©SHUTTERSTOCK.COM/JIMMOYHT

Forecasting Wind Speed Time Series Via Dendritic Neural Regression

Junkai Ji, Minhui Dong, and Qiuzhen Lin
Shenzhen University, CHINA

Kay Chen Tan
The Hong Kong Polytechnic University, HONG KONG SAR.

Abstract—Wind energy is considered one of the fastest growing renewable ('green') energy resources. Precise wind power forecasting is imperative to ensure reliable power system planning and wind farm operation. However, traditional methods cannot yield satisfactory forecasts because of the chaotic properties and high volatility of wind speed time series. To address this issue, the use of artificial neural networks has attracted increasing attention owing to their significantly enhanced prediction accuracy. Based on these considerations, a novel neural model with a dynamic dendrite structure, known as the dendritic neuron model (DNM), can be adopted for wind speed time series prediction. The DNM is a plausible biological neural model that was originally designed for classification problems; accordingly, this study proposes the use of a regressive version of the DNM, named dendritic neural regression (DNR), in which the dendrite strength of each branch is considered. To enhance the prediction performance, the recently proposed states of matter search (SMS) optimization algorithm is used to optimize the neural architecture for DNR. By virtue of the powerful search ability of the SMS algorithm, DNR can

Digital Object Identifier 10.1109/MCI.2021.3084416
Date of current version: 15 July 2021

Corresponding Author: Qiuzhen Lin (e-mail: qiuzhlin@szu.edu.cn).

1556-603X/21©2021IEEE

efficiently capture the nonlinear correlations among distinct features and dendritic branches. Extensive experimental results and statistical tests demonstrate that compared with other state-of-the-art prediction techniques, DNR can achieve highly competitive results in wind speed forecasting.

I. Introduction

With the rapid development of society, it has become challenging for the remaining fossil fuel supply to meet societal requirements worldwide; thus, the demand for sustainable renewable energy is growing. In recent years, wind energy, as a sustainable renewable energy source, has attracted particular attention. Wind is an abundant, pollution-free energy source that is widely distributed, has few geographical restrictions, and can be easily used anywhere. Building wind farms is one of the main challenges in the use of wind energy. Nevertheless, the cost of building wind farms is low, no additional energy source is required, and such farms do not impact the surrounding environment.

Since the energy supplied by wind farms mainly depends on the wind power, precise wind power forecasting is imperative to enable reliable power system planning and wind farm operation [1, 2]. Wind power is closely related to wind speed, and these two parameters share several characteristics, such as randomness, uncontrollability and intermittency [3]. Because of these characteristics, wind farm management is extremely challenging. Calculating the energy production of a wind farm is essential for assessing the economic feasibility of such a project prior to construction planning [4]. Therefore, accurate wind speed prediction is being strongly prioritized in this context [5]. More accurate forecasting capabilities correspond to larger reductions in the construction costs of wind farms [6].

To date, various models have been used for wind speed prediction. These methods can be classified into three categories: physical models, statistical models, and artificial intelligence models [7]. Examples of physical models include the Mesoscale Model Version 5 [8] and the Weather Research and Forecasting Model [9]. These numerical prediction models can achieve satisfactory performance in long-term wind speed prediction; however, such models require complex atmospheric information pertaining to pressure, temperature and other environmental factors and exhibit high computational complexity [10], [11].

Compared to physical models, statistical models are more widely used to forecast wind speed. Examples of such models include direct random time series models [12], autoregressive models [13] and autoregressive integrated moving average (ARIMA) models [14]. ARIMA models are regarded as a typical class of statistical models, and their prediction performance in short-term wind speed forecasting has been verified [15]. However, in general, the prediction performance of statistical models is flawed [16] because most statistical models are based on the assumption that the wind speed series follows a normal distribution, although this assumption is not valid in all cases. Moreover, statistical models have linear correlation structures and always yield large errors when applied to intermittent and stochastic wind speed series [17].

The third category pertains to artificial intelligence models, which can effectively overcome the shortcomings of the abovementioned methods. Considering the nonlinear nature of wind speed series, artificial intelligence algorithms that are designed for effectively solving nonlinear problems, such as artificial neural networks (ANNs) [16] and support vector machines (SVMs) [18], are suitable for wind speed forecasting. A previous comparison of prediction performance has demonstrated that artificial intelligence algorithms are faster and more accurate than statistical models [19]. SVMs are commonly used in prediction frameworks, and they can outperform ANNs in certain cases. However, the performance of an SVM is limited by its penalty settings and kernel parameters; consequently, algorithms for tuning these hyperparameters are necessary [20]. For example, a genetic algorithm was employed to enhance the prediction results of an SVM in [21]; a reduced SVM with feature selection, trained using the particle swarm optimization algorithm, was used to optimize the parameters of an SVM in [22]; and the performance of an SVM was enhanced using the cuckoo search algorithm in [23]. In addition to SVMs, an increasing number of ANNs and their variants have been proposed for wind speed prediction. For instance, a backpropagation (BP) neural network was employed to forecast a wind speed series in [24], a combination of an ANN and Markov chains was proposed for forecasting in [25], and a functional network was utilized for multistep wind speed prediction in [26]. Furthermore, a fine-tuned long short-term memory (LSTM) neural network hybridized with the crow search algorithm, the wavelet transform and feature selection was applied for short-term wind speed forecasting in [27]. In [28], a hybrid model involving a causal convolutional network and a gated recurrent unit architecture was used in wind speed prediction.

In general, physical models and traditional statistical models both have several limitations pertaining to the precision and robustness of wind speed time series prediction, whereas artificial intelligence models can effectively overcome these problems to offer more powerful prediction performance. Based on these considerations, the dendritic neuron model (DNM), which was recently developed based on inspiration from biological neurons in vivo [29], is adopted in this study for the prediction of wind speed time series. In the DNM, synaptic nonlinearity is implemented in a dendritic structure to effectively solve linearly inseparable problems, and this model has been applied to a variety of complex continuous functions [30]–[32]. The original DNM was specifically designed for classification problems. By discarding the unnecessary synapses and dendritic branches in the DNM, diverse dendritic structures can be produced to pursue an extremely high classification speed for each task. Notably, however, the structure of the original DNM is extremely simple, and it

The demand for sustainable renewable energy is growing. Wind energy, as a sustainable renewable energy source, has attracted particular attention.

ignores the mechanism by which the signal transformation strengths of the dendritic branches vary with their thickness [33]. Therefore, this study proposes the use of a variant of the DNM called dendritic neural regression (DNR), in which the intensity of each dendritic signal is considered to enhance the regression ability.

The neural architecture considerably influences the model performance of ANNs [34]. Accordingly, evolutionary algorithms have been widely used to realize neural architecture search tasks for performance optimization. For instance, Sun et al. proposed an automatic method of designing convolutional neural network (CNN) architectures for solving image classification problems based on genetic algorithms [35]. Lu et al. used a multiobjective evolutionary algorithm for CNN architecture design [36]. In contrast to that of conventional ANNs, the neural architecture of a DNR model is determined by the values of the weights and thresholds rather than by hyperparameters. Subsequently, an inherent pruning mechanism can be implemented to simplify the DNR model to produce unique neural architectures for particular real-world tasks, as described in our previous research [29], [37]. To further enhance the performance of wind speed forecasting, the recently proposed states of matter search (SMS) algorithm [38] can be used to optimize the neural architecture of the DNR model. The SMS algorithm is a global search algorithm that can escape from local minima more effectively than gradient-based optimization algorithms. In fact, many researchers have attempted to employ evolutionary algorithms to enhance the performance of ANNs for chaotic time series prediction. For example, particle swarm optimization has been introduced into echo state networks as a pretraining tool to optimize the untrained weights to address time series forecasting problems [39]. Furthermore, a genetic algorithm has been introduced into the Elman neural network to optimize the connection weights and thresholds to prevent the optimization from becoming trapped in local minima and enhance the training speed and success rate [40]. Moreover, a modified cuckoo search algorithm has been used to optimize wavelet neural networks to achieve higher generalization capabilities in chaotic time series forecasting [41]. In addition, an evolving fuzzy neural network predictor has been proposed to effectively capture the dynamic properties of multidimensional datasets and accurately track the system characteristics [42], while in [43], a novel strategy was proposed for evolving the structure of deep recurrent neural networks by means of ant colony optimization.

Compared with other ANNs, a DNR model can help enhance the performance of wind speed prediction because its plastic neural architecture can efficiently capture the nonlinear correlations among distinct features and different dendritic branches. In addition, a one-dimensional wind speed time series typically appears to exhibit intermittent and random features since it contains complex information projected from a higher-dimensional space. Thus, the phase space should be reconstructed based on Takens' theorem [44]. The time delays and numbers of embedding dimensions for a wind speed time series can be calculated using a mutual information (MI) approach and the false nearest neighbors (FNN) algorithm, respectively. To evaluate the forecasting results of the DNR-SMS method, a comprehensive experiment has been conducted in which the performance of the DNR approach has been compared with that of other commonly used time series prediction algorithms.

The main contributions of this study are as follows: First, a novel DNR model that considers the dendrite strength of each individual branch is proposed. This approach can significantly enhance the regression performance for wind speed forecasting. Second, due to its powerful search ability, the SMS algorithm is used to optimize the neural architecture of the DNR model. The SMS algorithm can escape from local minima more effectively than gradient-based optimization algorithms can. Third, the results of extensive experiments demonstrate that a DNR model trained using the SMS algorithm can achieve highly competitive performance in wind speed forecasting compared with other state-of-the-art prediction techniques. The remainder of the paper is organized as follows: Section II describes the DNR approach. Section III introduces the process of wind speed prediction. Section IV presents and discusses the experimental results. Finally, concluding remarks are presented in Section V.

II. Dendritic Neural Regression

In this study, we demonstrate the utilization of DNR for wind speed prediction and evaluate its performance at various time scales. As illustrated on the left side of Fig. 1, the neural architecture for DNR includes four layers. The first layer is the synaptic layer, which represents the specific tissue that receives electrical or chemical signals from other neurons. The second layer is the dendrite layer, which consists of many branches that integrate the output signals from the synaptic layer. The third layer is the membrane layer, which sums the outputs of the dendritic layer and transfers the result to the next layer. The final layer is the cell body (soma), which compares the signal against a given threshold. If the signal is larger than the threshold, the soma fires; otherwise, no action is performed.

A. Synaptic Layer

Synapses are a kind of neural tissue that conveys information between dendrites and axons or among dendrites of different

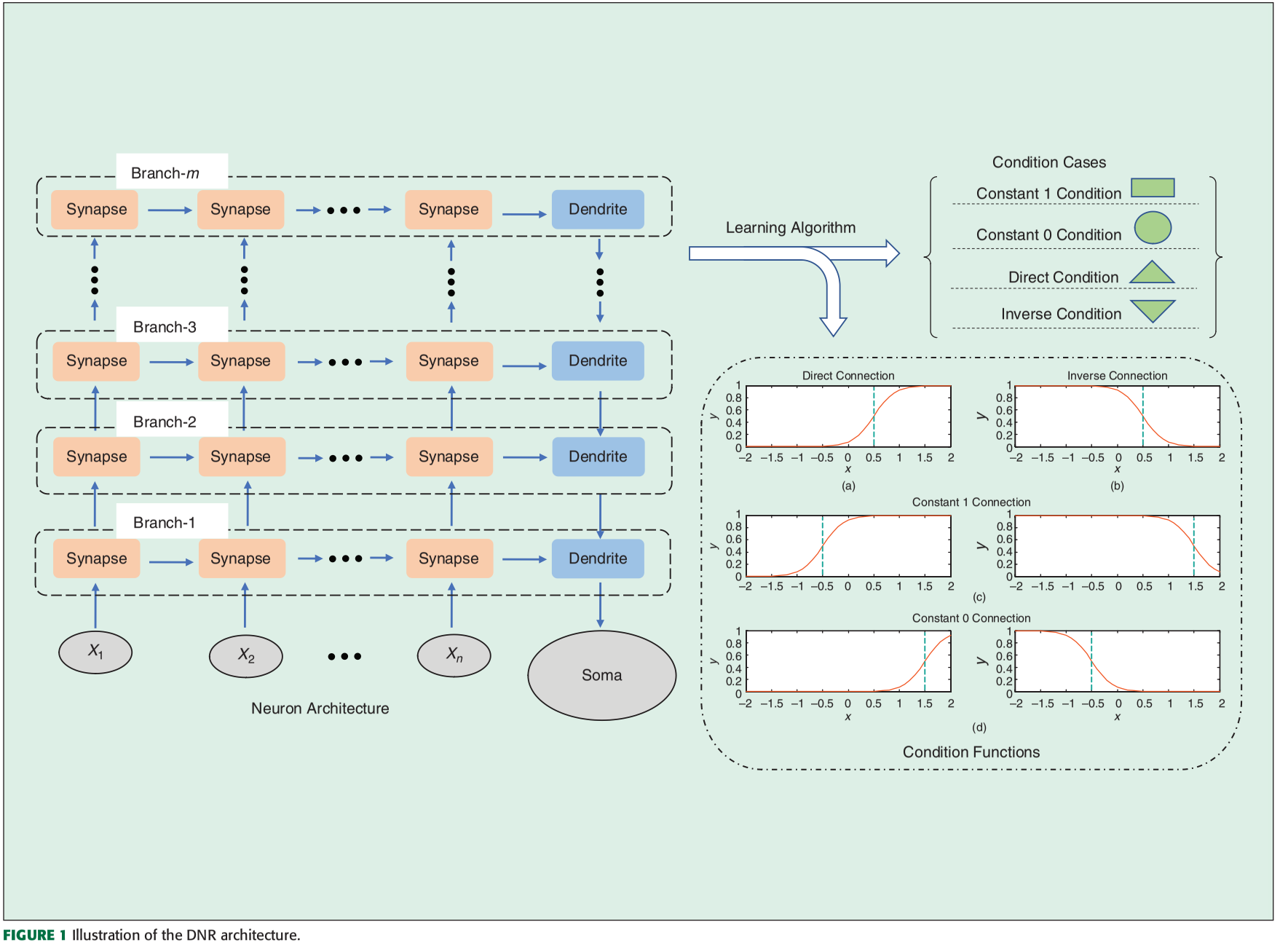

FIGURE 1 Illustration of the DNR architecture.

The neural architecture of DNR consists of four layers, namely the synaptic layer, dendritic layer, membrane layer, and soma layer.

neurons. These elements are distributed throughout the dendritic tree and possess various receptors for specific ions. Depending on the potential of the ions entering a receptor, the synapse changes its connection state and enters either an excitatory or inhibitory state [45]. The process of signal transmission can be described using the following equation:

$$y_{im} = \frac{1}{1 + e^{-k(w_{im}x_i - q_{im})}}, \tag{1}$$

where x_i represents the i-th input feature, whose range is [0,1], with $i \in [1, 2, \ldots, I]$; y_{im} is the output of the i-th synapse on the m-th dendritic branch, with $m \in [1, 2, \ldots, M]$; k is a hyperparameter that is a positive constant; and w_{im} and q_{im} are connection parameters that represent a weight and a threshold, respectively. To obtain the appropriate values for each problem, these connection parameters in a DNR model can be trained using a learning algorithm.

B. Dendrite Layer

Each branch in the dendrite layer receives the output signals from all synapses on that branch. The nonlinear relationship among these signals plays a key role in neural information processing for several sensory systems in biological networks, such as the visual and auditory systems [46], [47]; in DNR, this relationship can be expressed in terms of multiplication operations. Let Z_m represent the output of the m-th dendritic branch. The equation for a dendritic branch can be expressed as follows:

$$Z_m = \prod_{i=1}^{I} y_{im}. \tag{2}$$

C. Membrane Layer

The membrane layer combines all outputs from the dendrite layer through a summation operation. Let V represent the output of the membrane layer. The corresponding equation can be expressed as follows:

$$V = \sum_{m=1}^{M} (u_m * Z_m), \tag{3}$$

where u_m represents the strength of the m-th dendritic branch. This value is constant and is always set to 1 for each branch in the original DNM to simplify the neural architecture to accelerate the computation process [29]. However, in reality, the thicknesses and signal transformation strengths of the dendritic branches vary; thus, using a uniform u_m value for all branches may degrade the regression ability of the DNM. Therefore, in DNR, each u_m is regarded as a parameter that needs to be optimized via a learning algorithm; accordingly, these values are specified differently for different problems.

D. Cell Body (Soma)

The soma fires depending on whether the membrane potential exceeds a given threshold. This process can be mathematically described as a sigmoid operation on the product terms, as follows:

$$O = \frac{1}{1 + e^{-k(V - qs)}}, \tag{4}$$

where V represents the output of the membrane layer and k and qs are positive constant hyperparameters.

E. Connection Cases

On the right side of Fig. 1, the six functions of the synaptic layer are illustrated for various combinations of w_{im} and q_{im}. According to these six different functions, the connection states of the synaptic layer can be divided into four main categories, defined as follows: constant 1 connections, whose parameters satisfy $w_{im} < 0 < q_{im}$ or $0 < w_{im} < q_{im}$, implying that regardless of the input, the output is always excitatory; constant 0 connections, whose parameters satisfy $q_{im} < w_{im} < 0$ or $q_{im} < 0 < w_{im}$, implying that regardless of the input, the output remains inhibitory; excitatory connections, whose parameters satisfy $0 < q_{im} < w_{im}$, implying that the input and output are directly correlated; and inhibitory connections, whose parameters satisfy $w_{im} < q_{im} < 0$, implying that the input and output are inversely correlated.

F. Learning Algorithm

Because of the multiplication operations applied in the dendrite layer, the parameter space of the model appears to be extremely large and complicated. Additionally, weights are added to the output of each dendritic branch in the DNR model, leading to an increase in the dimensionality of the parameter space, which further increases the difficulty of optimizing the parameters. Consequently, it is difficult to perfectly train the parameters when using the traditional BP algorithm. Therefore, in this study, the SMS algorithm is adopted as a more suitable global optimization algorithm to optimize the DNR model. The SMS algorithm is an evolutionary algorithm that mimics the variation in the states of matter. Compared with the traditional BP algorithm, the SMS algorithm exhibits a higher search ability, is less likely to fall into local optima, and seldom results in overfitting. In this subsection, this algorithm is described in detail.

The process of searching for the best solution in the SMS algorithm can be expressed as a series of physical motions among molecules, which mimic the state transformations of matter [38]. Specifically, the SMS algorithm can be divided

into three phases: the gas phase, the liquid phase, and the solid phase. The gas phase corresponds to the previous solution set. In the gas state, the molecules are far from one another and widely distributed, and intense motions and collisions occur among them. The liquid phase represents the intermediate solution set. In the liquid state, the distances between molecules are considerably smaller than those in the gas state, the distribution is relatively concentrated, and the movements are relatively limited. Finally, the solid phase represents the later solution set. In the solid state, the distances between molecules are minimal, the distribution is highly concentrated, and only a small amount of motion occurs.

The process of searching for the best solution in the SMS algorithm can be expressed as a series of physical motions among molecules, which mimic the state transformations of matter.

In each phase, three operations are executed between the molecules, defined as follows:

(1) Direction Vector Operator: The purpose of the direction vector operator is to push other individuals to move toward the best individual in the population. Let P_i represent a member of the population, and let p_{best} represent the current best individual. P_{best} attracts other members of the population, causing them to move toward P_{best}. Let d_i be the direction vector of the i-th member. The corresponding equation is as follows:

$$d_i^{t+1} = d_i^t * \left(1 - \frac{t}{epoch}\right) * 0.5 + \frac{P_{\text{best}} - P_i}{\left\| P_{\text{best}} - P_i \right\|}, \tag{5}$$

where t denotes the current iteration number and $epoch$ denotes the total number of iterations. The obtained value d_i is used to calculate the velocity vector of the i-th member of the population, as follows:

$$v_i = d_i * \frac{\sum_{m=1}^{n} \left(b_m^{\text{high}} - b_m^{\text{low}}\right)}{n} * \gamma, \tag{6}$$

where γ is a constant positive value with a range of [0,1] and n represents the dimensionality of each member, namely, the number of parameters in the model. The upper and lower bounds on the m-th parameter are b_m^{high} and b_m^{low}, respectively. Next, we can employ v_i to update the position p_i, where the update function is defined as follows:

$$P_{i,m}^{t+1} = p_{i,m}^t + v_m * Rand_1 * \left(b_m^{\text{high}} - b_m^{\text{low}}\right) * \alpha, \tag{7}$$

where $Rand_1$ is a random number between 0 and 1 and α is a constant positive value with a range of [0,1].

(2) Collision Operator: When two molecules approach each other, they may collide. The collision operator emulates this process; if two individuals have collided, their direction vectors are exchanged, which ensures the diversity of the population and prevents premature convergence in the evolutionary process. Collision behavior occurs when two individuals are sufficiently close to each other. Let r denote the corresponding distance threshold, which is calculated as follows:

$$r = \frac{\sum_{m=1}^{n} \left(b_m^{\text{high}} - b_m^{\text{low}}\right)}{n} * \beta, \tag{8}$$

where β is a constant positive value with a range of [0,1]. If the distance between two individuals is less than r, they will undergo collision, and this phenomenon can be expressed as follows:

$$d_m = d_i \quad AND \quad d_i = d_m. \tag{9}$$

(3) Random Behavior: Following the rules of the SMS algorithm, during the iterative process, each individual will likely exhibit random behavior, i.e., the elements of the individuals will randomly change. The function for this random behavior can be expressed as follows:

$$P_{i,m}^{t+1} = \begin{cases} b_m^{\text{low}} + Rand_2 * \left(b_m^{\text{high}} - b_m^{\text{low}}\right), \\ \text{with probability } H, \\ P_{i,m}^{t+1}, \text{ with probability } (1 - H), \end{cases} \tag{10}$$

where H represents the occurrence probability of the random behavior.

The entire iterative process in the SMS algorithm is divided into three phases by establishing different γ, α, β and H values. In accordance with [38], we adopt the parameters presented in Table II for the three phases. The complete SMS process is described in Algorithm 1.

III. Wind Speed Forecasting Framework

The process of applying DNR for wind speed prediction is illustrated in Fig. 2. First, the time delay and embedding dimensionality of the series are calculated to reconstruct the phase space. Next, the maximum Lyapunov exponent is calculated from the time delay and embedding dimensionality. When this exponent exceeds zero, the wind speed series data are regarded as a chaotic time series. Subsequently, the

TABLE I Parameter settings of the SMS algorithm in different states.

STATE	DURATION	γ	α	β	PROBABILITY H
GAS	50%	0.8	0.8	[0.8,1]	0.9
LIQUID	40%	0.4	0.2	[0.0,0.6]	0.2
SOLID	10%	0.1	0	[0.0,0.1]	0.0

wind speed time series data are processed through a normalization operation. Finally, the DNR model can be employed for wind speed prediction.

A. Phase Space Reconstruction

It is difficult to forecast the future trend of a chaotic time series such as a wind speed series because of its irregularity. A chaotic time series can be considered to represent a type of random motion in a definite dynamical system. Thus, we need to restore the original dynamical system of the chaotic time series. The most common method to accomplish this is the delayed coordinate approach proposed by Takens [44]. For a chaotic time series $\{x(i)|i=1,2,\ldots,N\}$, under the assumption that the time delay τ and vector embedding dimensionality m have been calculated using a certain approach, a group of new vectors can be constructed as follows:

$$X=\begin{pmatrix} x_1 & x_2 & \cdots & x_{N-1-\tau(m-1)} \\ x_{\tau+1} & x_{\tau+2} & \cdots & x_{N-1-\tau(m-2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_{\tau(m-1)+1} & x_{\tau(m-1)+2} & \cdots & x_{N-1} \end{pmatrix}, \tag{11}$$

$$T=(x_{\tau(m-1)+2},\ x_{\tau(m-1)+3},\ \ldots,\ x_N), \tag{12}$$

where X and T are the input signals of the neural model and the target outputs, respectively. The matrix X can efficiently describe the original dynamical system if the appropriate values of the time delay τ and embedding dimensionality m are selected. The operation of constructing X from a time series $\{x(i)\,|\,i=1,2,\ldots,N\}$ is termed phase space reconstruction.

The core problem in performing this operation is to obtain the appropriate values for τ and m. Takens only proved the existence of the time delay and embedding dimensionality in theory and did not provide a method for determining their values. In general, a time series always contains finite sequences with noise. There is no single method that can accurately obtain the time delay and embedding dimensionality for every time series; instead, the appropriate algorithm must be chosen depending on the actual situation. In this study, the FNN algorithm is utilized to calculate the embedding dimensionality [48], and the MI algorithm is employed to obtain the time delay [49]. The details of these two methods are presented in the two subsequent subsections.

Algorithm 1 Pseudocode for the SMS algorithm.

Input: Population size N, Number of dimensions n, Maximum number of iterations *epoch*.
Result: Best solution P_{best}.
begin
  Initialize the population $\mathcal{X}=\{P_1,\ldots,P_N\}$, the direction vector set $\mathcal{D}^0=\{d_1^0,\ldots,d_N^0\}$, the maximum number of iterations *epoch* = 1000, the state count *phase* = 1, and the current iteration number $t=1$;
  repeat
    if *phase* == 1 **then**
      $\gamma=0.8,\ \alpha=0.8,\ \beta=0.9,\ H=0.9,\ Dend=epoch*0.5$;
    if *phase* == 2 **then**
      $\gamma=0.4,\ \alpha=0.2,\ \beta=0.5,\ H=0.2,\ Dend=epoch*0.9$;
    if *phase* == 3 **then**
      $\gamma=0.1,\ \alpha=0.0,\ \beta=0.0,\ H=0.0,\ Dend=epoch$;
    for $t, t<=Dend, t=t+1$ **do**
      Evaluate the fitness of the population $\mathcal{F}(\mathcal{X})=\{f(P_1),\ldots,f(P_N)\}$;
      Set the individual with the best fitness as P_{best};
      /* Perform direction vector operations */
      Calculate the new direction vector set $\mathcal{D}^t$ with Eq. (5);
      Use $\mathcal{D}^t$ to calculate the velocity vector set $\mathcal{V}=\{v_1,\ldots,v_N\}$ with Eq. (6);
      Update each individual P_n with Eq. (7);
      /* Perform collision operations */
      Calculate the threshold r using Eq. (8);
      Calculate the distance between each pair of points; if the distance is less than r, exchange the direction vectors of the points, as indicated in Eq. (9);
      /* Perform random behavior */
      Draw a random number *RA* in the range of [0,1]; if *RA* is less than H, execute random behavior with Eq. (10);
    phase = phase + 1;
  until *phase* > 3;
return the best solution P_{best}.

B. Mutual Information Algorithm

The MI algorithm is one of the most widely used methods for calculating the time delay of a time series and is based on the theory of mutual information. The information entropy, $H(X)$, which is used to represent the degree of uncertainty of X, can be expressed as follows:

$$H(X)=-\sum_{i=1}^{n}P(x_i)\log P(x_i), \tag{13}$$

where X denotes discrete random variables, $P(x_i)$ is the probability of the occurrence of event x, and n is the total number of states x. $H(X\,|\,Y)$ represents the conditional information entropy and can be expressed as

$$H(X|Y)=-\sum_{i=1}^{n}\sum_{j=1}^{m}P(y_j)\,P(x_i|y_j)\log P(x_i|y_j), \tag{14}$$

where m represents the total number of states y, $P(y_j)$ is the probability of event y occurring alone, and $P(x_i|y_j)$ denotes the conditional probability of event x occurring under the condition of the occurrence of event y. The MI entropy, $I(X,\,Y)$, can be calculated as

$$I(X,Y)=H(X)+H(Y)-H(X,Y), \tag{15}$$

where $H(X, Y)$ represents the joint information of X and Y and can be calculated using the following expression:

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j)\log P(x_i, y_j). \tag{16}$$

For the time series $\{x(i)\,|\,i=1,2,\ldots,N\}$, $P(x_l)$ and $P(x_{l+\tau})$ are the probabilities that x_l and $x_{l+\tau}$, respectively, appear in $\{x(i)\,|\,i=1,2,\ldots,N\}$. Based on these definitions, the MI entropy $I(x_l, x_{l+\tau})$ for a time delay τ can be specified as follows:

$$I(\tau) = I(x_l, x_{l+\tau}) = H(x_l) + H(x_{l+\tau}) - H(x_l, x_{l+\tau}). \tag{17}$$

According to the MI algorithm, the value of τ when $I(\tau)$ reaches a local minimum for the first time is taken as the final solution.

C. False Nearest Neighbors Algorithm

The FNN algorithm, which was proposed by Kennel in 1992 [48], is used to calculate the embedding dimensionality for a chaotic time series. This algorithm is based on the premise that a chaotic time series, such as a series of wind speed data, can be regarded as a set of continuously varying particles in a high-dimensional space mapped to a one-dimensional space. If the number of embedding dimensions is too small, the particles will be compressed and folded onto one another due to the insufficient extent of their spatial orbits, meaning that two adjacent points in the one-dimensional space may correspond to two particles separated by a large distance in the high-dimensional space. Two such adjacent particles are defined as false nearest neighbor points. In this case, the embedding dimensionality should be gradually increased to fully expand the spatial orbits. With an increase in the number of embedding dimensions, the particles are expected to gradually separate, and the number of false nearest neighbor points is expected to gradually decrease. Once the embedding dimensionality is set to a sufficient value such that all false nearest neighbor points are eliminated, the corresponding solution is considered to be the optimal solution.

Suppose that $z_i(m) = (y(i), y(\tau+i), \ldots, y((m-1)\tau+i))$ is a vector in an m-dimensional phase space and that $z_j(m)$ is the nearest neighbor point of z_i; the distance between $z_i(m)$ and $z_j(m)$ can be calculated using the following equation:

$$R_i(m) = \| z_i(m) - z_j(m) \|. \tag{18}$$

This distance changes as the number of dimensions increases. Thus, the updated distance between $z_i(m+1)$ and $z_j(m+1)$ can be expressed as follows:

$$R_i^2(m+1) = R_i^2(m) + \| z_i(i+m\tau) - z_j(j+m\tau) \|. \tag{19}$$

If $R_i(m+1)$ is considerably larger than $R_i(m)$, the two points are considered to be false nearest neighbor points. By modifying this equation, a more convenient function for judging false nearest neighbor points can be obtained as follows:

$$R_\tau = \frac{\| z_i(i+m\tau) - z_j(j+m\tau) \|}{R_i(m)}. \tag{20}$$

R_τ is compared against a given threshold θ, whose range is set to [10, 50]. If R_τ is larger than this threshold, the points are determined to be false nearest neighbor points. For a real-world chaotic time series, such as a wind speed series, the initial number of embedding dimensions is usually set to 2. Once the proportion of false nearest neighbor points is less than 5%, the corresponding number of embedding dimensions is selected as the final solution. In certain extreme cases, the proportion of false nearest neighbor points cannot decrease to 5%; in such a case, the critical dimensionality at which the number of false nearest neighbor points stops decreasing is considered the final solution.

D. Maximum Lyapunov Exponent

Before a machine learning technique is employed to forecast the future wind speed, the chaotic characteristics of the historical wind speed series must be confirmed. Lyapunov exponents are commonly used for this purpose. In 1983, it was proven that if at least one of the Lyapunov exponents in a dynamical system is positive, the corresponding time series can be considered chaotic [50]. Therefore, the maximum Lyapunov exponent of the time series needs to be calculated. If this exponent exceeds zero, the time series is considered to have chaotic characteristics. Among the various approaches for calculating Lyapunov exponents, Wolf's method [51], which is based on the phase space reconstruction theory of Takens, is considered to be the most effective.

For a series $X(t) = (x(t), x(\tau+t), \ldots, x((m-1)\tau+t)$ reconstructed using Takens' theory, as described above, the distance L_i between $X(t_i)$ and the closest point $X(t_n)$ can be expressed as follows:

$$L_i = \| X(t_n) - X(t_i) \|. \tag{21}$$

According to Wolf's method, the maximum Lyapunov exponent can be calculated as

$$\lambda_{\max} = \frac{1}{t_M - t_0}\sum_{i=0}^{M} \ln\frac{L_i}{L_i'}, \tag{22}$$

where t_0 and t_M represent the initial time and the final time, respectively, and $M = N - (m-1)\tau$. In addition, the interval for time series prediction can be calculated as follows [52], [53]:

$$\Delta t = \frac{1}{\lambda_{\max}}. \tag{23}$$

It is difficult to forecast the future trend of a chaotic time series such as a wind speed series because of its irregularity. But it can be considered to represent a type of random motion in a definite dynamical system.

This equation indicates that larger and smaller $\lambda_{\max}$ values correspond to closer and more distant predictions, respectively.

E. Normalization

For a raw time series, the first task is normalization. Normalization is a necessary and useful preprocessing step in the field of ANNs. By normalizing the data, all features can be mapped to the same scale, thereby reducing the computational cost and increasing the convergence speed of optimization algorithms. In this study, the one-dimensional wind speed time series $\{x(i)\,|\,i=1,2,\ldots,N\}$ is normalized using the min-max method, as follows:

$$Y_t = \frac{x_t - MIN(x_t)}{MAX(x_t) - MIN(x_t)}, \tag{24}$$

where $MAX(X_t)$ and $MIN(X_t)$ represent the maximum and minimum values of the input vector, respectively. Using Eq. (24), both the training and test samples are normalized to the range [0,1]. In addition to min-max normalization, other methods, such as mean and variance normalization and simple normalization, are also commonly used [54], [55].

IV. Experiments and Discussion

A. Experimental Setup

The wind speed data used for experimental analysis in this study pertain to a wind farm in Galicia and are available for free at http://www.evwind.es. To comprehensively evaluate the prediction performance of DNR, two sets of wind speed data with different time intervals, specifically, ten minutes and one day, were collected; these datasets are illustrated in Figs. 3(a) and (b), respectively. The two datasets were used to verify long-term and short-term predictions. Each dataset contained 1,150 data points. The collected data were divided into two subsets: training data and test data. The training data were used to train the DNR parameters, and the test data were utilized to evaluate the performance of the trained model in terms of several indicators. The use of different ratios of training data to test data can help determine the dependence of an algorithm on the number of training data; thus, three ratios were considered in our experiments: 1:1, 4:1 and 9:1. All experiments were conducted on a personal PC with a 2.90 GHz Intel(R) Core i7 CPU and 16 GB of memory using MATLAB R2018b.

B. Evaluation Indicators

To fairly evaluate the performance of each algorithm, several evaluation methods were adopted in this study. The details of these methods are given as follows.

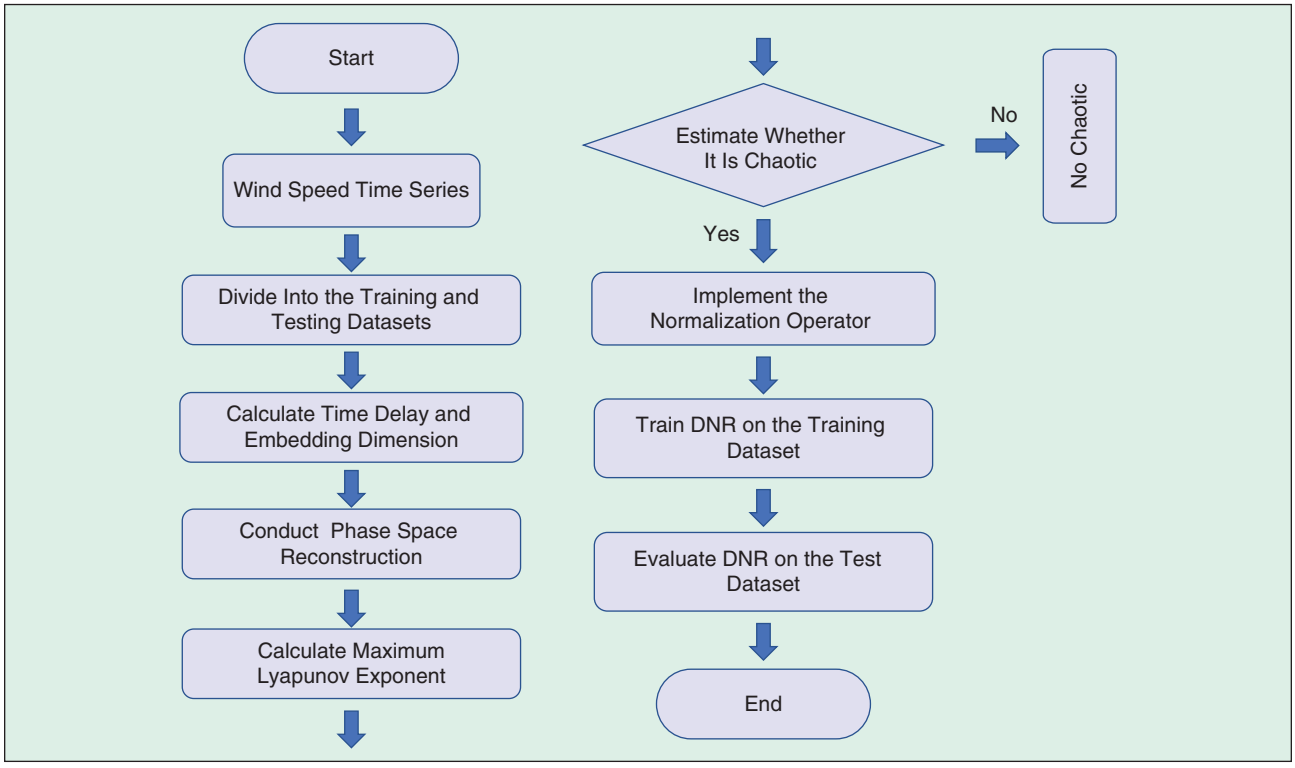

FIGURE 2 Flowchart of DNR-based wind speed forecasting.

(1) Evaluation Indicators: To compare the prediction performance of the algorithms, four widely used indicators were evaluated in our experiments: the mean-squared error (*MSE*), the mean absolute error (*MAE*), the root-mean-squared error (*RMSE*) and the correlation coefficient of the predictions (*R*). These indicators are defined as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(O-T)^2, \tag{25}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|T-O|, \tag{26}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(O-T)^2}, \tag{27}$$

$$R = \frac{\sum_{i=1}^{n}(T-\bar{T})(O-\bar{O})}{\sqrt{\sum_{i=1}^{n}(T-\bar{T})^2\sum_{i=1}^{n}(O-\bar{O})^2}}, \tag{28}$$

where O is the prediction output and T is the actual result. *MSE*, *MAE*, and *RMSE* represent different kinds of errors between the outputs and targets. Smaller indicator values correspond to higher prediction performance. R represents the degree of fit between the output curve of a prediction algorithm and the original curve. If R is equal to 1, the predicted curve fits the original curve perfectly.

(2) Nonparametric Statistical Test: This test detects whether a significant difference exists between the proposed method and another algorithm. In this study, the Wilcoxon rank sum test [56], [57] was conducted using the KEEL software [58]. The significance level was set to 5%. If the *p-value* of the Wilcoxon rank sum test was less than 5%, a significant difference in performance was considered to exist between the two algorithms being compared.

(3) Relative Graphs: Graphs related to an experiment enable more intuitive observation of the experimental results. The following graphs were generated in this study. First, fit graphs and correlation coefficient graphs were plotted to visualize how well the predicted results matched the target values. Second, a convergence graph was produced to illustrate the speed and stability of the proposed model during the convergence process.

C. Performance Comparison and Analysis

Following the approaches described above, the time delays, embedding dimensionalities and maximum Lyapunov exponents of the wind speed series at different time scales were calculated, and the results are presented in Table II. The maximum Lyapunov exponents presented in Table II imply that the two wind speed time series are chaotic. Next, phase space reconstruction was conducted based on the time delay and embedding dimensionality.

Before employing the DNR model to forecast the wind speed, three hyperparameters, *M*, *pop*, and *epoch*, which could influence the performance of the proposed algorithm, were established. *M* denotes the number of dendritic branches, and *Pop* and *epoch* represent the population size and maximum iteration time, respectively, for the SMS algorithm. It should be noted that determining the best hyperparameter combination is a difficult and time-consuming task. Although several approaches have been proposed for the automatic tuning of hyperparameters, the results are comparable to those of the trial-and-error approach. Based on empirical evidence, *k*, *qs*, *M*, *pop*, and *epoch* were set to 6, 0.8, 9, 100 and 1000, respectively, in this experiment.

Moreover, several commonly used prediction techniques were compared with the proposed model, including the

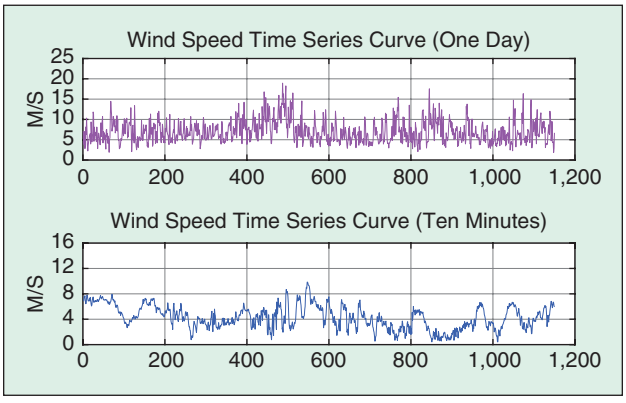

FIGURE 3 Wind speed curves for long-term and short-term predictions.

TABLE II Results for the time delays, embedding dimensionalities and maximum Lyapunov exponents of wind speed series at different time scales.

TIME INTERVAL	TIME DELAY (τ)	EMBEDDING DIMENSIONALITY (m)	MAXIMUM LYAPUNOV EXPONENT (λ_{max})	CHAOTIC
ONE DAY	2	4	0.1181	YES
TEN MIN	5	7	0.0330	YES

TABLE III Parameter settings of algorithms for predicting wind speed time series.

ALGORITHM	PARAMETERS
MLP	*HIDDENLAYER* = 10, *LEARNINGRATE* = 0.01, *EPOCH* = 1000
ENN	*LEARNINGRATE* = 0.01, *EPOCH* = 1000
SVR	LINEAR, POLYNOMIAL, RBF AND SIGMOID KERNELS
LSTM	*HIDDENUNITS* = 200, *EPOCH* = 1000
DNM-BP	*LEARNINGRATE* = 0.01, *EPOCH* = 1000
EDNM	*POPSIZE* = 100, *EPOCH* = 1000
DNR-BP	*LEARNINGRATE* = 0.01, *EPOCH* = 1000
DNR-SMS	*POPSIZE* = 100, *EPOCH* = 1000

TABLE IV Long-term prediction performance of all models for the experimental wind speed time series.

LONG-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 1:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
ENN	1.19E-01±3.77E-01	9.13E-07	2.28E-01±2.59E-01	9.13E-07	1.99E+00±3.50E-01	9.13E-07	4.01E-01±1.26E-01
MLP	2.86E-02±5.42E-03	9.13E-07	1.68E-01±1.54E-02	9.13E-07	2.24E+00±2.18E-01	9.12E-07	2.94E-01±1.18E-01
DT	2.33E-02±0.00E+00	9.13E-07	1.53E-01±0.00E+00	9.12E-07	2.04E+00±0.00E+00	9.12E-07	4.45E-01±0.00E+00
SVR-L	1.66E-02±0.00E+00	1.00E+00	1.29E-01±0.00E+00	1.00E+00	**1.68E+00±0.00E+00**	1.00E+00	5.59E-01±0.00E+00
SVR-P	2.07E-02±0.00E+00	9.13E-07	1.44E-01±0.00E+00	9.12E-07	1.98E+00±0.00E+00	9.12E-07	4.56E-01±0.00E+00
SVR-R	**1.65E-02±0.00E+00**	1.00E+00	**1.28E-01±0.00E+00**	1.00E+00	1.70E+00±0.00E+00	1.00E+00	**5.68E-01±0.00E+00**
SVR-S	1.68E-02±0.00E+00	9.94E-01	1.30E-01±0.00E+00	9.93E-01	1.69E+00±0.00E+00	1.00E+00	5.51E-01±0.00E+00
LSTM	1.99E-02±2.10E-04	9.13E-07	1.41E-01±7.48E-04	9.11E-07	1.95E+00±1.27E-02	9.12E-07	4.90E-01±1.25E-02
DNM-BP	1.70E-02±3.63E-04	5.69E-01	1.30E-01±1.39E-03	5.69E-01	1.70E+00±4.40E-02	9.41E-01	5.50E-01±1.97E-02
EDNM	1.72E-02±4.20E-04	2.10E-03	1.31E-01±1.60E-03	2.10E-03	1.72E+00±3.48E-02	2.11E-01	5.50E-01±1.35E-02
DNR-BP	1.75E-02±1.86E-03	3.04E-01	1.32E-01±6.59E-03	3.04E-01	1.73E+00±9.29E-02	5.73E-01	5.41E-01±7.08E-02
DNR-SMS	1.69E-02±2.04E-04	-	1.30E-01±7.85E-04	-	1.72E+00±1.07E-02	-	5.53E-01±5.73E-03

LONG-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 4:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
ENN	3.15E-02±4.48E-02	9.13E-07	1.62E-01±7.20E-02	9.13E-07	1.83E+00±1.21E-01	9.13E-07	3.92E-01±7.47E-02
MLP	2.72E-02±4.10E-03	9.13E-07	1.64E-01±1.24E-02	9.13E-07	2.12E+00±1.63E-01	9.13E-07	2.83E-01±1.21E-01
DT	2.57E-02±0.00E+00	9.11E-07	1.60E-01±0.00E+00	9.12E-07	2.13E+00±0.00E+00	9.12E-07	3.54E-01±0.00E+00
SVR-L	1.73E-02±0.00E+00	3.32E-06	1.32E-01±0.00E+00	3.32E-06	1.69E+00±0.00E+00	1.37E-06	5.01E-01±0.00E+00
SVR-P	2.01E-02±0.00E+00	9.13E-07	1.42E-01±0.00E+00	9.12E-07	1.88E+00±0.00E+00	9.12E-07	4.03E-01±0.00E+00
SVR-R	1.71E-02±0.00E+00	3.75E-01	1.31E-01±0.00E+00	3.83E-01	1.68E+00±0.00E+00	8.35E-05	5.08E-01±0.00E+00
SVR-S	1.75E-02±0.00E+00	9.11E-07	1.32E-01±0.00E+00	9.12E-07	1.70E+00±0.00E+00	9.12E-07	4.94E-01±0.00E+00
LSTM	1.86E-02±1.96E-04	9.12E-07	1.36E-01±7.19E-04	9.12E-07	1.81E+00±1.42E-02	9.13E-07	4.55E-01±7.19E-03
DNM-BP	1.76E-02±1.38E-03	1.30E-03	1.32E-01±4.80E-03	1.40E-03	1.71E+00±8.51E-02	2.70E-03	4.97E-01±8.60E-02
EDNM	1.76E-02±6.87E-04	6.91E-04	1.33E-01±2.55E-03	7.15E-04	1.70E+00±2.67E-02	7.70E-05	5.03E-01±2.40E-02
DNR-BP	1.74E-02±3.52E-04	2.00E-03	1.32E-01±1.32E-03	2.10E-03	1.68E+00±4.09E-02	1.38E-01	5.09E-01±5.46E-03
DNR-SMS	**1.71E-02±1.84E-04**	-	**1.31E-01±7.05E-04**	-	**1.67E+00±9.46E-03**	-	**5.14E-01±6.22E-03**

LONG-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 9:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
ENN	2.70E-02±3.88E-03	1.01E-06	1.64E-01±1.10E-02	1.01E-06	2.01E+00±7.23E-02	9.13E-07	4.05E-01±5.49E-02
MLP	3.94E-02±6.43E-03	9.13E-07	1.98E-01±1.60E-02	9.13E-07	2.56E+00±2.31E-01	9.13E-07	2.05E-01±1.03E-01
DT	3.36E-02±0.00E+00	9.12E-07	1.83E-01±0.00E+00	9.12E-07	2.36E+00±0.00E+00	9.12E-07	3.31E-01±0.00E+00
SVR-L	2.25E-02±0.00E+00	4.88E-06	1.50E-01±0.00E+00	4.88E-06	1.89E+00±0.00E+00	9.12E-07	4.76E-01±0.00E+00
SVR-P	2.58E-02±0.00E+00	9.12E-07	1.61E-01±0.00E+00	9.12E-07	2.08E+00±0.00E+00	9.12E-07	3.70E-01±0.00E+00
SVR-R	2.22E-02±0.00E+00	1.38E-01	1.49E-01±0.00E+00	1.33E-01	1.88E+00±0.00E+00	4.88E-06	4.85E-01±0.00E+00
SVR-S	2.28E-02±0.00E+00	9.12E-07	1.51E-01±0.00E+00	9.12E-07	1.91E+00±0.00E+00	9.12E-07	4.68E-01±0.00E+00
LSTM	2.41E-02±9.15E-04	9.13E-07	1.55E-01±2.93E-03	9.13E-07	2.08E+00±1.05E-01	9.13E-07	4.90E-01±5.96E-03
DNM-BP	2.24E-02±1.81E-03	4.75E-01	1.50E-01±5.53E-03	4.75E-01	1.89E+00±1.02E-01	1.16E-01	4.75E-01±1.05E-01
EDNM	2.25E-02±1.17E-03	8.10E-02	1.50E-01±3.80E-03	8.10E-02	1.88E+00±5.46E-02	9.23E-02	4.88E-01±3.28E-02
DNR-BP	2.24E-02±4.79E-04	1.18E-02	1.50E-01±1.58E-03	1.12E-02	1.87E+00±3.47E-02	5.66E-02	4.90E-01±1.22E-02
DNR-SMS	**2.21E-02±3.55E-04**	-	**1.49E-01±1.20E-03**	-	**1.86E+00±1.35E-02**	-	**4.94E-01±9.41E-03**

TABLE V Short-term prediction performance of all models for the experimental wind speed time series.

SHORT-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 1:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
ENN	4.35E-02±6.37E-02	9.13E-07	1.80E-01±1.06E-01	9.13E-07	9.84E-01±2.42E-01	9.13E-07	6.03E-01±2.53E-01
MLP	2.87E-02±1.08E-02	9.13E-07	1.66E-01±3.15E-02	9.13E-07	1.21E+00±2.24E-01	9.13E-07	5.75E-01±2.01E-01
DT	1.21E-02±0.00E+00	9.13E-07	1.10E-01±0.00E+00	9.13E-07	7.94E-01±0.00E+00	9.13E-07	8.09E-01±0.00E+00
SVR-L	**4.69E-03±0.00E+00**	1.00E+00	**6.85E-02±0.00E+00**	1.00E+00	**4.89E-01±0.00E+00**	1.00E+00	**9.26E-01±0.00E+00**
SVR-P	2.45E-02±0.00E+00	9.13E-07	1.56E-01±0.00E+00	9.13E-07	1.18E+00±0.00E+00	9.13E-07	8.13E-01±0.00E+00
SVR-R	5.85E-03±0.00E+00	8.59E-06	7.65E-02±0.00E+00	8.59E-06	5.50E-01±0.00E+00	1.37E-06	9.25E-01±0.00E+00
SVR-S	5.22E-03±0.00E+00	9.97E-01	7.22E-02±0.00E+00	9.97E-01	5.18E-01±0.00E+00	9.81E-01	9.15E-01±0.00E+00
LSTM	1.47E-02±1.99E-03	9.13E-07	1.21E-01±8.00E-03	9.13E-07	9.29E-01±6.56E-02	9.13E-07	8.63E-01±5.28E-02
DNM-BP	5.96E-03±3.04E-03	4.19E-01	7.59E-02±1.44E-02	4.67E-01	5.41E-01±1.08E-01	5.08E-01	9.15E-01±5.45E-02
EDNM	2.26E-02±2.96E-02	1.24E-06	1.30E-01±7.56E-02	1.24E-06	7.24E-01±2.95E-01	4.44E-06	7.25E-01±2.65E-01
DNR-BP	9.19E-03±1.98E-02	4.51E-01	8.29E-02±4.82E-02	4.75E-01	6.03E-01±3.84E-01	3.11E-01	8.99E-01±1.38E-01
DNR-SMS	5.43E-03±3.54E-04	-	7.36E-02±2.41E-03	-	5.24E-01±1.58E-02	-	9.21E-01±5.28E-03

SHORT-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 4:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
ENN	1.03E-02±3.13E-02	9.13E-07	7.87E-02±6.42E-02	9.13E-07	5.12E-01±6.91E-02	9.13E-07	8.86E-01±1.24E-01
MLP	1.80E-02±7.20E-03	9.13E-07	1.32E-01±2.66E-02	9.13E-07	1.01E+00±2.14E-01	9.13E-07	6.03E-01±1.78E-01
DT	5.89E-03±0.00E+00	9.13E-07	7.68E-02±0.00E+00	9.13E-07	5.90E-01±0.00E+00	9.13E-07	8.82E-01±0.00E+00
SVR-L	3.42E-03±0.00E+00	3.02E-06	5.85E-02±0.00E+00	2.48E-06	4.48E-01±0.00E+00	2.48E-06	9.35E-01±0.00E+00
SVR-P	1.49E-02±0.00E+00	9.13E-07	1.22E-01±0.00E+00	9.13E-07	9.48E-01±0.00E+00	9.13E-07	7.73E-01±0.00E+00
SVR-R	3.70E-03±0.00E+00	9.13E-07	6.09E-02±0.00E+00	9.13E-07	4.73E-01±0.00E+00	9.13E-07	9.31E-01±0.00E+00
SVR-S	3.63E-03±0.00E+00	1.01E-06	6.02E-02±0.00E+00	1.01E-06	4.66E-01±0.00E+00	9.13E-07	9.30E-01±0.00E+00
LSTM	7.46E-03±3.75E-04	9.13E-07	8.63E-02±2.17E-03	9.13E-07	6.96E-01±1.85E-02	9.13E-07	8.57E-01±8.12E-03
DNM-BP	7.54E-03±2.37E-02	6.67E-01	6.64E-02±5.60E-02	6.75E-01	5.12E-01±4.89E-01	8.67E-01	8.87E-01±2.75E-01
EDNM	3.43E-03±8.12E-04	5.90E-02	5.82E-02±6.05E-03	6.40E-02	4.29E-01±3.11E-02	7.71E-01	9.31E-01±1.69E-02
DNR-BP	9.29E-03±3.26E-02	7.79E-02	6.92E-02±6.70E-02	8.41E-02	5.39E-01±6.00E-01	4.35E-01	9.12E-01±1.43E-01
DNR-SMS	**3.15E-03±1.50E-04**	-	**5.61E-02±1.32E-03**	-	**4.27E-01±1.14E-02**	-	**9.36E-01±3.12E-03**

SHORT-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 9:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
ENN	4.47E-03±4.25E-04	9.13E-07	6.68E-02±3.12E-03	9.13E-07	4.98E-01±2.39E-02	9.13E-07	8.77E-01±1.21E-02
MLP	1.92E-02±4.41E-03	9.13E-07	1.37E-01±1.70E-02	9.13E-07	1.08E+00±1.46E-01	9.13E-07	4.58E-01±1.36E-01
DT	6.73E-03±0.00E+00	9.13E-07	8.20E-02±0.00E+00	9.13E-07	6.23E-01±0.00E+00	9.13E-07	8.14E-01±0.00E+00
SVR-L	3.81E-03±0.00E+00	9.13E-07	6.18E-02±0.00E+00	9.13E-07	4.64E-01±0.00E+00	9.13E-07	8.93E-01±0.00E+00
SVR-P	9.06E-03±0.00E+00	9.13E-07	9.52E-02±0.00E+00	9.13E-07	7.75E-01±0.00E+00	9.13E-07	8.24E-01±0.00E+00
SVR-R	3.85E-03±0.00E+00	9.13E-07	6.21E-02±0.00E+00	9.13E-07	4.69E-01±0.00E+00	9.13E-07	8.93E-01±0.00E+00
SVR-S	3.88E-03±0.00E+00	9.13E-07	6.23E-02±0.00E+00	9.13E-07	4.65E-01±0.00E+00	9.13E-07	8.93E-01±0.00E+00
LSTM	5.56E-03±2.16E-04	9.12E-07	7.46E-02±1.45E-03	9.13E-07	5.81E-01±1.42E-02	9.13E-07	8.51E-01±6.16E-03
DNM-BP	8.09E-03±2.36E-02	3.02E-06	7.10E-02±5.51E-02	3.02E-06	5.39E-01±4.84E-01	1.16E-01	8.90E-01±5.53E-02
EDNM	3.58E-03±2.74E-04	1.33E-01	5.98E-02±2.15E-03	1.29E-01	**4.41E-01±1.30E-02**	9.98E-01	9.02E-01±2.43E-03
DNR-BP	9.46E-03±3.09E-02	1.59E-04	7.29E-02±6.44E-02	1.59E-04	5.66E-01±5.90E-01	2.19E-02	8.83E-01±1.14E-01
DNR-SMS	**3.50E-03±7.30E-05**	-	**5.91E-02±6.17E-04**	-	4.46E-01±7.34E-03	-	**9.03E-01±2.07E-03**

Elman neural network (ENN) [59]; a multilayer perceptron (MLP); a decision tree (DT) model; support vector regression with a linear kernel (SVR-l), a polynomial kernel (SVR-p), a radial basis function (RBF) kernel (SVR-r), and a sigmoid kernel (SVR-s) [60]; a long short-term memory (LSTM) network; and several variants of the DNM, specifically, the original DNM, the DNM trained with the L-SHADE algorithm (EDNM), and a DNR model trained with the BP algorithm. The parameter settings for all algorithms are listed in Table III. To ensure fair comparisons, the experiments with all algorithms were conducted 30 times for each time series.

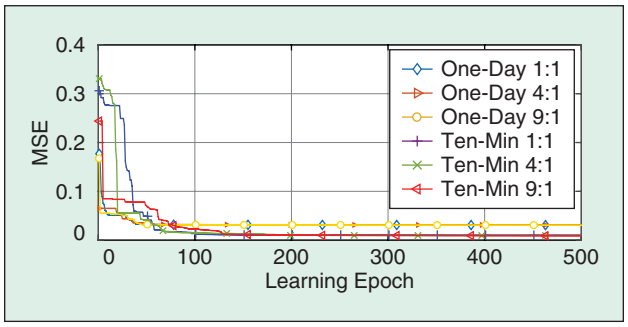

FIGURE 4 Convergence curves of the DNR model for the experimental wind speed time series.

Since the size of the training dataset can impact the forecasting accuracy, the prediction results of DNR-SMS were evaluated on wind speed data with different partition ratios between the training and test datasets, specifically, 1:1, 4:1 and 9:1. The average and standard deviation for each approach are summarized in Tables IV and V. Note that *MSE* and *RMSE* were calculated using the normalized data, whereas *MAE* and *R* were calculated using the raw data. Tables IV and V show that for most algorithms, the prediction results for both the long-term and short-term wind speed time series with a partition ratio of 4:1 were superior to those for a ratio of 1:1; moreover, the prediction results for a partition ratio of 9:1 were inferior to those for 4:1. These findings suggest that merely increasing the number of

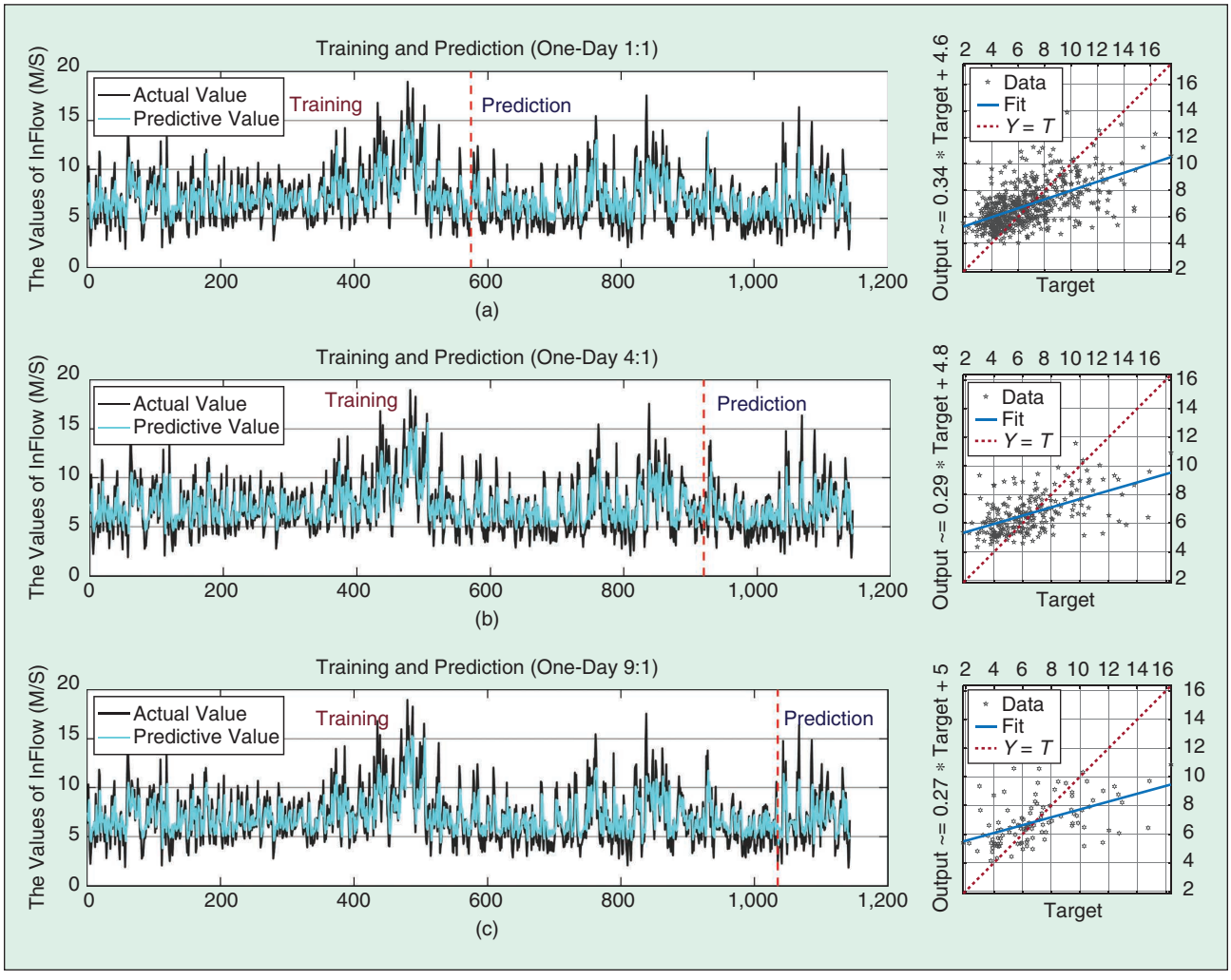

FIGURE 5 Training and prediction results and predicted correlation coefficients of DNR-SMS for the wind speed time series with an interval of one day and partition ratios of (a) 1:1, (b) 4:1, and (c) 9:1.

training samples does not always enhance the prediction performance and that a partition ratio of 4:1 may be a suitable choice. In addition, for a partition ratio of 1:1, SVR-r and SVR-l yielded superior results on both the long-term and short-term wind speed data compared to those obtained using the other algorithms. However, when the partition ratio was 4:1 or 9:1, DNR-SMS achieved the highest prediction performance among the tested algorithms. These findings imply that the training processes for SVR-r and SVR-l require less data than those for other algorithms. Thus, when the length of the wind speed time series is insufficient, SVR-r and SVR-l can provide more satisfactory results. However, if an adequate amount of wind speed time series data is available, DNR can demonstrate highly competitive performance.

To fairly evaluate the performance of each algorithm, several evaluation methods were adopted in this study, such as the evaluation indicators, nonparametric statistical test, and relative graphs.

To identify the significant differences between the compared algorithms, the *p-values* of the Wilcoxon rank sum test are also provided in Tables IV and V. Although SVR-r and SVR-l outperform DNR-SMS for the ratio of 1:1, the corresponding *p-values* are larger than 5%, indicating that the performance is not significantly higher than that of DNR-SMS. However, for ratios of 4:1 and 9:1, DNR-SMS can produce significantly superior results compared to the other algorithms, such as the ENN, the MLP, the various SVR methods and the LSTM network, based on the corresponding *p-values*. Moreover, compared with the prediction performance of the original DNM-BP algorithm and EDNM, that of the proposed DNR model is significantly higher, which implies that considering the strength of each dendritic branch individually can enhance

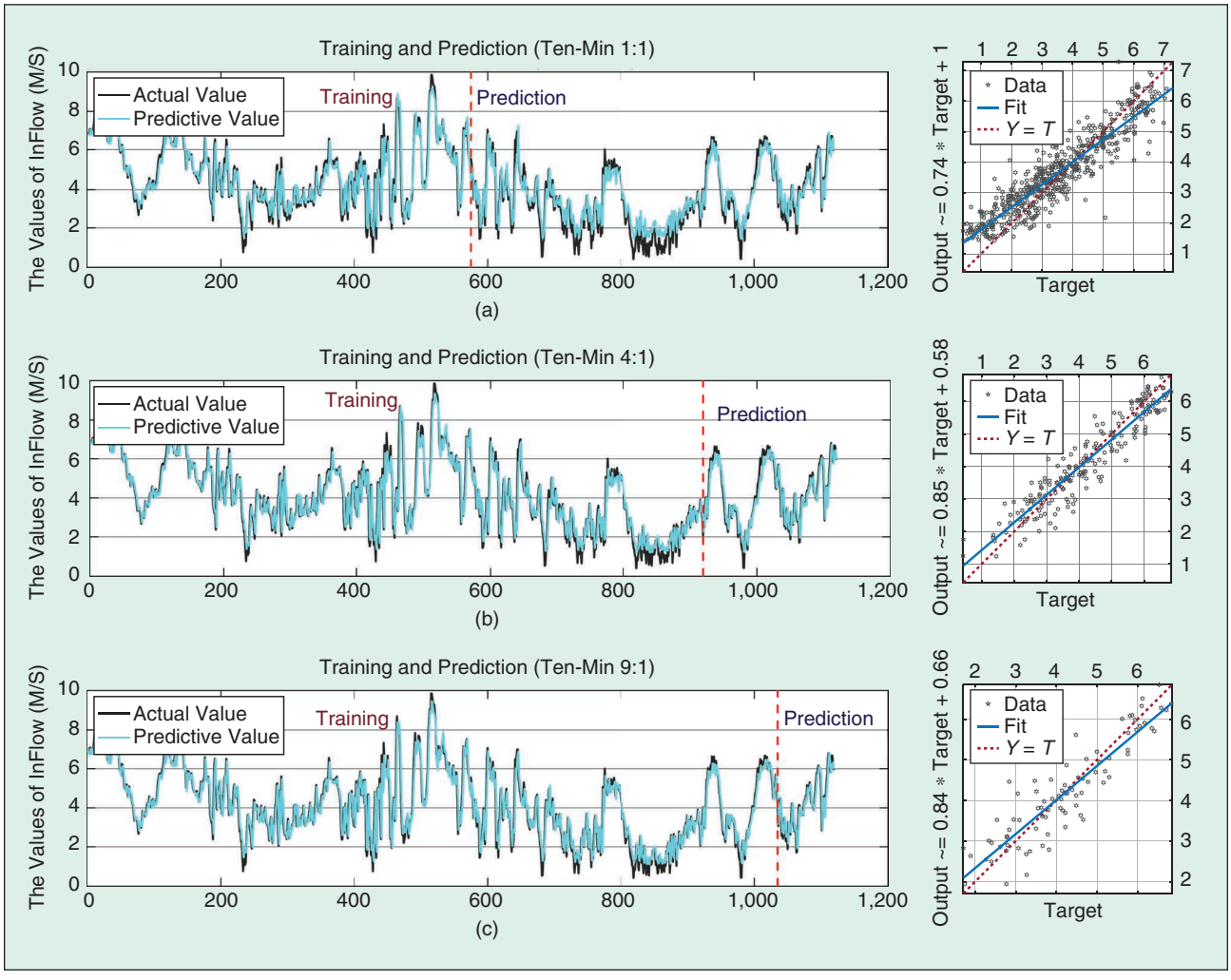

FIGURE 6 Training and prediction results and predicted correlation coefficients of DNR-SMS for the wind speed time series with an interval of ten minutes and partition ratios of (a) 1:1, (b) 4:1, and (c) 9:1.

Experimental results demonstrate that the DNR can provide satisfactory and stable forecasting performances for wind speed time series.

the regression ability. Thus, DNR is more suitable for solving prediction problems than the DNM. In addition, a comparison of DNR-SMS and DNR-BP is presented in Tables IV and V. DNR-SMS is significantly superior to DNR-BP in most cases, suggesting that the SMS algorithm, as a global optimization approach that does not easily become trapped in local minima, can offer enhanced optimization performance compared to that achieved using the BP algorithm.

Furthermore, the convergence curves of DNR-SMS for the wind speed time series are provided in Fig. 4. Although the maximum number of iterations of the SMS algorithm was set to 1000, convergence was attained in less than 200 iterations. Thus, the DNR model trained using the SMS algorithm could achieve prompt convergence, indicating that the SMS algorithm is an efficient optimization tool for real-world prediction problems. In addition, Figures 5 and 6 show the fit graphs and corresponding correlation coefficient graphs of DNR-SMS for long-term and short-term wind speed prediction, respectively. Figures 5 and 6 indicate that long-term wind speed time series prediction is more complex than short-term prediction. Although the forecast curves of DNR-SMS closely fit the actual curves for both long-term and short-term wind speed prediction with different partition ratios, the long-term prediction performance of DNR-SMS is higher than the short-term prediction performance, as indicated by the correlation coefficient graphs. In addition, for both long-term and short-term wind speed prediction, the performance of the DNR model trained using the SMS algorithm was compared with that of DNR models trained using other state-of-the-art evolutionary algorithms, namely, a genetic algorithm (GA), particle swarm optimization (PSO), adaptive differential evolution with an optional external archive (JADE) [61] and differential evolution with success-history-based parameter adaptation and linear population size reduction (L-SHADE) [62]. The experimental results are presented in Tables VI and VII. The SMS algorithm notably outperforms the other algorithms. Consequently, the SMS algorithm is recommended as the main optimization approach for DNR. These experimental results demonstrate that the proposed approach can enable satisfactory and stable forecasting for wind speed time series.

TABLE VI Long-term prediction performance of DNR models trained using different evolution algorithms for the experimental wind speed time series.

LONG-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 1:1							
ALGORITHM	***MSE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***RMSE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***MAE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***R* (MEAN ± STD (OR SD))**
DNR-GA	1.91E-02±1.55E-03	1.01E-06	1.38E-01±5.50E-03	1.01E-06	1.84E+00±1.16E-01	4.89E-06	5.02E-01±5.98E-02
DNR-PSO	1.73E-02±6.64E-04	1.01E-02	1.32E-01±2.50E-03	1.06E-02	1.72E+00±1.73E-02	3.48E-01	5.46E-01±1.68E-02
DNR-JADE	1.74E-02±6.89E-04	9.13E-07	1.32E-01±2.59E-03	9.12E-07	1.73E+00±4.62E-02	9.12E-07	5.46E-01±2.10E-02
DNR-L-SHADE	1.81E-02±2.15E-03	1.46E-02	1.34E-01±7.48E-03	1.54E-02	1.76E+00±1.08E-01	8.41E-02	5.27E-01±6.00E-02
DNR-SMS	**1.69E-02±2.04E-04**	-	**1.30E-01±7.85E-04**	-	**1.72E+00±1.07E-02**	-	**5.53E-01±5.73E-03**
LONG-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 4:1							
ALGORITHM	***MSE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***RMSE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***MAE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***R* (MEAN ± STD (OR SD))**
DNR-GA	1.91E-02±1.43E-03	9.13E-07	1.38E-01±5.03E-03	9.13E-07	1.79E+00±9.18E-02	1.01E-06	4.66E-01±5.43E-02
DNR-PSO	1.74E-02±4.84E-04	1.50E-02	1.32E-01±1.82E-03	1.54E-02	1.69E+00±1.97E-02	6.67E-04	5.06E-01±1.40E-02
DNR-JADE	1.78E-02±1.01E-03	2.65E-05	1.34E-01±3.66E-03	2.54E-05	1.72E+00±3.98E-02	7.13E-06	5.00E-01±3.17E-02
DNR-L-SHADE	1.85E-02±1.66E-03	1.03E-05	1.36E-01±5.93E-03	1.13E-05	1.73E+00±6.58E-02	7.13E-06	4.81E-01±4.08E-02
DNR-SMS	**1.71E-02±1.84E-04**	-	**1.31E-01±7.05E-04**	-	**1.67E+00±9.46E-03**	-	**5.14E-01±6.22E-03**
LONG-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 9:1							
ALGORITHM	***MSE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***RMSE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***MAE* (MEAN ± STD (OR SD))**	***P*-VALUE**	***R* (MEAN ± STD (OR SD))**
DNR-GA	2.46E-02±2.20E-03	2.25E-06	1.57E-01±6.81E-03	2.25E-06	1.97E+00±1.05E-01	3.66E-06	4.34E-01±6.40E-02
DNR-PSO	2.35E-02±2.80E-03	2.27E-03	1.53E-01±8.46E-03	2.27E-03	1.91E+00±6.54E-02	5.51E-05	4.66E-01±5.01E-02
DNR-JADE	2.30E-02±1.19E-03	8.25E-04	1.52E-01±3.91E-03	9.49E-04	1.90E+00±4.89E-02	1.86E-03	4.82E-01±2.57E-02
DNR-L-SHADE	2.35E-02±1.49E-03	2.13E-05	1.53E-01±4.80E-03	2.13E-05	1.90E+00±4.06E-02	6.52E-05	4.67E-01±3.16E-02
DNR-SMS	**2.21E-02±3.55E-04**	-	**1.49E-01±1.20E-03**	-	**1.86E+00±1.35E-02**	-	**4.94E-01±9.41E-03**

TABLE VII Short-term prediction performance of DNR models trained using different evolution algorithms for the experimental wind speed time series.

SHORT-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 1:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
DNR-GA	1.14E-02±4.36E-03	1.12E-06	1.05E-01±1.94E-02	1.12E-06	7.56E-01±1.47E-01	1.24E-06	8.46E-01±7.47E-02
DNR-PSO	1.09E-02±1.51E-02	1.37E-06	9.63E-02±4.08E-02	1.37E-06	6.11E-01±1.49E-01	3.02E-06	8.46E-01±1.52E-01
DNR-JADE	8.55E-03±5.98E-03	2.54E-04	8.91E-02±2.47E-02	2.75E-04	5.99E-01±1.13E-01	2.12E-03	8.80E-01±9.16E-02
DNR-L-SHADE	1.27E-02±1.50E-02	9.13E-07	1.05E-01±4.06E-02	9.13E-07	6.41E-01±1.44E-01	1.12E-06	8.11E-01±1.45E-01
DNR-SMS	**5.43E-03±3.54E-04**	-	**7.36E-02±2.41E-03**	-	**5.24E-01±1.58E-02**	-	**9.21E-01±5.28E-03**

SHORT-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 4:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
DNR-GA	7.04E-03±3.07E-03	1.85E-06	8.20E-02±1.79E-02	1.85E-06	6.31E-01±1.44E-01	2.25E-06	8.80E-01±6.31E-02
DNR-PSO	3.35E-03±5.19E-04	6.15E-02	5.78E-02±4.26E-03	6.40E-02	4.28E-01±2.23E-02	5.00E-01	9.32E-01±9.73E-03
DNR-JADE	3.92E-03±1.38E-03	8.25E-04	6.19E-02±9.57E-03	8.85E-04	4.61E-01±7.11E-02	2.92E-02	9.29E-01±2.09E-02
DNR-L-SHADE	3.66E-03±1.53E-03	3.40E-01	5.96E-02±1.05E-02	3.40E-01	4.21E-01±2.54E-02	9.19E-01	9.27E-01±2.95E-02
DNR-SMS	**3.15E-03±1.50E-04**	-	**5.61E-02±1.32E-03**	-	**4.27E-01±1.14E-02**	-	**9.36E-01±3.12E-03**

SHORT-TERM WIND SPEED TIME SERIES WITH A PARTITION RATIO OF 9:1

ALGORITHM	*MSE* (MEAN ± STD (OR SD))	*P*-VALUE	*RMSE* (MEAN ± STD (OR SD))	*P*-VALUE	*MAE* (MEAN ± STD (OR SD))	*P*-VALUE	*R* (MEAN ± STD (OR SD))
DNR-GA	7.52E-03±2.95E-03	1.24E-06	8.50E-02±1.72E-02	1.24E-06	6.47E-01±1.37E-01	9.13E-07	8.26E-01±8.36E-02
DNR-PSO	3.98E-03±2.90E-03	9.79E-01	6.14E-02±1.46E-02	9.79E-01	4.65E-01±1.19E-01	7.95E-01	8.86E-01±1.06E-01
DNR-JADE	4.17E-03±5.79E-04	3.02E-06	6.44E-02±4.41E-03	3.02E-06	4.80E-01±3.67E-02	2.02E-04	8.95E-01±1.43E-02
DNR-L-SHADE	3.55E-03±1.95E-04	5.25E-01	5.95E-02±1.59E-03	5.08E-01	4.46E-01±1.34E-02	7.64E-01	9.02E-01±6.46E-03
DNR-SMS	**3.50E-03±7.30E-05**	-	**5.91E-02±6.17E-04**	-	**4.46E-01±7.34E-03**	-	**9.03E-01±2.07E-03**

V. Conclusion

A novel DNR method inspired by biological neurons is employed to predict wind speed time series. To improve the prediction results, the neural architecture for DNR is trained using the SMS algorithm, which is a global optimization algorithm with powerful search capabilities. Before the prediction process, the time delay and the number of embedding dimensions are calculated using the MI algorithm and the FNN algorithm, respectively. Based on the results, phase space reconstruction of the wind speed data is performed, and the maximum Lyapunov exponent is calculated to detect the chaotic nature of the wind speed series. Comprehensive experiments conducted to evaluate the effectiveness of the proposed DNR-SMS approach are reported, in which several commonly used prediction approaches are considered for comparison and four diverse evaluation indicators are employed. The experimental results and statistical tests demonstrate that DNR can outperform the other compared approaches. The proposed DNR-SMS model is thus an efficient tool for wind speed prediction due to its high prediction accuracy and robust stability. Further research can be conducted in the following directions. First, other optimization algorithms can be evaluated to enhance the forecasting performance of the DNR model. Second, DNR-SMS can be applied to solve other prediction problems, such as container throughput forecasting and power load prediction.

Acknowledgments

This work was supported by a Project of the Guangdong Basic and Applied Basic Research Fund (No. 2019A1515111139), by the National Natural Science Foundation of China under Grants 61876110 and 61876162, by a Key Program of the Joint Funds of the National Natural Science Foundation of China under Grant U1713212, and in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant PolyU11202418 and Grant PolyU11209219.

References

[1] W. Yang, J. Wang, H. Lu, T. Niu, and P. Du, "Hybrid wind energy forecasting and analysis system based on divide and conquer scheme: A case study in China," *J. Cleaner Production*, vol. 222, pp. 942–959, 2019. doi: 10.1016/j.jclepro.2019.03.036.

[2] C. Tian, Y. Hao, and J. Hu, "A novel wind speed forecasting system based on hybrid data preprocessing and multi-objective optimization," *Appl. Energy*, vol. 231, pp. 301–319, 2018. doi: 10.1016/j.apenergy.2018.09.012.

[3] H. Li, J. Wang, H. Lu, and Z. Guo, "Research and application of a combined model based on variable weight for short term wind speed forecasting," *Renewable Energy*, vol. 116, pp. 669–684, 2018. doi: 10.1016/j.renene.2017.09.089.

[4] M. Imal, M. Sekkeli, C. Yildiz, and F. Kececioglu, "Wind energy potential estimation and evaluation of electricity generation in Kahramanmaras, Turkey," *Energy Educ. Sci. Technol. A, Energy Sci. Res.*, vol. 30, no. 1, pp. 661–672, 2012.

[5] J. Zhao, Z.-H. Guo, Z.-Y. Su, Z.-Y. Zhao, X. Xiao, and F. Liu, "An improved multi-step forecasting model based on WRF ensembles and creative fuzzy systems for wind speed," *Appl. Energy*, vol. 162, pp. 808–826, 2016. doi: 10.1016/j.apenergy.2015.10.145.
[6] C. Y ild iz, M. Tekin, A. Gani, Ö. F. Keçeciog˘lu, H. Aç ikgöz, and M. S¸ekkeli, "A day-ahead wind power scenario generation, reduction, and quality test tool," *Sustainability*, vol. 9, no. 5, p. 864, 2017. doi: 10.3390/su9050864.
[7] D. Wang, H. Luo, O. Grunder, and Y. Lin, "Multi-step ahead wind speed forecasting using an improved wavelet neural network combining variational mode decomposition and phase space reconstruction," *Renewable Energy*, vol. 113, pp. 1345–1358, 2017. doi: 10.1016/j.renene.2017.06.095.
[8] J. Wang, P. Du, T. Niu, and W. Yang, "A novel hybrid system based on a new proposed algorithm–multi-objective whale optimization algorithm for wind speed forecasting," *Appl. Energy*, vol. 208, pp. 344–360, 2017. doi: 10.1016/j.apenergy.2017.10.031.
[9] J. Song, J. Wang, and H. Lu, "A novel combined model based on advanced optimization algorithm for short-term wind speed forecasting," *Appl. Energy*, vol. 215, pp. 643–658, 2018. doi: 10.1016/j.apenergy.2018.02.070.
[10] T. Howard and P. Clark, "Correction and downscaling of NWP wind speed forecasts," *Meteorol. Appl., J. Forecasting, Pract. Appl., Train. Techn. Model.*, vol. 14, no. 2, pp. 105–116, 2007. doi: 10.1002/met.12.
[11] P. Du, J. Wang, W. Yang, and T. Niu, "A novel hybrid model for short-term wind power forecasting," *Appl. Soft Comput.*, vol. 80, pp. 93–106, 2019. doi: 10.1016/j.asoc.2019.03.035.
[12] Y. Wang, J. Wang, G. Zhao, and Y. Dong, "Application of residual modification approach in seasonal ARIMA for electricity demand forecasting: A case study of China," *Energy Pol.*, vol. 48, pp. 284–294, 2012. doi: 10.1016/j.enpol.2012.05.026.
[13] J. Wang, J. Heng, L. Xiao, and C. Wang, "Research and application of a combined model based on multi-objective optimization for multi-step ahead wind speed forecasting," *Energy*, vol. 125, pp. 591–613, 2017. doi: 10.1016/j.energy.2017.02.150.
[14] R. G. Kavasseri and K. Seetharaman, "Day-ahead wind speed forecasting using f-ARIMA models," *Renewable Energy*, vol. 34, no. 5, pp. 1388–1393, 2009. doi: 10.1016/j.renene.2008.09.006.
[15] R. Li and Y. Jin, "A wind speed interval prediction system based on multi-objective optimization for machine learning method," *Appl. Energy*, vol. 228, pp. 2207–2220, 2018. doi: 10.1016/j.apenergy.2018.07.032.
[16] H. Liu, Z. Duan, Y. Li, and H. Lu, "A novel ensemble model of different mother wavelets for wind speed multi-step forecasting," *Appl. Energy*, vol. 228, pp. 1783–1800, 2018. doi: 10.1016/j.apenergy.2018.07.050.
[17] Q. Zhou, C. Wang, and G. Zhang, "Hybrid forecasting system based on an optimal model selection strategy for different wind speed forecasting problems," *Appl. Energy*, vol. 250, pp. 1559–1580, 2019. doi: 10.1016/j.apenergy.2019.05.016.
[18] Y. Hao and C. Tian, "A novel two-stage forecasting model based on error factor and ensemble method for multi-step wind power forecasting," *Appl. Energy*, vol. 238, pp. 368–383, 2019. doi: 10.1016/j.apenergy.2019.01.063.
[19] A. Sfetsos, "A comparison of various forecasting techniques applied to mean hourly wind speed time series," *Renewable Energy*, vol. 21, no. 1, pp. 23–35, 2000. doi: 10.1016/S0960-1481(99)00125-1.
[20] Y. Wang, Y. Yu, S. Cao, X. Zhang, and S. Gao, "A review of applications of artificial intelligent algorithms in wind farms," *Artif. Intell. Rev.*, pp. 1–54, 2019. doi: 10.1007/s10462-019-09768-7.
[21] A. Liu, Y. Xue, J. HU, and L. LIU, "Ultra-short-term wind power forecasting based on SVM optimized by GA," *Power Syst. Protection Control*, vol. 43, no. 2, pp. 90–95, 2015.
[22] X. Kong, X. Liu, R. Shi, and K. Y. Lee, "Wind speed prediction using reduced support vector machines with feature selection," *Neurocomputing*, vol. 169, pp. 449–456, 2015. doi: 10.1016/j.neucom.2014.09.090.
[23] P. Jiang, Y. Wang, and J. Wang, "Short-term wind speed forecasting using a hybrid model," *Energy*, vol. 119, pp. 561–577, 2017. doi: 10.1016/j.energy.2016.10.040.
[24] Z.-h. Guo, J. Wu, H.-y. Lu, and J.-z. Wang, "A case study on a hybrid wind speed forecasting method using bp neural network," *Knowl.-Based Syst.*, vol. 24, no. 7, pp. 1048–1056, 2011.
[25] A. Ahmed and M. Khalid, "An intelligent framework for short-term multi-step wind speed forecasting based on functional networks," *Appl. Energy*, vol. 225, pp. 902–911, 2018. doi: 10.1016/j.apenergy.2018.04.101.
[26] S. P. Kani and M. Ardehali, "Very short-term wind speed prediction: A new artificial neural network–Markov chain model," *Energy Convers. Manage.*, vol. 52, no. 1, pp. 738–745, 2011. doi: 10.1016/j.enconman.2010.07.053.
[27] G. Memarzadeh and F. Keynia, "A new short-term wind speed forecasting method based on fine-tuned LSTM neural network and optimal input sets," *Energy Convers. Manage.*, vol. 213, p. 112,824, 2020.
[28] G. Zhang and D. Liu, "Causal convolutional gated recurrent unit network with multiple decomposition methods for short-term wind speed forecasting," *Energy Convers. Manage.*, vol. 226, p. 113,500, 2020.
[29] J. Ji, S. Gao, J. Cheng, Z. Tang, and Y. Todo, "An approximate logic neuron model with a dendritic structure," *Neurocomputing*, vol. 173, pp. 1775–1783, 2016. doi: 10.1016/j.neucom.2015.09.052.
[30] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, and Z. Tang, "Financial time series prediction using a dendritic neuron model," *Knowl.-Based Syst.*, vol. 105, pp. 214–224, 2016. doi: 10.1016/j.knosys.2016.05.031.
[31] W. Chen, J. Sun, S. Gao, J.-J. Cheng, J. Wang, and Y. Todo, "Using a single dendritic neuron to forecast tourist arrivals to Japan," *IEICE Trans. Inf. Syst.*, vol. 100, no. 1, pp. 190–202, 2017. doi: 10.1587/transinf.2016EDP7152.
[32] J. Ji, S. Song, Y. Tang, S. Gao, Z. Tang, and Y. Todo, "Approximate logic neuron model trained by states of matter search algorithm," *Knowl.-Based Syst.*, vol. 163, pp. 120–130, 2019. doi: 10.1016/j.knosys.2018.08.020.
[33] J. K. Makara, A. Losonczy, Q. Wen, and J. C. Magee, "Experience-dependent compartmentalized dendritic plasticity in rat hippocampal ca1 pyramidal neurons," *Nature Neurosci.*, vol. 12, no. 12, p. 1485, 2009. doi: 10.1038/nn.2428.
[34] T. Elsken, J. H. Metzen, F. Hutter et al., "Neural architecture search: A survey." *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.
[35] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, 2020. doi: 10.1109/TCYB.2020.2983860.
[36] Z. Lu, I. Whalen, Y. Dhebar, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "Multi-objective evolutionary design of deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, 2020. doi: 10.1109/TEVC.2020.3024708.
[37] J. Ji, Y. Tang, L. Ma, J. Li, Q. Lin, Z. Tang, and Y. Todo, "Accuracy versus simplification in an approximate logic neural model," *IEEE Trans. Neural Netw. Learn. Syst.*, 2020.
[38] E. Cuevas, A. Echavarría, and M. A. Ramírez-Ortegón, "An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation," *Appl. Intell.*, vol. 40, no. 2, pp. 256–272, 2014. doi: 10.1007/s10489-013-0458-0.
[39] N. Chouikhi, B. Ammar, N. Rokbani, and A. M. Alimi, "PSO-based analysis of echo state network parameters for time series forecasting," *Appl. Soft Comput.*, vol. 55, pp. 211–225, 2017. doi: 10.1016/j.asoc.2017.01.049.
[40] W. Jia, D. Zhao, Y. Zheng, and S. Hou, "A novel optimized GA–Elman neural network algorithm," *Neural Comput. Appl.*, vol. 31, no. 2, pp. 449–459, 2019. doi: 10.1007/s00521-017-3076-7.
[41] P. Ong and Z. Zainuddin, "Optimizing wavelet neural networks using modified cuckoo search for multi-step ahead chaotic time series prediction," *Appl. Soft Comput.*, vol. 80, pp. 374–386, 2019. doi: 10.1016/j.asoc.2019.04.016.
[42] D. Z. Li, W. Wang, and F. Ismail, "An evolving fuzzy neural predictor for multi-dimensional system state forecasting," *Neurocomputing*, vol. 145, pp. 381–391, 2014. doi: 10.1016/j.neucom.2014.05.014.
[43] T. Desell, S. Clachar, J. Higgins, and B. Wild, "Evolving deep recurrent neural networks using ant colony optimization," in *Proc. Eur. Conf. Evol. Comput. Combinatorial Optimization*. Springer-Verlag, 2015, pp. 86–98.
[44] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*. SpringerVerlag, 1981, pp. 366–381.
[45] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons*. London: Oxford Univ. Press, 1998.
[46] E. Salinas and L. Abbott, "A model of multiplicative neural responses in parietal cortex," *Proc. Nat. Acad. Sci.*, vol. 93, no. 21, pp. 11,956–11,961, 1996. doi: 10.1073/pnas.93.21.11956.
[47] F. Gabbiani, H. G. Krapp, C. Koch, and G. Laurent, "Multiplicative computation in a visual neuron sensitive to looming," *Nature*, vol. 420, no. 6913, pp. 320–324, 2002.
[48] M. B. Kennel, R. Brown, and H. D. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Phys. Rev. A*, vol. 45, no. 6, p. 3403, 1992. doi: 10.1103/PhysRevA.45.3403.
[49] A. M. Fraser and H. L. Swinney, "Independent coordinates for strange attractors from mutual information," *Phys. Rev. A*, vol. 33, no. 2, p. 1134, 1986. doi: 10.1103/PhysRevA.33.1134.
[50] C. Grebogi, E. Ott, and J. A. Yorke, "Crises, sudden changes in chaotic attractors, and transient chaos," *Phys. D, Nonlinear Phenomena*, vol. 7, nos. 1–3, pp. 181–200, 1983. doi: 10.1016/0167-2789(83)90126-4.
[51] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," *Phys. D, Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1985. doi: 10.1016/0167-2789(85)90011-9.
[52] P. Shang, X. Na, and S. Kamae, "Chaotic analysis of time series in the sediment transport phenomenon," *Chaos, Solitons Fractals*, vol. 41, no. 1, pp. 368–379, 2009. doi: 10.1016/j.chaos.2008.01.014.
[53] H. Abarbanel, *Analysis of Observed Chaotic Data*. Springer Science & Business Media, 2012.
[54] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Trans. Nuclear Sci.*, vol. 44, no. 3, pp. 1464–1468, 1997. doi: 10.1109/23.589532.
[55] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecasting*, vol. 14, no. 1, pp. 35–62, 1998. doi: 10.1016/S0169-2070(97)00044-7.
[56] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC–2005 special session on real parameter optimization," *J. Heurist.*, vol. 15, no. 6, p. 617, 2009. doi: 10.1007/s10732-008-9080-4.
[57] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011. doi: 10.1016/j.swevo.2011.02.002.
[58] J. Alcalá-Fdez et al., "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2009. doi: 10.1007/s00500-008-0323-y.
[59] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, 1990. doi: 10.1207/s15516709cog1402_1.
[60] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[61] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, 2009. doi: 10.1109/TEVC.2009.2014613.
[62] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *Proc. 2014 IEEE Congr. Evol. Comput. (CEC)*, pp. 1658–1665. doi: 10.1109/CEC.2014.6900380.