involved in the modeling phase [5].

Statistical methods are utilized to construct mathematical models by tabulating historical EC data, following which EC is forecasted and analyzed. Zeng et al. [8] employed Gaussian process regression to forecast the actual electricity consumption of buildings of various types. The findings revealed that Gaussian process regression is capable of predicting not only the electricity consumption of different inputs, but also that of buildings belonging to diverse categories. Atalay et al. [9] developed a multiple linear regression model, a time series model, and a discrete grey model for predicting HVAC power consumption in commercial buildings located in Paris, France. The experimental results indicated that the multiple linear regression model outperformed both the time series model and the grey model. The ARIMA model predicts the EC and greenhouse gas emissions of iron and steel enterprises and offers excellent EC prediction performance [10]. Li et al. [11] introduced an EC prediction model, IPSO-ANN, where the input variables were established using the dimensionality reduction method of principal component analysis. The results demonstrated that the IPSO-ANN model provides higher prediction accuracy than its counterparts. Although the aforementioned prediction models can accurately predict EC, the random and nonlinear nature of building EC data reveals deficiencies in these methods [12]. These deficiencies are reflected in three main ways: first, the general statistical model fails to account for the nonlinear relationship between the input and output; second, the predictive performance of extended linear regression relies heavily on the correct selection of the activity function, which poses a major challenge; third, the multicollinearity of the input features may influence the prediction results of the statistical model.

In predicting building EC, machine learning methods, also referred to as data-driven methods, can be more efficient than traditional approaches such as engineering methods. Modeling relies on a range of mathematical and statistical algorithms and a wealth of historical data regarding building EC. Typically, building operational data, weather data, internal load data, and other relevant information are used as the inputs, and the corresponding building EC is used as the output. Models for predicting EC can be obtained through training. At present, the prevalent models include artificial neural networks [13], support vector machines [14,15], random forests [16,17], and decision trees (DTs) [18]. Due to the complexity of building EC behavior and the volatility of energy demand, obtaining accurate and reliable building EC prediction results remains a challenging task. Furthermore, in recent years, the scale and dimensions of building EC data have increased significantly due to continuous hardware upgrades and more frequent data collection. This increase poses a significant challenge to the prediction ability of building EC models. Consequently, an increasing number of researchers are focusing on deep learning algorithms that offer superior learning abilities [19,20].

Energy efficiency prediction is a typical machine learning problem. Single prediction models are widely used due to their rapid calculation speed and ease of implementation. However, there are still some shortcomings. First, the multiple linear regression method, which is fast and easy to implement, is inadequate for complex problems. Second, support vector machines are better at balancing prediction accuracy and calculation speed, but determining the optimal kernel function is challenging. Furthermore, in comparison to the previous two approaches, artificial neural networks can handle nonlinear issues and have relatively high forecasting accuracy; however, they require significant historical data for model training and the determination of numerous parameters. Traditional optimization algorithms, such as the back propagation (BP) algorithm, are sensitive to initial conditions and easily fall into local optima [21]. However, evolutionary algorithms differ from traditional algorithms; they are population-based intelligent search methods with self-adaptability, self-adjustment, and parallel search capabilities. Evolutionary algorithms are capable of solving functions that are nondifferentiable, nonmicroscopic, or discontinuous, even for high-dimensional decision variables. Additionally, they can find approximations of optimal values for uncertain functions after several iterations. The common evolutionary algorithms include the genetic algorithm (GA) [22], the states of matter search (SMS) algorithm [23], particle swarm optimization (PSO) [24,25], the cuckoo search (CS) algorithm [26], the firefly algorithm (FA) [27], the gravitational search algorithm (GSA) [28], and differential evolution (DE) [29]. The “no free lunch” theorem has been proven, which states that there is no universally applicable optimization algorithm or learning method when searching for the optimal solution or learning the best model. In other words, there is no one method that performs best in all situations for different problems. This theorem emphasizes the importance of algorithm selection and design, as the performance of an algorithm depends on the characteristics of the problem [30,31]. Therefore, to address the task of EE prediction, we designed a variant algorithm based on DE. In DE, the best individuals are used to generate the next generation via mutation, recombination, crossover, and selection operations. This evolutionary mechanism exhibits superior search performance in comparison to the conventional evolutionary algorithm. Conventional DE performs optimally in addressing optimization issues featuring relatively simple function structures, but falls short in addressing complex optimization issues. This is particularly applicable in optimization models that have complex structures, where the decision variables have high dimensionality and the objective function has multiple local optima. In such situations, DE converges slowly and may experience stagnation during the search process.

By considering the energy field and relevant data features in detail, we attempt to apply an improved dendritic neuron model to energy-related fields. In our previous work, the effectiveness of the dendritic neuron model was established in various areas, including medical diagnosis [32,33], classification [21,34–36], time series prediction [37–40], and multiobjective optimization [41,42]. This paper proposes the evolutionary dendritic neural regression (EDNR) model for predicting building EE. As the weight space involved in EDNR is complex and has a large landscape, we further propose a complex network-based differential evolution (CNDE) algorithm as a global optimization algorithm for training EDNR. The proposed CNDE algorithm can increase individual diversity during the optimization search process while maintaining computational efficiency. This enables DE to achieve the global optimum, which enhances the accuracy of EDNR in building energy efficiency prediction. The extensive experimental results demonstrate that EDNR possesses considerable potential for predicting building EE, leading to satisfactory results in the prediction of both heating and cooling loads. Using simulated datasets, we propose EDNR, which effectively captures the nonlinear relationships between features and labels. Thus, this work aims to provide more competitive prediction methods for the field of building EE, while further expanding the interdisciplinary research between machine learning and other domains.

In summary, the primary innovations and contributions of this study are as follows: (a) A new method for residential building EE prediction is proposed and applied for the first time. (b) A population guidance strategy based on the scale-free property of complex networks is proposed to enhance the optimization capability of DE. (c) The proposed CNDE algorithm can achieve a better balance between exploration and exploitation in the search process, and the experimental results show that EDNR has better accuracy and stability in predicting EE.

The remainder of this paper is organized as follows: Section 2 introduces the relevant work on EDNR and CNDE in detail; Section 3 describes the experimental setup, parameter settings, experimental data, evaluation metrics, experimental results, and discussion; finally, Section 4 provides a summary and discusses future work.

2. Materials

In this study, we present an EDNR model trained through the differential evolution of complex network structure topologies. It is capable of effectively performing prediction or classification tasks in machine learning by mapping the nonlinear relationship between input feature information and target results. In this section, we first introduce the structural framework of DNR and then present its learning algorithm in detail.

2.1. Dendritic Neural Regression

DNR and traditional MLP models share relatively simple topologies. MLPs utilize information interactions among multiple neurons to realize nonlinear relations, which may result in slow model training and the easy attainment of local optimal solutions. In contrast, DNR uses a unique dendritic structure to acquire a nonlinear interpretation of the problem. Additionally, it exhibits a faster convergence rate and higher prediction accuracy. A DNR model comprises four layers: synaptic, dendrite, membrane, and cellular layers. It exemplifies a typical feedforward model, as illustrated in Figure 1. Following appropriate calculations within each layer, the data are transmitted to the subsequent layer until the model produces its final output.

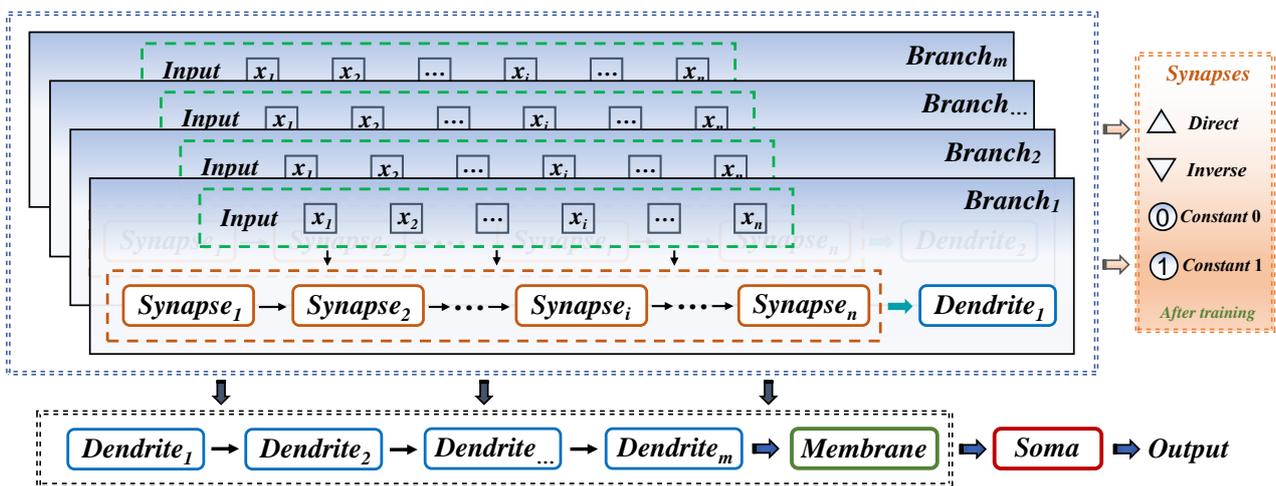


Figure 1. Architectural description of EDNR.

2.1.1. Synaptic Layer

The synaptic layer, which typically links another neuron or cell body, is a vital structure for signaling. In DNR, the input signal passes through the synaptic layer of each branch before being transmitted to the dendrite layer. We define the function related to the synaptic layer for the i -th input on the m -th branch as follows:

$$S_{i,m} = \frac{1}{1 + e^{-k(w_{i,m}x_i - h_{i,m})}}, i \in [1, 2, \dots, n], m \in [1, 2, \dots, M], \quad (1)$$

where $S_{i,m}$ represents the output of the synaptic layer and serves as the input for the next layer; x_i denotes the i -th input, which must be standardized within the range of 0 to 1; and $w_{i,m}$ and $h_{i,m}$ represent the weights and thresholds, respectively, which are obtained through the training of the learning algorithm. Additionally, depending on the values of $w_{i,m}$ and $h_{i,m}$, as illustrated in Figure 2, the synaptic layer encompasses six connection states, which may be interpreted as follows:

(a) Constant 1 connection: If there is any input, Figure 2a shows that the output is approximately 1.

$$C_a = \{(w, h) \mid h_{i,m} < 0 < w_{i,m} \text{ or } h_{i,m} < w_{i,m} < 0\} \tag{2}$$

(b) Constant 0 connection: If there is any input, Figure 2b shows that the output is approximately 0.

$$C_b = \{(w, h) \mid 0 < w_{i,m} < h_{i,m} \text{ or } w_{i,m} < 0 < h_{i,m}\} \tag{3}$$

(c) Direct connection: The output is proportional to the input, as illustrated in Figure 2c.

$$C_c = \{(w, h) \mid 0 < h_{i,m} < w_{i,m}\} \tag{4}$$

(d) Inverse connection: The output is inversely proportional to the input, as illustrated in Figure 2d.

$$C_d = \{(w, h) \mid w_{i,m} < h_{i,m} < 0\} \tag{5}$$

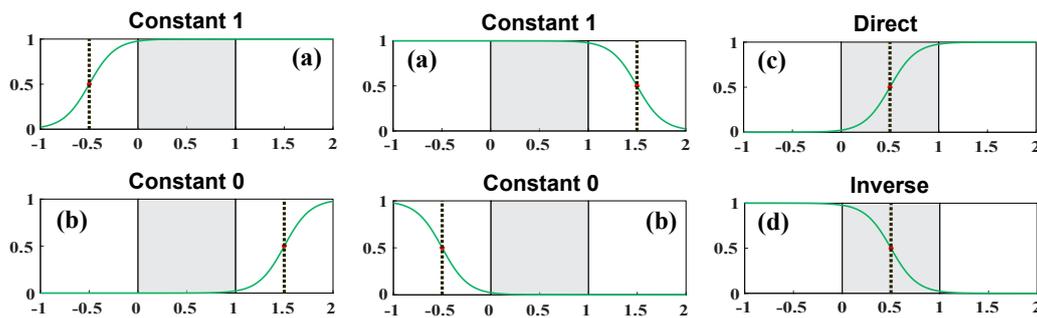


Figure 2. Connection cases of synapses.

2.1.2. Dendritic Layer

The dendritic layer connects the synaptic layer nodes of each branch in series, and due to the nonlinearity of the synaptic layer, the collected information is multiplied by the following formula:

$$D_m = \prod_{i=1}^N S_{i,m}, \tag{6}$$

where D_m represents the output of the m -th layer branch and N denotes the number of synaptic layers.

2.1.3. Membrane Layer

The cell membrane connects the dendritic layers of each branch, gathers incoming information from those layers, and delivers it to the cell body layer via a summation operation. This operation can be defined as

$$\mathcal{V} = \sum_{m=1}^M D_m, \tag{7}$$

where \mathcal{V} denotes the calculated output result of the membrane layer and M represents the number of dendritic layers in the DNR model.

2.1.4. Cell Body (Soma)

After multiple layers of processing, the data from the sample are eventually conveyed to the cell body layer. By utilizing the sigmoid function as the activation function to calculate the final output, the cell body layer can be represented as

$$\mathcal{O} = \frac{1}{1 + e^{-k_{soma}(\mathcal{V} - \theta_{soma})}}, \tag{8}$$

where θ_{soma} is the threshold value controlling the activation or inhibition of neurons. Both θ_{soma} and k_{soma} are positive integers.

2.2. Learning Algorithm

This section initially outlines the structure of the conventional DE algorithm, followed by an introduction to existing complex network models, including the principles and characteristics of scale-free network models in complex networks. Subsequently, by integrating the concept of the node degree in complex networks into DE, a complex network model based on the DE algorithm is proposed, including a description of the framework and implementation process of CNDE.

2.2.1. Differential Evolution

DE is a population-based stochastic optimization algorithm that utilizes genetic evolution. Similar to other evolutionary algorithms, DE incorporates four genetic operations: initialization, mutation, crossover, and selection. DE leverages the difference information between population individuals to facilitate mutation, followed by a probability-based approach for integrating crossover. Finally, the algorithm enhances the population through a greedy selection mechanism. Obviously, DE can retain superior individuals in the upcoming generation, leading to a larger number of high-quality individuals. The algorithm's comprehensive procedure is outlined below:

A. Initialization:

First, a random function conforming to a uniform distribution is employed in DE to generate the initial population. The population size Ps and the search space dimension D are determined by the following formula:

$$x_{i,j}^{(o)} = x_{i,j}^{(low)} + rand(0, 1) \cdot (x_{i,j}^{(up)} - x_{i,j}^{(low)}), \tag{9}$$

where $i \in [1, 2, \dots, Ps]$, $j \in [1, 2, \dots, D]$, and $x_{i,j}^{(up)}$ and $x_{i,j}^{(low)}$ denote the upper and lower bounds that individuals search within the j -th dimension, respectively. Additionally, $rand(0,1)$ represents decimal values that are uniformly distributed within the interval $(0, 1)$.

B. Mutation:

In evolutionary computation, mutation refers to altering the value of a particular location through random perturbations. The preservation of population diversity through the manipulation of mutations is a critical evolutionary technique in DE. Several typical mutation methods are listed as follows:

- DE/rand/1

$$\mathcal{V}_i^{(t)} = \mathcal{X}_1^{(t)} + \mathcal{R} \cdot (\mathcal{X}_2^{(t)} - \mathcal{X}_3^{(t)}), \tag{10}$$

- DE/best/1

$$\mathcal{V}_i^{(t)} = \mathcal{X}_{best}^{(t)} + \mathcal{R} \cdot (\mathcal{X}_1^{(t)} - \mathcal{X}_2^{(t)}), \tag{11}$$

- DE/rand/2

$$\mathcal{V}_i^{(t)} = \mathcal{X}_1^{(t)} + \mathcal{R}_1 \cdot (\mathcal{X}_2^{(t)} - \mathcal{X}_3^{(t)}) + \mathcal{R}_2 \cdot (\mathcal{X}_4^{(t)} - \mathcal{X}_5^{(t)}), \tag{12}$$

- DE/best/2

$$\mathcal{V}_i^{(t)} = \mathcal{X}_{best}^{(t)} + \mathcal{R}_1 \cdot (\mathcal{X}_1^{(t)} - \mathcal{X}_2^{(t)}) + \mathcal{R}_2 \cdot (\mathcal{X}_3^{(t)} - \mathcal{X}_4^{(t)}), \tag{13}$$

where \mathcal{R} , \mathcal{R}_1 , and \mathcal{R}_2 are scaling factors with values between 0 and 1 and $\mathcal{X}_{best}^{(t)}$ represents the current leading individual after the t -th iteration. The aforementioned four mutation strategies exhibit varying search effects, thus requiring a reasonable balance between exploration and exploitation depending on the status of global and local searches when tackling optimization problems.

C. Crossover:

The previous mutation operation generates a vector \mathcal{V}_i . Recombination will be performed on the target vectors \mathcal{X}_i and \mathcal{V}_i during the crossover operation to enhance population diversity. The formula for the j -th dimension is defined as follows:

$$u_{i,j}^{(t)} = \begin{cases} v_{i,j}^{(t)}, & \text{if } \text{rand}_j(0,1) \leq C_r \text{ or } j = j_{rand} \\ x_{i,j}^{(t)}, & \text{otherwise} \end{cases}, \tag{14}$$

where C_r represents the crossover probability, and it is clear that C_r determines the number of target vector individuals; $\text{rand}(0,1)$ indicates a random decimal value between 0 and 1; $j = j_{rand}$ represents a random integer within the interval $[1, D]$.

D. Selection:

To maintain a constant population size during the iteration process, DE generates offspring using one-to-one greedy selection between the target and test individuals. This means that the individuals with better fitness are retained for the next generation. The process is as follows:

$$\mathcal{X}_i^{(t+1)} = \begin{cases} \mathcal{U}_i^{(t)}, & \text{if } f(\mathcal{U}_i^{(t)}) \leq f(\mathcal{X}_i^{(t)}) \\ \mathcal{X}_i^{(t)}, & \text{otherwise.} \end{cases} \tag{15}$$

Notably, DE employs a greedy selection process, which can swiftly arrive at the optimal solution of a function. However, excessive greediness can cause the algorithm to converge early before reaching the global optimum, resulting in the model falling into a local optimum. Therefore, maintaining diversity within the population is crucial throughout the algorithm's iterations. To achieve this goal, we developed a complex-network-model-based DE algorithm that effectively ensures population diversity during the execution of DE.

2.2.2. Complex Network Model

Complex networks exhibit self-organization, self-similarity, attractor, small-world, and scale-free properties. These networks can be categorized into random, small-world, and scale-free networks. Many real-world networks typically originate at a small scale and progress through the inclusion of new nodes and relationships. Barabási and Albert (BA) identified two evolutionary mechanisms for growth and preferred connection, from which they established a BA scale-free network evolution model. The degree distribution of nodes in this model follows a power-law distribution:

$$P(k) \propto k^{-\beta}, \tag{16}$$

where β represents the scale-free factor, which typically takes a value of (2,3), and $P(k)$ denotes the probability that a node is connected. The BA model is generated as follows:

- Initialization:

First, a network is created with m_0 nodes at time t_0 , which are typically fully connected or regular in a particular topology. The total number of nodes is subsequently recorded as N_0 for the initialized network $T(0)$.

- **Growth:**
A new node n_i enters the current network $T(i)$ at t_i ($i \in \mathbb{Z}^+$) with m ($m \leq m_0$) initially connected edges, and n_i is connected in the selection of m nodes in the network $T(i)$ without repeating connections.
- **Preferential connections:**
The probability of node n_j in the current network $T(i)$ acquiring a connected edge to a new node n_i is determined by the following equation:

$$\prod(k_j) = k_j / \sum_{h=1}^{N_i} k_h, \quad (17)$$

where N_i represents the scale of the current network $T(i)$ at t_i and the denominator denotes the total degree of $T(i)$. Figure 3 illustrates the evolution of the initial network scale and total degree for $m_0 = 3$ and $N_0 = 6$, respectively. Notably, there is only one new node connecting edges in this process ($m = 1$), and the number of evolutions T is 5. As shown in Figure 3, the majority of edges connecting new nodes to the network are linked to nodes with higher degrees, suggesting incipient power law behavior in the degree distribution of the network $T(i)$.

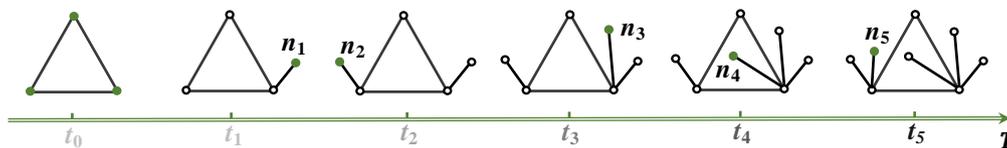


Figure 3. The generation process of the scale-free network.

2.2.3. Proposed Complex-Network-Based DE

To utilize EDNR for regression or prediction, effective supervised training is a prerequisite and is generally considered an optimization problem of the weight space. While DE algorithms have demonstrated satisfactory performance in numerous optimization problems, the efficacy of DE is dependent on relevant parameters and genetic operators, including the mutation operator, crossover operator, population size, and crossover probability. Although the performance of DE algorithms has improved somewhat with the implementation of parameter tuning strategies or operator design [43,44], it remains unsatisfactory in addressing complex optimization problems. Therefore, regarding the diversity of the population involved in generating new individuals, the scale-free network model is incorporated into the DE search process. To enhance accuracy and efficiency, the population is sorted after each iteration based on the node degree and individual fitness from the lowest to the highest in the complex network model. Consequently, the proposed algorithm is called CNDE.

From the process of generating the BA model, the network exhibits a greater number of nodes with a low degree and fewer nodes with a high degree. The scale-free property of this intricate network model necessitates altering the positions of individuals after each iteration of the DE model to efficiently sustain the disparity between individuals. Notably, to enhance the search efficiency of CNDE, a scale-free network that corresponds to the population size must be developed before executing DE. This model stores the individuals after each DE execution and selects nodes based on the relative degree of fitness. Figure 4 displays the network model in the search space. During the CNDE search process, high-quality individuals are deposited in high-degree nodes, while lower-quality individuals are placed in low-degree nodes. Furthermore, each individual randomly selects a neighbor for position swapping, thereby preventing a superindividual from leading the population into a local optimum. Based on this mechanism, CNDE can effectively balance exploration and exploitation. The algorithm framework of our proposed CNDE is essentially similar to that of traditional DE. The complexity of CNDE is not greatly increased, but individual

positions are adjusted after each iteration. To achieve this, we modified the update rules for individual locations in DE:

$$\mathcal{V}_i^{(t)} = \mathcal{X}_i^{(t)} + \text{rand}(0,1) \cdot (\mathcal{X}_{i,\text{neighbor}}^{(t)} - \mathcal{X}_i^{(t)}), \quad (18)$$

where $\mathcal{V}_i^{(t)}$ represents the updated solution and $\mathcal{X}_{i,\text{neighbor}}^{(t)}$ is a neighbor randomly selected from the current state of $\mathcal{X}_i^{(t)}$. From the above formula, it is evident that the exchange of information among proficient individuals will decrease, while inferior individuals will have a greater probability of gaining more beneficial information from competent individuals to enhance their knowledge, thereby efficiently sustaining population diversity, indicating that CNDE produces a trade-off between global exploration and local exploitation. The procedure for implementation is outlined in Algorithm 1.

Algorithm 1: Complex-network-based DE (CNDE).

Input: Parameters m_0, F_0, Cr_0, Ps , and G_{max}

Output: Globally optimal solution

begin

The maximum number of iterations of DE (G_{max}) and population size (Ps) are determined.

A BA model with scale-free properties is created based on the initialized m_0 .

A complex network is built until the number of nodes is Ps .

The Ps solution space is randomly initialized.

The fitness of each solution is evaluated.

while $t < G_{max}$ **do**

The solutions are sorted and labeled according to their fitness.

The solutions are placed in order into the generated complex network nodes.

Neighbors are randomly selected, and the positions of the solutions are updated.

$$v_{i,j}^{(t)} = x_i^{(t)} + \text{rand}(0,1) (x_{i,\text{neighbor}}^{(t)} - x_i^{(t)})$$

for $j = 1 : D$ **do**

Mutation and crossover operations are performed.

$$v_i^{(t)} = x_i^{(t)} + \mathcal{R}_1^{(t)} (x_{best}^{(t)} - x_i^{(t)}) + \mathcal{R}_2^{(t)} (x_{r_1}^{(t)} - x_{r_2}^{(t)}),$$

$$u_{i,j}^{(t)} = \begin{cases} v_{i,j}^{(t)}, & \text{if } \text{rand}_j(0,1) \leq Cr \cup j = j_{rand} \\ x_{i,j}^{(t)}, & \text{otherwise} \end{cases}$$

The fitness of each current solution is evaluated.

A selection operation is performed to determine the individuals for the next iteration.

$$\mathcal{X}_i^{(t+1)} = \begin{cases} \mathcal{U}_i^{(t)}, & \text{if } f(\mathcal{U}_i^{(t)}) \leq f(\mathcal{X}_i^{(t)}), \\ \mathcal{X}_i^{(t)}, & \text{otherwise.} \end{cases}$$

$t = t + 1$.

The globally optimal solution is returned.

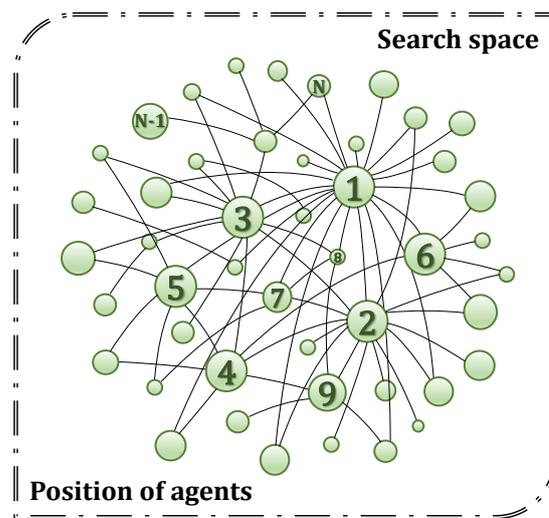


Figure 4. Complex network model with scale-free characteristics in the search space.

3. Experimental Studies

To validate the predictive ability of the proposed EDNR model for building EE, we conducted comprehensive comparative experiments, including DNR training with different heuristic algorithms to enhance its performance, and popular machine learning models were chosen as competitors. The experimental findings demonstrate that the proposed CNDE algorithm can train a DNR model effectively, and the resulting EDNR model excels in predicting both heating and cooling loads for building EE. The experiments described in this section had the following components: description of the dataset, experimental setup, normalization of the data, parameter configuration and discussion, interpretation of the performance evaluation metrics, comparison and analysis of the experimental results, and computational complexity analysis. Finally, a statistical analysis of the experimental results was performed. To confirm the efficacy and superiority of the proposed method, we selected eight widely used intelligent algorithms as competitors to CNDE for training DNR models. Additionally, eight competitive machine learning models were selected for comparative experiments with EDNR. To prevent the possibility of experimental error, we conducted the experiments for each set of methods 30 times independently and calculated the average of the results. The experiments were conducted on a personal computer equipped with a 13th Gen Intel(R) Core(TM) i7-13770F 2.10 GHz CPU, with 32 GB of memory, and the MATLAB R2020a software.

3.1. Dataset Description and Normalization

In this study, we used a standard dataset created by Angeliki Xifara, a British engineer, and processed by the Centre for Industrial and Applied Mathematics at the University of Oxford, U.K., to study and model the heating and cooling load requirements (energy efficiency) of buildings. This dataset is mainly used to study or model the heating and cooling loads of buildings. The researchers used Ecotect to simulate the energy profiles of 12 different building shape characteristics, such as building orientation, height, roof area, glazing distribution, and size. A total of 768 building samples were obtained from the simulation, where each data sample set contains eight features and two real values of EE loads and all attributes are positive integers or decimals.

Data normalization is a crucial step in data preprocessing and has significant implications for both machine learning and time similarity searches. Normalization aims to provide equal weight to all data attributes, simplifying comparisons and aggregations between attributes. Since the input range of the EDNR synaptic layer is between 0 and 1, applying a normalization operation to the raw data is necessary, which can not only

reduce the computational complexity, but also aid in algorithm convergence. The formula for normalization is as follows:

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}, \tag{19}$$

where x_{max} and x_{min} represent the maximum and minimum values of the data, respectively, and x'_i denotes the normalized data. Notably, since the soma layer of the EDNR model outputs within the interval of [0,1], the final objective of the experiment is to determine the predicted and actual values. Hence, we need to perform inverse normalization on the EDNR output and subsequently calculate the error between the predicted and actual values.

3.2. Parameter Settings

For EDNR to achieve outstanding performance, we must initially determine and evaluate the most appropriate parameter combinations for CNDE. In Section 2.2.2, we generated a complex network model with an initial node value of 3, as illustrated in Figure 3. Consequently, the initial node m_0 is a critical parameter that directly influences the topology of the final network model. Therefore, to obtain the optimal m_0 value while ensuring that the other parameters remain the same, we conducted two sets of experiments on heating and cooling loads. Each set consisted of 20 independent experiments, and the experimental results are summarized in Table 1. The optimal values of m_0 are 5 and 6 for dealing with heating and cooling load problems, respectively.

For our proposed CNDE optimization algorithm, in addition to the initialization parameter m_0 used in constructing complex network models, the population size Ps is crucial in determining computational accuracy. Thus, it is essential to determine an appropriate Ps value that balances both computational cost and optimization performance. In general, larger Ps values tend to result in better optimization performance, albeit at the expense of higher computational costs and slower convergence rates. Conversely, reducing Ps can decrease the optimization ability. According to the prediction task and considering both the computational cost and the fairness of the algorithmic competition, we set the Ps values of all population-based optimization algorithms to 50, with the maximum number of function evaluations being $10^5 * D$ in our experiments. For EDNR, the number k of the synaptic layer, M of the branch number, and the threshold θ_{soma} of the cell body layer are crucial customized parameters. To identify enhanced parameter combinations, we conducted 16 sets of experiments employing the $L_{16}(4^3)$ orthogonal array method and established two distinct parameter combinations based on different prediction targets, and the results are shown in Table 2. Additionally, Table 3 lists the significant parameter settings of all algorithms and models involved in the experiments, which were derived from relevant literature.

Table 1. Prediction results for energy consumption based on EDNR processing at different m_0 values.

Heating load	$m_0 = 3$	$m_0 = 4$	$m_0 = 5$	$m_0 = 6$	$m_0 = 7$	$m_0 = 8$
MSE	8.229	5.004	3.794	4.873	5.980	6.116
Cooling load	$m_0 = 3$	$m_0 = 4$	$m_0 = 5$	$m_0 = 6$	$m_0 = 7$	$m_0 = 8$
MSE	9.451	7.502	3.557	1.621	4.983	5.754

Table 2. EDNR prediction results (RMSE) for heating and cooling loads with different parameter combinations.

No.	k	θ_{soma}	M	Heating Load	Cooling Load
1	3	0.1	3	4.841	6.101
2	3	0.3	5	4.295	4.229
3	3	0.6	7	3.510	5.913
4	3	0.9	10	2.029	3.991

Table 2. Cont.

No.	k	θ_{soma}	M	Heating Load	Cooling Load
5	6	0.1	5	6.174	5.644
6	6	0.3	3	3.916	6.973
7	6	0.6	10	2.811	3.814
8	6	0.9	7	1.607	6.118
9	9	0.1	7	7.221	7.007
10	9	0.3	10	5.177	2.951
11	9	0.6	3	3.009	1.307
12	9	0.9	5	1.993	2.374
13	12	0.1	10	4.224	7.210
14	12	0.3	7	2.099	5.221
15	12	0.6	5	5.287	6.791
16	12	0.9	3	4.667	3.222

Table 3. Parameter settings for the learning algorithms and machine learning models.

Algorithms/Models	Parameters
GA [34]	$Var = 0.1, Cop = 0.3$
CS [26]	$\alpha = 0.01, P_a = 0.25$
FA [45]	$\alpha = 0.2, \gamma = 1, \beta_0 = 1$
GSA [28]	$\alpha = 20, G_0 = 100$
PSO [46]	$w \in [0.4, 0.9], c = 2$
SMS [23]	$\alpha \in [0.8, 0.2, 0], \gamma \in [0.8, 0.4, 0.1]$
DE [47]	$\beta \in [1.0, 0.6, 0.1], H \in [0.9, 0.2, 0]$
JADE [48]	$Cr = 0.9, F = 0.7$
ENN [49]	$Cr = 0.5, F = 0.5$
MLP [50]	$Learningrate = 0.01$
SVRs [51]	$Hiddenlayer = 10, learningrate = 0.01$
DT [52]	$Cost(c) = 0.5, g = 0.2$
BP [37]	$Minleaf = 25$
EDNR	$Learningrate = 0.05$
	$k = 6/9, \theta_{soma} = 0.9/0.6, M = 7/3$

3.3. Evaluation Metrics

The accuracy pertains to the level of resemblance between the predicted and actual values. To demonstrate the dissimilarities in performance between the different methods, we selected four frequently utilized evaluation metrics:

- The mean absolute error (MAE) is the average error between the predicted and actual values without considering outliers and can be expressed using the following formula:

$$MAE = \frac{1}{n} \sum_{i=1}^n |T_i - O_i|. \quad (20)$$

- The mean absolute percentage error (MAPE) is a relative measure that is sensitive to errors. Its formula is as follows:

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{T_i - O_i}{T_i} \right|. \quad (21)$$

- The mean squared error (MSE) represents the expected value of the square of the difference between the predicted and actual values. This indicator reflects the accuracy of the prediction method and its sensitivity to errors. The smaller the MSE is, the better the prediction accuracy. The calculation method is as follows:

$$MSE = \frac{1}{n} \sum_{i=1}^n (O_i - T_i)^2. \quad (22)$$

- The root-mean-squared error (RMSE) is the square root of the MSE, and it is calculated arithmetically so that its magnitude aligns with that of the original indicator. The definition is as follows:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (O_i - T_i)^2}. \quad (23)$$

where T_i and O_i represent the actual and predicted values, respectively, and n is the sample size. Notably, all the error results were derived from the test datasets, and the experimental training and test sets were randomly divided.

3.4. Performance Comparison

In this section, we present detailed experimental results and comparative analyses. The experimental results are categorized into two parts: prediction results, which were obtained via various optimization algorithms that were employed to train the DNR models, and EDNR results, which were obtained via machine learning models. For the fairness of the experiments, both the experimental training and test sets were randomly selected at a ratio of 7:3. Additionally, each experiment was independently performed 30 times.

3.4.1. Comparison with Different Learning Algorithms

In this study, we present CNDE, a complex-network-model-based DE algorithm. To verify the optimization capabilities of CNDE, we conducted comparative experiments with eight population-based heuristic algorithms: GA, CS, FA, GSA, PSO, SMS, DE, and JADE. To compare each algorithm at its optimal performance, the optimal parameter combination for each algorithm participating in the experiment was selected according to the pertinent literature. The nine aforementioned algorithms were applied as learning algorithms to train the DNR models, and the resulting models were subsequently utilized to predict the EE of buildings. The results of the experiment are displayed in Tables 4 and 5. We highlight the performance of the best performing method in terms of the four evaluation indicators. Tables 4 and 5 illustrate that the CNDE algorithm outperformed the others in forecasting both heating and cooling loads, indicating its superior optimization performance in model training; hence, it effectively aids EDNR in overcoming the issue of local optimization during the search process. In addition, it is essential to consider the convergence speed of the model as a crucial performance indicator of the method. Figure 5 shows the error convergence curves of all the models. According to Figure 5, the EDNR model that was obtained via CNDE training exhibited the fastest convergence and the lowest error rate, irrespective of the predicted heating or cooling load. This demonstrates the excellent performance of CNDE in addressing such problems.

To enhance the clarity of the experimental results, error bar charts that display both the result details and interexperiment variability are presented. Furthermore, each algorithm's stability is demonstrated through the use of a box-and-whiskers graph. Figure 6 illustrates the results in detail, while Figure 7 depicts the MAE of each method following the 30 experiments. The dots in the figure correspond to the results of a single experiment. As shown in Figures 6 and 7, CNDE, an EDNR learning algorithm, exhibited both reduced error and enhanced stability. Moreover, from Tables 4 and 5 and Figures 6 and 7, it can be observed that SMS and JADE also obtained superior performance, indicating their high generalization ability for addressing common optimization problems.

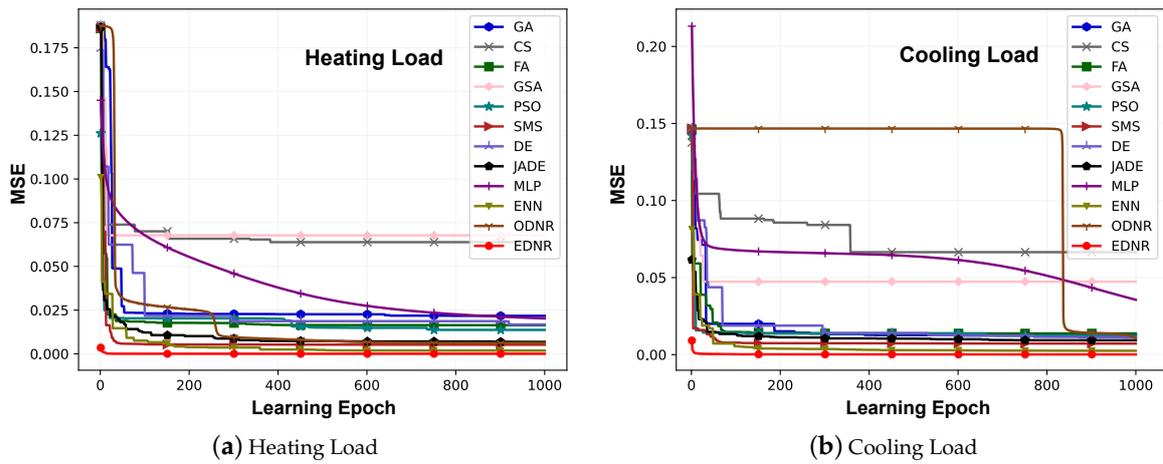


Figure 5. Convergence curves of the heating and cooling loads predicted by all the models.

Table 4. Heating load prediction results of DNR after training using different optimization algorithms.

	MAE Mean ± Std	MAPE Mean ± Std	MSE Mean ± Std	RMSE Mean ± Std
GA	$4.84 \times 10^0 \pm 9.75 \times 10^{-1}$ (↓)	$2.11 \times 10^{-1} \pm 5.63 \times 10^{-2}$ (↓)	$3.18 \times 10^1 \pm 9.44 \times 10^0$ (↓)	$5.58 \times 10^0 \pm 8.33 \times 10^{-1}$ (↓)
CS	$6.16 \times 10^0 \pm 1.07 \times 10^0$ (↓)	$2.82 \times 10^{-1} \pm 5.31 \times 10^{-2}$ (↓)	$5.09 \times 10^1 \pm 1.57 \times 10^1$ (↓)	$7.06 \times 10^0 \pm 1.10 \times 10^0$ (↓)
FA	$4.72 \times 10^0 \pm 1.05 \times 10^0$ (↓)	$2.08 \times 10^{-1} \pm 6.61 \times 10^{-2}$ (↓)	$3.07 \times 10^1 \pm 9.32 \times 10^0$ (↓)	$5.47 \times 10^0 \pm 9.16 \times 10^{-1}$ (↓)
GSA	$7.63 \times 10^0 \pm 3.09 \times 10^0$ (↓)	$3.27 \times 10^{-1} \pm 9.62 \times 10^{-2}$ (↓)	$9.12 \times 10^1 \pm 9.92 \times 10^1$ (↓)	$8.82 \times 10^0 \pm 3.72 \times 10^0$ (↓)
PSO	$3.01 \times 10^0 \pm 9.71 \times 10^{-1}$ (↓)	$1.22 \times 10^{-1} \pm 4.25 \times 10^{-2}$ (↓)	$1.41 \times 10^1 \pm 8.78 \times 10^0$ (↓)	$3.62 \times 10^0 \pm 1.01 \times 10^0$ (↓)
SMS	$2.40 \times 10^0 \pm 2.95 \times 10^{-1}$ (↓)	$1.01 \times 10^{-1} \pm 1.17 \times 10^{-2}$ (↓)	$8.57 \times 10^0 \pm 9.39 \times 10^{-1}$ (↓)	$2.92 \times 10^0 \pm 1.64 \times 10^{-1}$ (↓)
DE	$4.78 \times 10^0 \pm 9.44 \times 10^{-1}$ (↓)	$2.03 \times 10^{-1} \pm 5.62 \times 10^{-2}$ (↓)	$3.31 \times 10^1 \pm 8.54 \times 10^0$ (↓)	$5.71 \times 10^0 \pm 7.36 \times 10^{-1}$ (↓)
JADE	$2.91 \times 10^0 \pm 5.17 \times 10^{-1}$ (↓)	$1.20 \times 10^{-1} \pm 2.00 \times 10^{-2}$ (↓)	$1.18 \times 10^1 \pm 3.81 \times 10^0$ (↓)	$3.39 \times 10^0 \pm 5.35 \times 10^{-1}$ (↓)
CNDE	$1.26 \times 10^0 \pm 1.06 \times 10^0$	$5.70 \times 10^{-2} \pm 5.31 \times 10^{-2}$	$3.81 \times 10^0 \pm 7.07 \times 10^0$	$1.54 \times 10^0 \pm 1.22 \times 10^0$

Table 5. Cooling load prediction results of DNR after training using different optimization algorithms.

	MAE Mean ± Std	MAPE Mean ± Std	MSE Mean ± Std	RMSE Mean ± Std
GA	$3.71 \times 10^0 \pm 6.01 \times 10^{-1}$ (↓)	$1.38 \times 10^{-1} \pm 2.98 \times 10^{-2}$ (↓)	$2.16 \times 10^1 \pm 5.09 \times 10^0$ (↓)	$4.61 \times 10^0 \pm 5.61 \times 10^{-1}$ (↓)
CS	$5.43 \times 10^0 \pm 1.75 \times 10^0$ (↓)	$2.03 \times 10^{-1} \pm 5.39 \times 10^{-2}$ (↓)	$4.86 \times 10^1 \pm 3.40 \times 10^1$ (↓)	$6.63 \times 10^0 \pm 2.19 \times 10^0$ (↓)
FA	$3.90 \times 10^0 \pm 4.32 \times 10^{-1}$ (↓)	$1.54 \times 10^{-1} \pm 2.38 \times 10^{-2}$ (↓)	$2.17 \times 10^1 \pm 3.29 \times 10^0$ (↓)	$4.64 \times 10^0 \pm 3.58 \times 10^{-1}$ (↓)
GSA	$9.11 \times 10^0 \pm 3.18 \times 10^0$ (↓)	$3.08 \times 10^{-1} \pm 9.06 \times 10^{-2}$ (↓)	$1.45 \times 10^2 \pm 9.27 \times 10^1$ (↓)	$1.14 \times 10^1 \pm 3.91 \times 10^0$ (↓)
PSO	$2.98 \times 10^0 \pm 5.86 \times 10^{-1}$ (↓)	$1.10 \times 10^{-1} \pm 2.49 \times 10^{-2}$ (↓)	$1.50 \times 10^1 \pm 4.33 \times 10^0$ (↓)	$3.84 \times 10^0 \pm 5.52 \times 10^{-1}$ (↓)
SMS	$2.31 \times 10^0 \pm 2.41 \times 10^{-2}$ (↓)	$8.67 \times 10^{-2} \pm 1.36 \times 10^{-3}$ (↓)	$9.75 \times 10^0 \pm 1.20 \times 10^{-1}$ (↓)	$3.12 \times 10^0 \pm 1.92 \times 10^{-2}$ (↓)
DE	$4.04 \times 10^0 \pm 5.71 \times 10^{-1}$ (↓)	$1.54 \times 10^{-1} \pm 2.56 \times 10^{-2}$ (↓)	$2.43 \times 10^1 \pm 5.20 \times 10^0$ (↓)	$4.90 \times 10^0 \pm 5.37 \times 10^{-1}$ (↓)
JADE	$2.50 \times 10^0 \pm 3.12 \times 10^{-1}$ (↓)	$9.18 \times 10^{-2} \pm 1.07 \times 10^{-2}$ (↓)	$1.12 \times 10^1 \pm 2.32 \times 10^0$ (↓)	$3.33 \times 10^0 \pm 3.31 \times 10^{-1}$ (↓)
CNDE	$9.82 \times 10^{-1} \pm 1.57 \times 10^{-1}$	$3.85 \times 10^{-2} \pm 6.91 \times 10^{-3}$	$1.65 \times 10^0 \pm 4.79 \times 10^{-1}$	$1.27 \times 10^0 \pm 1.83 \times 10^{-1}$

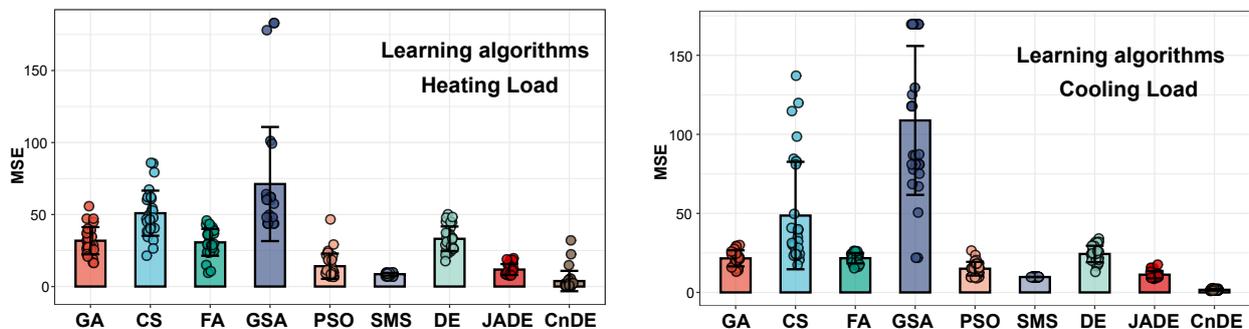


Figure 6. Error bars of the results of 30 experiments after DNR training using different optimization algorithms.

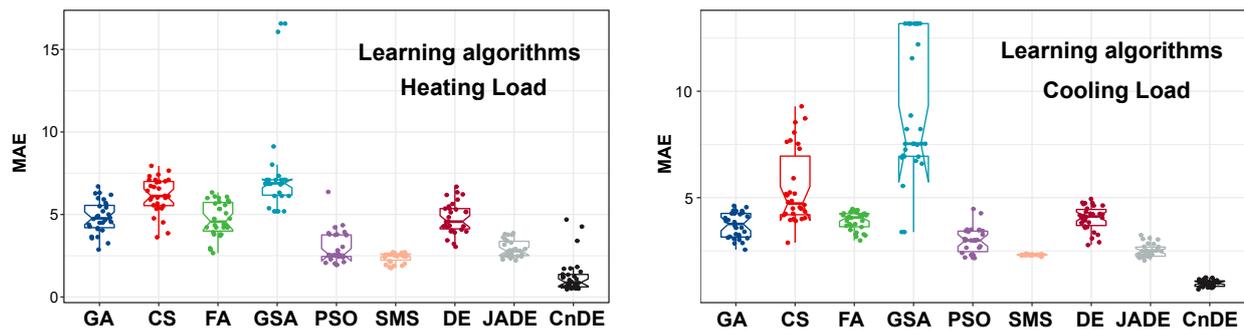


Figure 7. Box-and-whiskers plots of the results of 30 experiments after DNR training using different algorithms.

3.4.2. Comparison with Other Machine Learning Models

The preceding sections examined and confirmed the effectiveness and consistency of the CNDE algorithm. To further demonstrate the superior performance of EDNR, in addition to comparing intelligent algorithms as DNR training algorithms internally, we selected eight prevalent machine learning models as competitors for EDNR. These models include the Elman neural network (ENN), multilayer perceptron (MLP), four different kernel function support vector regressions, SVR(L), SVR(P), SVR(R), and SVR(S), DT models, and the original DNR (ODNR) model, which is trained using the conventional BP algorithm. Table 3 summarizes all the model parameters referenced in the relevant literature. Tables 6 and 7 present the experimental results, which show that EDNR achieved the best performance in both heating and cooling EE prediction. Additionally, the DT model demonstrated robust performance in predicting the heating load, with evaluation metrics similar to those of EDNR. This suggests that DTs can accurately capture the nonlinear relationships between the feature input and output for heating load prediction. Nonetheless, as shown in Figure 8, the DT model exhibited erratic performance when dealing with cooling loads.

It is obvious from Figures 8 and 9 that both SVR(S) and ODNR exhibited poor performance. On the one hand, ODNR is optimized by the BP algorithm, which employs a gradient-based mechanism and is easily affected by the initial point during the search process, eventually leading to local optimization. Consequently, ODNR fails to achieve optimal performance. On the other hand, SVR(S) employs the sigmoid function as its kernel function, thereby rendering it equivalent to a multilayer perceptron neural network. Although this function automatically determines the weights of the hidden layer nodes with respect to the input nodes during the training process, it may increase the complexity of the model, and it may be challenging to understand the balance between model overfitting and underfitting. However, SVR(R) with only one parameter performed better, indicating that SVR(R) is more effective at handling tasks with diverse decision boundaries. During the optimization process of CNDE, the complex node ordering operation ensures population diversity and balances exploration and exploitation. This results in a better qualified EDNR model for EE prediction after training. In summary, numerous experimental results demonstrate that our proposed EDNR model exhibited robust competitiveness in predicting building EE and has achieved satisfactory outcomes.

3.4.3. Computational Complexity

Computational complexity is a crucial measure for evaluating algorithm effectiveness, providing a qualitative description of an algorithm's runtime. Figure 10 depicts the independent runtimes for each model. On the one hand, the ENN required the most time to execute both heating and cooling load predictions as a result of its specific algorithmic structure. On the other hand, both SVR and DT required small amounts of time, owing to their typical machine learning models, and efficiently obtained the globally optimal solution while dealing with convex optimization problems, avoiding the curse of dimensionality. From

Figure 10, it can be observed that the DNR model trained by the evolutionary algorithm had a similar time cost, while the ODNR model trained by the BP algorithm took a short time. The heuristic algorithm requires a slightly more intricate computational process and, thus, consumed more time. However, the BP algorithm relies on gradient information and is susceptible to falling into local optima, resulting in premature convergence of the algorithm. Upon closer inspection, it can be found that the EDNR model trained by CNDE required slightly less time than the other evolutionary algorithms. Notably, as shown in Figure 5, EDNR can converge in relatively few iterations, which means that EDNR actually does not take much time to train; in other words, the unique network-node-guided search mechanism enables it to converge rapidly, resulting in a shorter running time.

Table 6. Experimental results of different models in heating load prediction.

	MAE Mean ± Std	MAPE Mean ± Std	MSE Mean ± Std	RMSE Mean ± Std
ENN	$2.45 \times 10^0 \pm 9.82 \times 10^{-2}$ (↓)	$1.02 \times 10^{-1} \pm 4.62 \times 10^{-3}$ (↓)	$8.90 \times 10^0 \pm 4.54 \times 10^{-1}$ (↓)	$2.96 \times 10^0 \pm 6.99 \times 10^{-2}$ (↓)
MLP	$4.19 \times 10^0 \pm 1.11 \times 10^0$ (↓)	$1.60 \times 10^{-1} \pm 4.62 \times 10^{-2}$ (↓)	$3.28 \times 10^1 \pm 1.64 \times 10^1$ (↓)	$5.57 \times 10^0 \pm 1.37 \times 10^0$ (↓)
SVR(L)	$3.03 \times 10^0 \pm 1.36 \times 10^{-15}$ (↓)	$1.46 \times 10^{-1} \pm 5.65 \times 10^{-17}$ (↓)	$1.28 \times 10^1 \pm 3.61 \times 10^{-15}$ (↓)	$3.58 \times 10^0 \pm 4.52 \times 10^{-16}$ (↓)
SVR(P)	$2.32 \times 10^0 \pm 1.36 \times 10^{-15}$ (↓)	$9.71 \times 10^{-2} \pm 1.41 \times 10^{-17}$ (↓)	$9.18 \times 10^0 \pm 5.42 \times 10^{-15}$ (↓)	$3.03 \times 10^0 \pm 0.00 \times 10^0$ (↓)
SVR(R)	$2.44 \times 10^0 \pm 4.52 \times 10^{-16}$ (↓)	$9.74 \times 10^{-2} \pm 2.82 \times 10^{-17}$ (↓)	$8.65 \times 10^0 \pm 3.61 \times 10^{-15}$ (↓)	$2.94 \times 10^0 \pm 0.00 \times 10^0$ (↓)
SVR(S)	$1.14 \times 10^1 \pm 0.00 \times 10^0$ (↓)	$5.00 \times 10^{-1} \pm 1.13 \times 10^{-16}$ (↓)	$2.07 \times 10^2 \pm 2.89 \times 10^{-14}$ (↓)	$1.44 \times 10^1 \pm 9.03 \times 10^{-15}$ (↓)
DT	$1.64 \times 10^0 \pm 0.00 \times 10^0$ (↓)	$6.28 \times 10^{-2} \pm 0.00 \times 10^0$ (↓)	$4.66 \times 10^0 \pm 9.03 \times 10^{-16}$ (↓)	$2.16 \times 10^0 \pm 1.81 \times 10^{-15}$ (↓)
ODNR	$1.47 \times 10^1 \pm 4.77 \times 10^0$ (↓)	$5.24 \times 10^{-1} \pm 1.64 \times 10^{-1}$ (↓)	$3.34 \times 10^2 \pm 1.28 \times 10^2$ (↓)	$1.74 \times 10^1 \pm 5.59 \times 10^0$ (↓)
EDNR	$1.26 \times 10^0 \pm 1.06 \times 10^0$	$5.70 \times 10^{-2} \pm 5.31 \times 10^{-2}$	$3.81 \times 10^0 \pm 7.07 \times 10^0$	$1.54 \times 10^0 \pm 1.22 \times 10^0$

Table 7. Experimental results of different models in cooling load prediction.

	MAE Mean ± Std	MAPE Mean ± Std	MSE Mean ± Std	RMSE Mean ± Std
ENN	$2.23 \times 10^0 \pm 8.19 \times 10^{-2}$ (↓)	$8.27 \times 10^{-2} \pm 3.60 \times 10^{-3}$ (↓)	$9.41 \times 10^0 \pm 4.11 \times 10^{-1}$ (↓)	$3.07 \times 10^0 \pm 6.67 \times 10^{-2}$ (↓)
MLP	$3.89 \times 10^0 \pm 9.68 \times 10^{-1}$ (↓)	$1.39 \times 10^{-1} \pm 4.28 \times 10^{-2}$ (↓)	$2.95 \times 10^1 \pm 1.26 \times 10^1$ (↓)	$5.32 \times 10^0 \pm 1.11 \times 10^0$ (↓)
SVR(L)	$2.25 \times 10^0 \pm 9.03 \times 10^{-16}$ (↓)	$8.69 \times 10^{-2} \pm 2.82 \times 10^{-17}$ (↓)	$1.00 \times 10^1 \pm 1.81 \times 10^{-15}$ (↓)	$3.17 \times 10^0 \pm 2.26 \times 10^{-15}$ (↓)
SVR(P)	$2.51 \times 10^0 \pm 9.03 \times 10^{-16}$ (↓)	$9.53 \times 10^{-2} \pm 2.82 \times 10^{-17}$ (↓)	$1.05 \times 10^1 \pm 5.42 \times 10^{-15}$ (↓)	$3.24 \times 10^0 \pm 2.26 \times 10^{-15}$ (↓)
SVR(R)	$2.14 \times 10^0 \pm 1.36 \times 10^{-15}$ (↓)	$7.81 \times 10^{-2} \pm 4.23 \times 10^{-17}$ (↓)	$9.06 \times 10^0 \pm 0.00 \times 10^0$ (↓)	$3.01 \times 10^0 \pm 4.52 \times 10^{-16}$ (↓)
SVR(S)	$1.11 \times 10^1 \pm 1.81 \times 10^{-15}$ (↓)	$4.44 \times 10^{-1} \pm 1.13 \times 10^{-16}$ (↓)	$1.97 \times 10^2 \pm 5.78 \times 10^{-14}$ (↓)	$1.40 \times 10^1 \pm 5.42 \times 10^{-15}$ (↓)
DT	$1.66 \times 10^0 \pm 2.26 \times 10^{-16}$ (↓)	$5.95 \times 10^{-2} \pm 1.41 \times 10^{-17}$ (↓)	$4.75 \times 10^0 \pm 9.03 \times 10^{-16}$ (↓)	$2.18 \times 10^0 \pm 1.36 \times 10^{-15}$ (↓)
ODNR	$1.10 \times 10^1 \pm 4.37 \times 10^0$ (↓)	$3.52 \times 10^{-1} \pm 1.34 \times 10^{-1}$ (↓)	$2.18 \times 10^2 \pm 1.05 \times 10^2$ (↓)	$1.38 \times 10^1 \pm 5.33 \times 10^0$ (↓)
EDNR	$9.82 \times 10^{-1} \pm 1.57 \times 10^{-1}$	$3.85 \times 10^{-2} \pm 6.91 \times 10^{-3}$	$1.65 \times 10^0 \pm 4.79 \times 10^{-1}$	$1.27 \times 10^0 \pm 1.83 \times 10^{-1}$

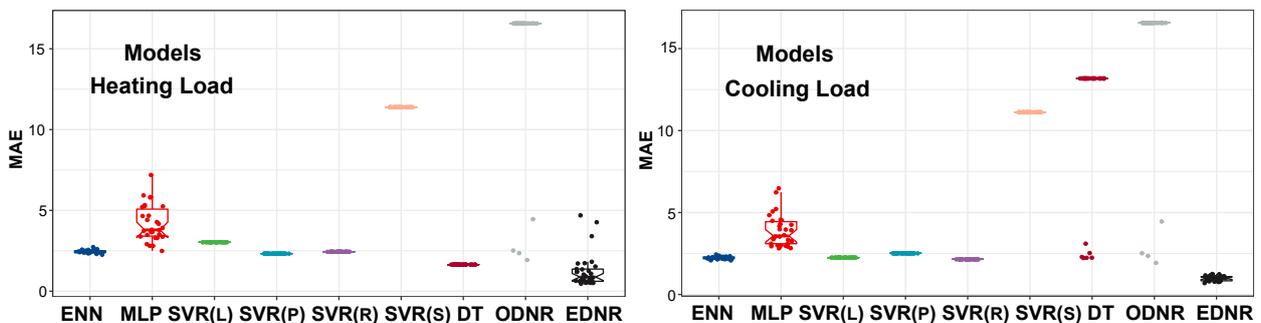


Figure 8. Box-and-whiskers plots of the results of 30 experiments with different models.

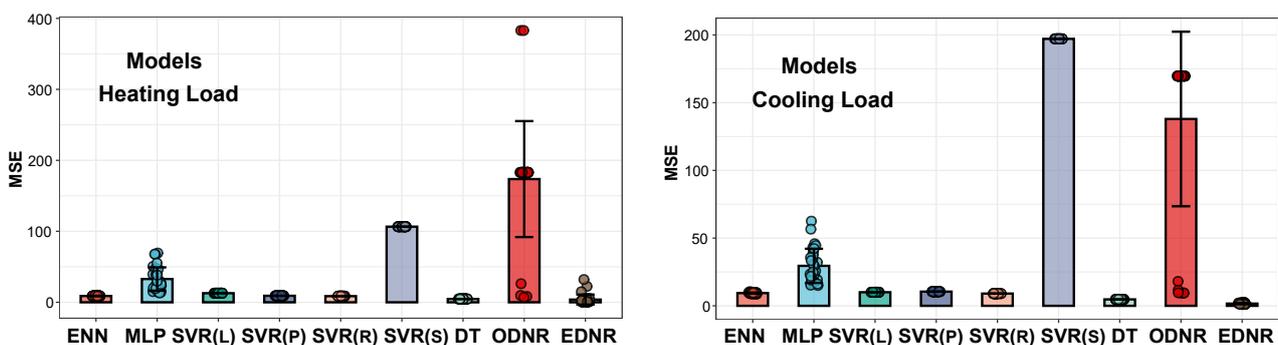


Figure 9. Error bars of the results of 30 experiments with machine learning models.

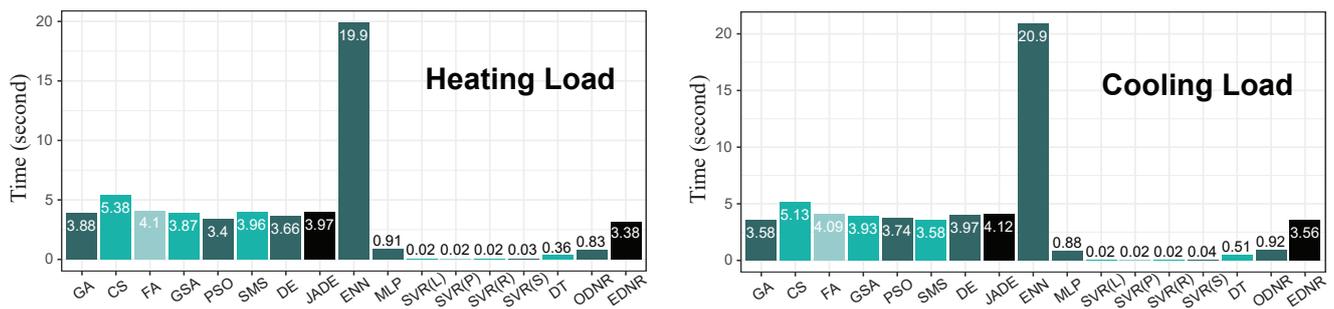


Figure 10. Time consumptions of all models for predicting heating and cooling loads (s).

3.5. Statistical Analysis

To comprehensively evaluate the performance of EDNR in predicting building EE, we used two statistical tests to analyze the results of 30 independent experiments: the Friedman test and the Wilcoxon rank-sum test [53,54]. The significance level p of the Wilcoxon rank-sum test was set at 0.05. $p < 0.05$ indicates that EDNR significantly outperformed the competitor. The results of the Friedman test and Wilcoxon rank sum test are summarized in Table 8.

In Table 8, the symbols \uparrow , \downarrow , and \circ indicate the relative strengths and weaknesses of the method compared to EDNR. \downarrow shows a significant difference, meaning that EDNR had significant advantages compared with the other methods; \circ represents no significant difference between the two methods; \uparrow shows that EDNR performed less effectively than the other methods. $w/t/l$ is the number of $\downarrow/\circ/\uparrow$ in each corresponding method. According to the rankings presented in Table 8, there is minimal discrepancy in the ranking order between the two sets of techniques for both the heating and cooling load prediction tasks. Notably, EDNR obtained the lowest scores, i.e., the highest rankings, for both types of problems, indicating that EDNR not only demonstrates high accuracy in predicting EE, but also shows model stability. In addition, the $w/t/l$ values of EDNR in predicting the EE were 15/1/0 and 16/0/0, respectively, which fully demonstrates the superior performance of EDNR in addressing building EE forecasting. In fact, these results can be inferred from previous data tables or visualization images. In summary, the two nonparametric statistical analysis methods validate the effectiveness and comprehensiveness of EDNR in assessing heating and cooling loads in buildings.

Table 8. Statistical test results of all competitive methods for energy efficiency forecasting.

Models	EDNR(CNDE)	GA	CS	FA	GSA	PSO	SMS	DE	JADE	
Heating load	Score	4.5661	9.0057	10.1014	8.9091	12.2119	8.1216	5.4071	9.4005	7.5515
	Ranking	1	11	14	10	15	9	3	13	7
	p -value	-	0 (\downarrow)	0 (\downarrow)	0 (\downarrow)	0 (\downarrow)	0 (\downarrow)	0.0320 (\downarrow)	0 (\downarrow)	0 (\downarrow)
	$w/t/l$	15/1/0	ENN	MLP	SVR(L)	SVR(P)	SVR(R)	SVR(S)	DT	ODNR(BP)
	Score	6.0111	9.3310	7.8082	6.5651	5.6670	13.1127	4.8025	14.0512	
Cooling load	Score	3.0755	6.0029	7.7072	6.1910	9.2001	5.7708	4.9814	6.3055	5.3011
	Ranking	1	10	14	11	15	9	5	12	8
	p -value	-	0 (\downarrow)	0 (\downarrow)	0 (\downarrow)	0 (\downarrow)	0 (\downarrow)	0.0081 (\downarrow)	0 (\downarrow)	0 (\downarrow)
	$w/t/l$	16/0/0	ENN	MLP	SVR(L)	SVR(P)	SVR(R)	SVR(S)	DT	ODNR(BP)
	Score	4.8122	6.7776	5.0291	5.1148	4.7002	9.7118	3.9221	10.0812	
Ranking	4	13	6	7	3	16	2	17		
p -value	-	0.0097 (\downarrow)	0 (\downarrow)	0.0008 (\downarrow)	0 (\downarrow)	0.0122 (\downarrow)	0 (\downarrow)	0.0339 (\downarrow)	0 (\downarrow)	

4. Conclusions

Enhancing the ability to monitor and forecast EE in the building sector holds promise for optimizing energy scheduling and improving building efficiency. This paper introduces

a novel model, EDNR, tailored for predicting residential building EE. To bolster the model's predictive ability, we proposed a new DE algorithm guided by complex network principles to optimize the parameter space of EDNR. The guided search strategy ensures both individual diversity and enhances the algorithm's search efficacy. The results obtained by employing various optimization algorithms for DNR model training highlight the significant optimization capability of CNDE. Comparative experiments with eight machine learning models demonstrated that EDNR exhibits superior predictive performance and stability. Consequently, EDNR has emerged as a viable and competitive approach for EE prediction.

Although the CNDE algorithm boasts powerful search capabilities, integrating the complex-network-guided search strategy with current mainstream DE variants such as jSO [55] and OLSHADE-CS [56] presents challenges. This is due primarily to the difficulty in maintaining the scale-free properties of the network model when reducing the population size, which is analogous to reducing the number of nodes in a complex network. Consequently, the population size of the optimization algorithm compatible with the proposed guidance mechanism must remain relatively stable. Furthermore, EDNR grapples with the curse of dimensionality, where the parameter space expands exponentially when tackling high-dimensional problems, resulting in a sluggish learning process. In our future research, we aim to address these challenges by developing adaptive methods for customizing the parameters of EDNR. Additionally, we plan to explore combining EDNR with feature selection or feature extraction techniques to mitigate the curse of dimensionality when dealing with high-dimensional problems. Moreover, efforts will be made to enhance the generalizability of EDNR across various domains.

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

Funding: This study was partially supported by the Qinglan Project of Jiangsu Universities, the Talent Development Project of Taizhou University (No. TZXY2018QDJJ006), the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20220619), the Young Science and Technology Talent Support Project of Taizhou, and the National Natural Science Foundation of China (Grant No. 62203069).

Informed Consent Statement: Informed consent was obtained from all the subjects involved in the study.

Data Availability Statement: All data are contained within the article.

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

References

1. Pérez-Lombard, L.; Ortiz, J.; Pout, C. A review on buildings energy consumption information. *Energy Build.* **2008**, *40*, 394–398. [[CrossRef](#)]
2. Olu-Ajayi, R.; Alaka, H.; Sulaimon, I.; Balogun, H.; Wusu, G.; Yusuf, W.; Adegoke, M. Building energy performance prediction: A reliability analysis and evaluation of feature selection methods. *Expert Syst. Appl.* **2023**, *225*, 120109. [[CrossRef](#)]
3. Santamouris, M.; Papanikolaou, N.; Livada, I.; Koronakis, I.; Georgakis, C.; Argiriou, A.; Assimakopoulos, D. On the impact of urban climate on the energy consumption of buildings. *Solar Energy* **2001**, *70*, 201–216. [[CrossRef](#)]
4. Deb, C.; Zhang, F.; Yang, J.; Lee, S.E.; Shah, K.W. A review on time series forecasting techniques for building energy consumption. *Renew. Sustain. Energy Rev.* **2017**, *74*, 902–924. [[CrossRef](#)]
5. Zhang, W.; Wu, Y.; Calautit, J.K. A review on occupancy prediction through machine learning for enhancing energy efficiency, air quality and thermal comfort in the built environment. *Renew. Sustain. Energy Rev.* **2022**, *167*, 112704. [[CrossRef](#)]
6. Shabunko, V.; Lim, C.; Mathew, S. EnergyPlus models for the benchmarking of residential buildings in Brunei Darussalam. *Energy Build.* **2018**, *169*, 507–516. [[CrossRef](#)]
7. Asadi, S.; Amiri, S.S.; Mottahedi, M. On the development of multi-linear regression analysis to assess energy consumption in the early stages of building design. *Energy Build.* **2014**, *85*, 246–255. [[CrossRef](#)]

8. Zeng, A.; Ho, H.; Yu, Y. Prediction of building electricity usage using Gaussian Process Regression. *J. Build. Eng.* **2020**, *28*, 101054. [[CrossRef](#)]
9. Atalay, S.D.; Calis, G.; Kus, G.; Kuru, M. Performance analyses of statistical approaches for modeling electricity consumption of a commercial building in France. *Energy Build.* **2019**, *195*, 82–92. [[CrossRef](#)]
10. Sen, P.; Roy, M.; Pal, P. Application of ARIMA for forecasting energy consumption and GHG emission: A case study of an Indian pig iron manufacturing organization. *Energy* **2016**, *116*, 1031–1038. [[CrossRef](#)]
11. Li, K.; Hu, C.; Liu, G.; Xue, W. Building's electricity consumption prediction using optimized artificial neural networks and principal component analysis. *Energy Build.* **2015**, *108*, 106–113. [[CrossRef](#)]
12. Fan, C.; Sun, Y.; Xiao, F.; Ma, J.; Lee, D.; Wang, J.; Tseng, Y.C. Statistical investigations of transfer learning-based methodology for short-term building energy predictions. *Appl. Energy* **2020**, *262*, 114499. [[CrossRef](#)]
13. Lu, C.; Li, S.; Lu, Z. Building energy prediction using artificial neural networks: A literature survey. *Energy Build.* **2022**, *262*, 111718. [[CrossRef](#)]
14. Ma, Z.; Ye, C.; Li, H.; Ma, W. Applying support vector machines to predict building energy consumption in China. *Energy Procedia* **2018**, *152*, 780–786. [[CrossRef](#)]
15. Shao, M.; Wang, X.; Bu, Z.; Chen, X.; Wang, Y. Prediction of energy consumption in hotel buildings via support vector machines. *Sustain. Cities Soc.* **2020**, *57*, 102128. [[CrossRef](#)]
16. Wang, Z.; Wang, Y.; Zeng, R.; Srinivasan, R.S.; Ahrentzen, S. Random Forest based hourly building energy prediction. *Energy Build.* **2018**, *171*, 11–25. [[CrossRef](#)]
17. Zhou, X.; Ren, J.; An, J.; Yan, D.; Shi, X.; Jin, X. Predicting open-plan office window operating behavior using the random forest algorithm. *J. Build. Eng.* **2021**, *42*, 102514. [[CrossRef](#)]
18. Yu, Z.; Haghghat, F.; Fung, B.C.; Yoshino, H. A decision tree method for building energy demand modeling. *Energy Build.* **2010**, *42*, 1637–1646. [[CrossRef](#)]
19. Fu, G. Deep belief network based ensemble approach for cooling load forecasting of air-conditioning system. *Energy* **2018**, *148*, 269–282. [[CrossRef](#)]
20. Peng, J.; Kimmig, A.; Wang, J.; Liu, X.; Niu, Z.; Ovtcharova, J. Dual-stage attention-based long-short-term memory neural networks for energy demand prediction. *Energy Build.* **2021**, *249*, 111211. [[CrossRef](#)]
21. Ji, J.; Gao, S.; Cheng, J.; Tang, Z.; Todo, Y. An approximate logic neuron model with a dendritic structure. *Neurocomputing* **2016**, *173*, 1775–1783. [[CrossRef](#)]
22. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [[CrossRef](#)]
23. Cuevas, E.; Echavarría, A.; Ramírez-Ortegón, M.A. An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation. *Appl. Intell.* **2014**, *40*, 256–272. [[CrossRef](#)]
24. 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; Volume 4, pp. 1942–1948.
25. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [[CrossRef](#)]
26. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
27. Yang, X.S. Firefly algorithms for multimodal optimization. In Proceedings of the International Symposium on Stochastic Algorithms, Sapporo, Japan, 26–28 October 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 169–178.
28. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [[CrossRef](#)]
29. Storn, R.; Price, K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [[CrossRef](#)]
30. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [[CrossRef](#)]
31. Orosz, T.; Rassólkín, A.; Kallaste, A.; Arsénio, P.; Pánek, D.; Kaska, J.; Karban, P. Robust design optimization and emerging technologies for electrical machines: Challenges and open problems. *Appl. Sci.* **2020**, *10*, 6653. [[CrossRef](#)]
32. Tang, C.; Ji, J.; Tang, Y.; Gao, S.; Tang, Z.; Todo, Y. A novel machine learning technique for computer-aided diagnosis. *Eng. Appl. Artif. Intell.* **2020**, *92*, 103627. [[CrossRef](#)]
33. Sha, Z.; Hu, L.; Todo, Y.; Ji, J.; Gao, S.; Tang, Z. A breast cancer classifier using a neuron model with dendritic nonlinearity. *IEICE Trans. Inf. Syst.* **2015**, *98*, 1365–1376. [[CrossRef](#)]
34. Ji, J.; Song, S.; Tang, Y.; Gao, S.; Tang, Z.; Todo, Y. Approximate logic neuron model trained by states of matter search algorithm. *Knowl.-Based Syst.* **2019**, *163*, 120–130. [[CrossRef](#)]
35. Tang, Y.; Ji, J.; Zhu, Y.; Gao, S.; Tang, Z.; Todo, Y. A differential evolution-oriented pruning neural network model for bankruptcy prediction. *Complexity* **2019**, *2019*, 8682124. [[CrossRef](#)]
36. Ji, J.; Tang, Y.; Ma, L.; Li, J.; Lin, Q.; Tang, Z.; Todo, Y. Accuracy versus simplification in an approximate logic neural model. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 5194–5207. [[CrossRef](#)]
37. Zhou, T.; Gao, S.; Wang, J.; Chu, C.; Todo, Y.; Tang, Z. Financial time series prediction using a dendritic neuron model. *Knowl.-Based Syst.* **2016**, *105*, 214–224. [[CrossRef](#)]
38. Song, Z.; Tang, Y.; Ji, J.; Todo, Y. Evaluating a dendritic neuron model for wind speed forecasting. *Knowl.-Based Syst.* **2020**, *201*, 106052. [[CrossRef](#)]

39. Dong, M.; Tang, C.; Ji, J.; Lin, Q.; Wong, K.C. Transmission trend of the COVID-19 pandemic predicted by dendritic neural regression. *Appl. Soft Comput.* **2021**, *111*, 107683. [[CrossRef](#)] [[PubMed](#)]
40. Tang, Y.; Song, Z.; Zhu, Y.; Hou, M.; Tang, C.; Ji, J. Adopting a dendritic neural model for predicting stock price index movement. *Expert Syst. Appl.* **2022**, *205*, 117637. [[CrossRef](#)]
41. Ji, J.; Zhao, J.; Lin, Q.; Tan, K.C. Competitive Decomposition-Based Multiobjective Architecture Search for the Dendritic Neural Model. *IEEE Trans. Cybern.* **2022**, *53*, 6829–6842. [[CrossRef](#)]
42. Ji, J.; Tang, C.; Zhao, J.; Tang, Z.; Todo, Y. A survey on dendritic neuron model: Mechanisms, algorithms and practical applications. *Neurocomputing* **2022**, *489*, 390–406. [[CrossRef](#)]
43. Opara, K.R.; Arabas, J. Differential Evolution: A survey of theoretical analyses. *Swarm Evol. Comput.* **2019**, *44*, 546–558. [[CrossRef](#)]
44. Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A. Differential Evolution: A review of more than two decades of research. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103479.
45. Yang, X.S.; Slowik, A. Firefly algorithm. In *Swarm Intelligence Algorithms*; CRC Press: Boca Raton, FL, USA, 2020; pp. 163–174.
46. Xia, X.; Gui, L.; He, G.; Xie, C.; Wei, B.; Xing, Y.; Wu, R.; Tang, Y. A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm. *J. Comput. Sci.* **2018**, *26*, 488–500. [[CrossRef](#)]
47. Verma, P.; Sanyal, K.; Srinivsan, D.; Swarup, K. Information exchange based clustered differential evolution for constrained generation-transmission expansion planning. *Swarm Evol. Comput.* **2019**, *44*, 863–875. [[CrossRef](#)]
48. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [[CrossRef](#)]
49. Zhang, Y.; Wang, X.; Tang, H. An improved Elman neural network with piecewise weighted gradient for time series prediction. *Neurocomputing* **2019**, *359*, 199–208. [[CrossRef](#)]
50. Tang, J.; Deng, C.; Huang, G.B. Extreme learning machine for multilayer perceptron. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *27*, 809–821. [[CrossRef](#)] [[PubMed](#)]
51. Chang, C.C.; Lin, C.J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol. (TIST)* **2011**, *2*, 1–27. [[CrossRef](#)]
52. Song, Y.Y.; Ying, L. Decision tree methods: Applications for classification and prediction. *Shanghai Arch. Psychiatry* **2015**, *27*, 130.
53. García, S.; Fernández, A.; Luengo, J.; Herrera, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.* **2010**, *180*, 2044–2064. [[CrossRef](#)]
54. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [[CrossRef](#)]
55. Brest, J.; Maučec, M.S.; Bošković, B. Single objective real-parameter optimization: Algorithm jSO. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 1311–1318.
56. Kumar, A.; Biswas, P.P.; Suganthan, P.N. Differential evolution with orthogonal array-based initialization and a novel selection strategy. *Swarm Evol. Comput.* **2022**, *68*, 101010. [[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.