*energies*

MDPI

*Article*

# Application of Augmented Echo State Networks and Genetic Algorithm to Improve Short-Term Wind Speed Forecasting

**Hugo T. V. Gouveia [1], Murilo A. Souza [1], Aida A. Ferreira [2], Jonata C. de Albuquerque [1], Otoni Nóbrega Neto [1], Milde Maria da Silva Lira [1] and Ronaldo R. B. de Aquino [1,*],**

[1] Department of Electrical Engineering, Federal University of Pernambuco, Recife 50670-901, Brazil; hugo.gouveia@ufpe.br (H.T.V.G.); murilo.asouza@ufpe.br (M.A.S.); aidaferreira@recife.ifpe.edu.br (A.A.F.); jonata.albuquerque@ufpe.br (J.C.d.A.); otoni.nobrega@ufpe.br (O.N.N.); milde@ufpe.br (M.M.d.S.L.)
[2] Department of Electrical Systems, Federal Institute of Pernambuco, Recife 50740-545, Brazil
* Correspondence: ronaldo.aquino@ufpe.br

**Abstract:** The large-scale integration into electrical systems of intermittent power-generation sources, such as wind power plants, requires greater efforts and knowledge from operators to keep these systems operating efficiently. These sources require reliable output power forecasts to set up the optimal operating point of the electrical system. In previous research, the authors developed an evolutionary approach algorithm called RCDESIGN to optimize the hyperparameters and topology of Echo State Networks (ESN), and applied the model in different time series forecasting, including wind speed. In this paper, RCDESIGN was modified in some aspects of the genetic algorithm, and now it optimizes an ESN with augmented states (ESN-AS) and has been called RCDESIGN-AS. The evolutionary algorithm allows the search for the best parameters and topology of the recurrent neural network to be performed simultaneously. In addition, RCDESIGN-AS has the important characteristic of requiring little computational effort and processing time since it is not necessary for the eigenvalues of the reservoir weight matrix to be reduced and also due to the fact that the augmented states make it possible to reduce the number of neurons in the reservoir. The method was applied for wind speed forecasting with a 24-h ahead horizon using real data of wind speed from five cities in the Northeast Region of Brazil. All results obtained with the proposed method overcame forecasting performed by the persistence method, obtaining prediction gains ranging from 60% to 80% in relation to this reference method. In some datasets, the proposed method also yielded better results than the traditional ESN, showing that RCDESIGN-AS can be a powerful tool for wind-speed forecasting and possibly for other types of time series.

**Keywords:** artificial neural networks; forecasting; genetic algorithms; time series analysis; wind energy

check for updates

**Citation:** Gouveia, H.T.V.; Souza, M.A.; Ferreira, A.A.; de Albuquerque, J.C.; Nóbrega Neto, O.; da Silva Lira, M.M.; de Aquino, R.R.B. Application of Augmented Echo State Networks and Genetic Algorithm to Improve Short-Term Wind Speed Forecasting. *Energies* **2023**, *16*, 2635. https://doi.org/10.3390/en16062635

Received: 6 January 2023
Revised: 28 February 2023
Accepted: 8 March 2023
Published: 10 March 2023

**Copyright:** © 2023 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

Wind energy is currently one of the renewable energy sources with the highest growth rates in the energy mix of several countries. Thus, the smart integration of wind farms into electrical power systems is currently a challenge for many of these countries since it is a source that is intrinsically dependent on specific climatic conditions for the production of electricity. In other words, the availability of wind power is almost entirely linked to the availability of wind. Power systems are subject to greater unpredictable variations in power flows as the installed capacity of this type of intermittent generation increases. The problem is that, in these systems, generation must always be equal to demand. Therefore, as a direct consequence of the increased penetration level of wind generation, there is a need for new methods for balancing generation and electrical demand [1].

Real-time operation requires short-term forecasts of wind generation in the order of seconds, minutes and a few hours, as well as the integration of these forecasts in centralized system controls [1].

For wind generation forecasting, the short-term horizon, more specifically, the next day horizon, will always be of great relevance. Most of the operators of the power systems carry out the programming of the unit synchronism for this horizon. Most traders also focus on the daily market for energy trading and ancillary services. The one-day ahead horizon is characterized by the definition of system operation planning for the next day, including the selection of large thermoelectric plants that can take several hours to start so that they will be able to meet the demand when needed. The importance of using good wind generation forecasts for the next day's timing planning has been widely discussed and incorporated into most power systems, as the consequence of disregarding wind generation forecasts may be non-optimized planning [2].

The main proposed models in the literature for wind energy forecasting are mostly divided into physical, statistical and artificial intelligence models [3,4]. Time series forecasting using statistical methods can be applied for wind forecasting. However, most of these models assume linearity and stationary conditions over the time series, and these constraints are not always met with actual wind energy data [5,6].

Artificial Intelligence (AI) techniques such as support vector machines, random forests, clusters, fuzzy logic and artificial neural networks (ANN) have been used more frequently to perform wind power forecasting in recent years. Particularly, ANN showed good performance, surpassing several cutting-edge techniques [7]. However, most of these models use univariate time series, considering only endogenous variables [8,9]. Deep learning (DL) methods have also been used for wind generation forecasting [10]. These models are formed by multiple hidden layers, and because of that, they are able to extract more hidden features from the data, usually yielding better performance in the prediction task than traditional machine learning methods. In addition to data from the target wind station, [11] uses meteorological data from additional neighboring stations as input to a mixed-input feature ANN to improve the forecast performance. In [12], two DL networks (Long Short-Term Memory and Convolutional Neural Network) are applied to perform a short-term wind power forecast. The results of these networks are combined through an induced ordered weighted average operator (IOWA), which assigns weights to each forecasting network by the level of its forecasting accuracy at each time point.

Recurrent Neural Networks (RNN) are a specific class of ANN that is gaining attention because they can model the time dynamics of a series in a natural way [13]. At each step in time, an input vector activates the network by updating its hidden state through activation functions and uses it to predict its output. Its nonlinear dynamics allow modeling and predicting highly complex time series. It is important to observe the long time consumed in the modeling process and the large amount of data required in the training process. The ESN belongs to a paradigm of RNNs, known as Reservoir Computing (RC), where the weights in the hidden layer ("reservoir") are not trainable.

An Echo State Network (Figure 1) consists of a large, fixed, recurrent "reservoir" network from which the desired output is obtained by training suitable output connection weights [14].

In its classical approach, the reservoir is a large RNN containing hundreds of artificial neurons whose connection weights are not modified by training. Only the weights of the reservoir connections to the output are adjusted by linear regression. Jaeger [14] described how an augmented ESN can be conjoined with the "recursive least squares" (RLS) algorithm, a method for fast online adaptation known from linear systems. RCDESIGN (Reservoir Computing and Design Training) is a method that combines an evolutionary algorithm with RC and, at the same time, finds the best values of parameters, topology and weight matrices without rescaling the reservoir matrix by its spectral radius; it was proposed by Ferreira, Ludermir and Aquino [15].

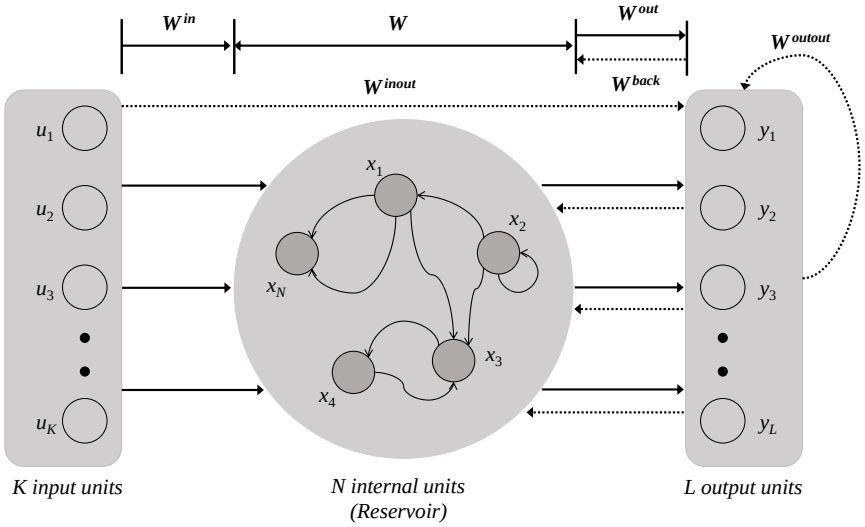


**Figure 1.** An Echo State Network. Solid arrows: fixed weights; dashed arrows: trainable weights. Adapted from Ferreira, Ludermir and Aquino [15].

This paper addresses the problem of forecasting average hourly wind speeds within a 24-h ahead horizon for five different locations in northeastern Brazil. Models created in this work are based in Gouveia [16], which applied RCDESIGN-AS to adjust the parameters of ESN-AS with linear (*identity*) activation functions in reservoir neurons. The promising results of the proposed method are compared with those derived from the forecasting made with the *Persistence Method* (as reference) and also with the classical ESN approach.

The subsequent sections of this work are divided according to the following order: Section 2 presents an overview of Reservoir Computing; Section 3 provides a detailed description of the evolutionary strategy for RCDESIGN-AS; Section 4 describes the Persistence Method; Section 5 shows statistics of the data sets used in this paper; Sections 6 and 7 present the results and conclusions, respectively.

## 2. Reservoir Computing

In RC, the reservoir is a kind of RNN that receives inputs that vary with time and performs neural computation. Conceptually, forming the output layer from a reservoir is a non-temporal supervised mapping task from $x(t)$ to $y_{target}(t)$, which is a well-researched domain in machine learning, and there is a wide variety of methods available that can be applied to solve this problem [15].

### *2.1. General Description of Echo State Networks*

In this paper, an ESN is used as a learning system for time series forecasting. For the sake of a simple notation, we address only single-input, single-output systems, as described by Jaeger [17]. The input and output at discrete-time $t \geq 1$ are scalars $u(t)$ and $y(t)$, respectively. The state of reservoir neurons is the $N$-element column vector $\mathbf{x}(t) = (x_1(t), ..., x_N(t))$. The weights of internal reservoir connections are stored in a square matrix $\mathbf{W} = (w_{ij})$ of order $N$. The weights of connections between $u(t)$ and $y(t)$ with the $N$ reservoir neurons are stored in column vectors $\mathbf{w^{in}} = (w_i^{in})$ and $\mathbf{w^{back}} = (w_i^{back})$, respectively. In addition to these, vectors $\mathbf{w^{bias}} = (w_i^{bias})$ and $\mathbf{v}(t)$ (random noise) are also stored, all of which are applied for updating the reservoir state according to (1).

The weight of connection going from $u(t)$ to $y(t)$ is a scalar $w^{inout}$, and connections from the reservoir to $y(t)$ are stored in $N$-element column weight vector $\mathbf{w^{out}} = (w_i^{out})$. In addition to these, scalars $w^{outout}$ and $w^{biasout}$ are also stored, all of which are applied to update the output according to (2).

$$\mathbf{x}(t+1) = \mathbf{f}((1-\alpha)\mathbf{x}(t) + \alpha(\mathbf{w^{in}}u(t+1) + \mathbf{W}\mathbf{x}(t) + \mathbf{w^{back}}y(t+1) + \mathbf{w^{bias}} + \mathbf{v}(t+1))), \quad (1)$$

$$y(t+1) = \mathrm{f}^{\mathrm{out}}(\mathrm{w}^{\mathrm{inout}}u(t+1) + (\mathbf{w^{out}})^T\mathbf{x}(t+1) + \mathrm{w}^{\mathrm{outout}}y(t) + \mathrm{w}^{\mathrm{biasout}}), \quad (2)$$

where **f** stands for an element-wise application of neuron activation functions, which can be linear (*identity*) or nonlinear (*hyperbolic tangent*); $\alpha$ is a scalar parameter called *leak rate* used for adjusting dynamics of the reservoir, as described by Lukosevicius [18]; $\mathrm{f}^{\mathrm{out}}$ is the output activation function (*identity*). In (2), superscript $\cdot^T$ denotes transpose.

Training is performed to adjust only the weights of connections to the output ($\mathrm{w}^{\mathrm{inout}}$, $\mathbf{w^{out}}$, $\mathrm{w}^{\mathrm{outout}}$, $\mathrm{w}^{\mathrm{biasout}}$), while the weights of connections to the reservoir ($\mathbf{w^{in}}$, $\mathbf{W}$, $\mathbf{w^{back}}$ and $\mathbf{w^{bias}}$) are fixed and defined before training. The matrix $\mathbf{W}$ typically has sparse connectivity.

### *2.2. Supervised Readout Training for Classical ESN Approach*

From this section, for the remainder of this article, we call $\mathbf{w_{out}}$ the vector formed by the row-wise concatenation [$\mathrm{w}^{\mathrm{inout}}$, $(\mathbf{w^{out}})^T$, $\mathrm{w}^{\mathrm{outout}}$, $\mathrm{w}^{\mathrm{biasout}}$]. Therefore, $\mathbf{w_{out}}$ is the readout vector, which has a number of elements as $\tilde{N} = N + 3$.

Supervised readout training is nothing more than solving a quadratic error minimization problem between the ESN outputs and the desired outputs (targets). The amount of readout weights is typically less than the number of available linear equations, so it is usual to use Linear Regression to solve this type of over-determined system of linear equations. The theory of ESNs can be found in Jaeger [17]. A similar idea has been independently investigated under the name of "liquid state networks" [19].

The signals from input ($u$) and output ($y_{target}$) used during classical ESN training are stationary. When updated according to (1), the network state may become asymptotically independent of initial conditions. This means that regardless of its initial state if the network is updated with the same input sequence in both cases, the resulting state sequences converge to each other, that is $\mathbf{x}(t) \to \tilde{\mathbf{x}}(t)$. If this condition holds, the reservoir state will asymptotically depend only on the input history, and the network is called an Echo State Network. A sufficient condition for the echo state property is the contractivity of $\mathbf{W}$ [17].

A direct method that uses the Moore–Penrose pseudo-inverse to adjust the output weights so that the target output is approximated by the ESN was proposed by Jaeger [20]. ESN training is carried out according to the following steps:

1. Create $\alpha$, $\mathbf{w^{in}}$, $\mathbf{w^{back}}$, $\mathbf{w^{bias}}$ and $\mathbf{v}$;
2. Define $\mathbf{W}$ and then scale it to ensure that its spectral radius is less than 1 ($|\lambda_{max}| < 1$);
3. Run the RNN by driving it with the teaching input signal. Dismiss data from initial transient and collect remaining input and network states row-wise into a $P \times (N+3)$ matrix $\mathbf{M}$, as shown below:

$$\mathbf{M} = \begin{bmatrix} u(1) & \mathbf{x}^T(1) & y(0) & 1 \\ u(2) & \mathbf{x}^T(2) & y(1) & 1 \\ \vdots & \vdots & \vdots & \vdots \\ u(P) & \mathbf{x}^T(P) & y(P-1) & 1 \end{bmatrix};$$

4. In parallel, collect the remaining training pre-signals into a column vector $\mathbf{r}$, as shown below:

$$\mathbf{r} = \begin{bmatrix} (f^{out})^{-1}y_{target}(1) \\ (f^{out})^{-1}y_{target}(2) \\ \vdots \\ (f^{out})^{-1}y_{target}(P) \end{bmatrix};$$

5. Compute the pseudo-inverse $\mathbf{M}^+$, and then calculate $\mathbf{w_{out}} = (\mathbf{M}^+\mathbf{r})^T$;
6. Write $\mathbf{w_{out}}$ into the output connections; the ESN is now trained.

### *2.3. Increasing the Power of ESNs with Augmented States*

The modeling power of an ESN improves when the reservoir size increases. Jaeger [14,20] mentions that a simple way to increase the power of the ESN with little computational effort is to use additional nonlinear transformations of the state $\mathbf{x}(t)$, using a quadratic representation of the reservoir states, which we call ESN-AS. For these networks, the internal units update (1) is unchanged, but there is a slight modification in the method for calculating the output signal update, that is accomplished with the following variant of (2):

$$\begin{aligned} y(t+1) = \mathrm{f}^{\mathrm{out}}(\mathrm{w}^{\mathrm{inout}}u(t+1) + (\mathbf{w^{out}})^T\mathbf{x}(t+1) + \\ \mathrm{w}^{\mathrm{outout}}y(t) + \mathrm{w}^{\mathrm{biasout}} + \mathrm{w}^{\mathrm{in2out}}u^2(t+1) + \\ (\mathbf{w^{out2}})^T\mathbf{x}^2(t+1) + \mathrm{w}^{\mathrm{out2out}}y^2(t) + \mathrm{w}^{\mathrm{bias2out}}), \end{aligned} \tag{3}$$

where $\mathrm{w}^{\mathrm{in2out}}$, $\mathrm{w}^{\mathrm{out2out}}$ and $\mathrm{w}^{\mathrm{bias2out}}$ are scalar weights; $u^2$ and $y^2$ represent the square of the input and output signals, respectively; $\mathbf{w^{out2}} = (w_i^{out2})$ and $\mathbf{x}^2 = (x_i^2)$ are both column vectors with $N$ elements each, where the elements of the latter are equal to the respective elements of $\mathbf{x}$ squared.

According to the modifications described above for updating the ESN-AS output, the readout vector is also modified, being represented by the row-wise concatenation $\mathbf{w_{out-sq}}$ = [$\mathrm{w}^{\mathrm{inout}}$, $(\mathbf{w^{out}})^T$, $\mathrm{w}^{\mathrm{outout}}$, $\mathrm{w}^{\mathrm{biasout}}$,$\mathrm{w}^{\mathrm{in2out}}$, $(\mathbf{w^{out2}})^T$, $\mathrm{w}^{\mathrm{out2out}}$, $\mathrm{w}^{\mathrm{bias2out}}$]. Therefore, $\mathbf{w_{out-sq}}$ is the readout vector, the number of elements in which is $\hat{N} = 2N + 6$. The adjustment of the readout weights for an ESN-AS is carried out according to the following steps:

1. (unchanged);
2. Define $\mathbf{W}$ (with RCDESIGN-AS there is no need to scale the matrix to ensure that the spectral radius is less than 1. Thus a smaller amount of calculations is required, resulting in less computational effort);
3. Run the RNN by driving it with the teaching input signal. Dismiss data from initial transient and collect the remaining input and network states row-wise into a $P \times (2N + 6)$ matrix $\mathbf{M_{sq}}$, which is formed by the horizontal concatenation of matrices $\mathbf{M}$ and $\mathbf{M^2}$ (the elements of which are equal to the respective elements of $\mathbf{M}$ squared);
4. (unchanged);
5. Compute the pseudo-inverse $(\mathbf{M_{sq}})^+$, and then calculate $\mathbf{w_{out-sq}} = ((\mathbf{M_{sq}})^+\mathbf{r})^T$ (RCDESIGN-AS also considers the use of Ridge Regression as an alternative for readout training);
6. Write $\mathbf{w_{out-sq}}$ into the output connections; the ESN-AS is now trained.

### *2.4. Alternatives for Training Readout*

Readout training is generally a procedure that has good numerical stability but requires large hardware memory capacity, especially with the increase in the dimensions of the state matrix ($\mathbf{M}$). Thus, an alternative to work around this possible problem is to limit the number of neurons in the reservoir and/or the number of ESN training patterns. According to Lukosevicius [18], another way to get around the issue is to formulate the problem as the following system of normal equations:

$$\mathbf{w_{out}}\mathbf{M}^T\mathbf{M} = \mathbf{r}^T\mathbf{M}. \tag{4}$$

A natural solution for the system of normal equations in (4) would be:

$$\mathbf{w_{out}} = \mathbf{r}^T\mathbf{M}(\mathbf{M}^T\mathbf{M})^{-\mathbf{1}}. \tag{5}$$

The readout training method using (5) has less numerical stability when compared to the classical approach that uses the pseudo-inverse of the state matrix. According to Lukosevicius [18], this stability problem can be mitigated by using $(\mathbf{M}^T\mathbf{M})^+$ instead of $(\mathbf{M}^T\mathbf{M})^{-1}$, which can provide even faster calculations. The system of normal equations also allows the use of Ridge Regression according to the following equation:

$$\mathbf{w_{out}} = \mathbf{r}^T\mathbf{M}(\mathbf{M}^T\mathbf{M} + \lambda^2\mathbf{I})^{-1}, \tag{6}$$

where the scalar $\lambda$ is the regularization parameter of Ridge Regression, and $\mathbf{I}$ is the identity matrix, which is equal to the number of columns in matrix $\mathbf{M}$.

In addition to improving the numerical stability for solving the optimization problem, the regularization parameter $\lambda$ contributes to reducing the magnitude of the readout vector ($\mathbf{w_{out}}$), mitigating sensitivity to noise, and preventing overfitting.

## 3. Creating Models

The adjustment of the best parameters for an ESN is usually performed through an exhaustive search or through random experiments, which makes this process of defining parameters inefficient. Jaeger [21] described tuning strategies for the three most important global control parameters (network size, spectral radius of reservoir weight matrix and scaling of input). Bayesian Optimization (BO) was used to robustly tune the hyperparameters of an ESN [22]. The *robustness* is achieved when the model performs consistently between validation and test sets. In [23], the reservoir size, spectral radius, and leak rate of an ESN are optimized by covariance matrix adaption evolutionary strategy (CMA-ES). The results obtained in the CMA-ES were compared with the traditional ESN, a LSTM network, a GA-ESN (ESN that was optimized by genetic algorithm) and a feedforward neural network, and it outperformed all of those models.

The evolutionary algorithms are very efficient in searching for optimal solutions to a wide variety of problems because they do not impose many of the limitations found in traditional methods [24]. Thus, we used RCDESIGN-AS as an evolutionary method to optimize the choice of the best parameters of the networks (ESN and ESN-AS) for wind speed forecasting.

The fitness function of RCDESIGN-AS considers the performance of the networks for both sets, training and validation, seeking to obtain networks with better generalization capability and reduced chances of overfitting [15]. The fitness function is calculated as follows:

$$fitness = \overline{MAE}_{Train} + ||\overline{MAE}_{Train} - \overline{MAE}_{Valid}||, \tag{7}$$

where *fitness* is the value to be minimized by the evolutionary algorithm, and the MAE (Mean Absolute Error) is calculated as in (8). Since RCDESIGN-AS uses *10-fold cross-validation*, $\overline{MAE}_{Train}$ is the average of MAE in the training set, and $\overline{MAE}_{Valid}$ is the average of MAE in the validation set.

$$\text{MAE} = \frac{1}{D}\sum_{i=1}^{D}|y(i) - y_{target}(i)|, \tag{8}$$

where $D$ is the total number of data in the set, $y(i)$ and $y_{target}(i)$ are actual and desired (target) outputs of the network, respectively.

The evolutionary algorithm performs an iterative method in which a set of solutions (individuals) passes through selection and reproduction processes. The algorithm creates a new generation of individuals (called children) from the selection of individuals from the current population (called parents). RCDESIGN-AS uses three different ways to produce the children of the next generation: *Elite*, *Crossover* and *Mutation*. Parents with better fitness values have their genetic codes fully copied for their children. One part of the current generation is selected by the *stochastic uniform selection* method [25] to produce children by crossing the parents' genetic codes. The genomes of the remaining parents are copied and

some genes can suffer a random change (with a very low probability) to produce children with mutations.

The population $P^k$ is represented by a group of vectors $\mathbf{c^i}$, where $k$ represents a certain generation of the evolutionary process and $\mathbf{c^i}$ represents a particular individual in the population. The value of $k$ will be limited to the maximum number of generations ($N_G$), and the size of set $P^k$ is defined by parameter $N_I$ of the algorithm, which indicates the number of individuals in the population. Considering that the notation $c_j^i$ symbolizes a characteristic (gene) $j$ for the individual denoted as $i$, we have the following conceptual division of $\mathbf{c^i}$, as shown in Figure 2.

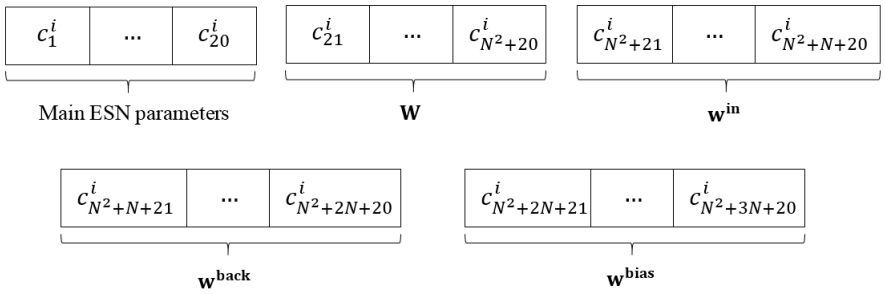


**Figure 2.** Conceptual division of $\mathbf{c^i}$. Adapted from Ferreira, Ludermir and Aquino [15].

A detailed description of the genetic representation of the solutions is presented below:

- $c_1^i$—Defines the number of neurons $(N)$ in the reservoir and, consequently, the sizes of $\mathbf{w^{in}}$, $\mathbf{W}$, $\mathbf{w^{back}}$ and $\mathbf{w^{bias}}$, varying between [10, 50];
- $c_2^i$—If 1, there is a connection between input and output through $w^{inout}$, if 0, there is no connection;
- $c_3^i$—If 1, there is a feedback connection between output and itself through $w^{outout}$, if 0, there is no connection;
- $c_4^i$—If 1, there is a *bias* connected to output through $w^{biasout}$, if 0, there is no connection;
- $c_5^i$—If 1, there are feedback connections between output and reservoir through $\mathbf{w^{back}}$, if 0, there are no connections;
- $c_6^i$—If 1, there are *biases* connected to the reservoir through $\mathbf{w^{bias}}$, if 0, there are no connections;
- $cs_7^i$—Defines the density of reservoir connections, that is, the amount of non-null elements in $\mathbf{W}$, varying between [0.01, 0.1];
- $c_8^i$—Defines the density of connections between the input and reservoir, that is, the amount of non-null elements in $\mathbf{w^{in}}$, varying between [0.01, 1];
- $c_9^i$ —Defines the density of feedback connections between the output and reservoir, that is, the amount of non-null elements in $\mathbf{w^{back}}$, varying between [0.01, 1];
- $c_{10}^i$—Scaling factor applied to the input signal, between [1, 10];
- $c_{11}^i$—Scaling factor applied to the feedback between the output and reservoir, varying between [0, 10];
- $c_{12}^i$—Scaling factor applied to the *bias* in the reservoir, between [0, 10];
- $c_{13}^i$—Defines the maximum value of random noise ($\mathbf{v}$) applied to reservoir states, varying between $[0, 10^{-8}]$;
- $c_{14}^i$—If 1, the reservoir activation functions are *tanh()*, if 0, the activation functions are *identity*;
- $c_{15}^i$—If 1, the readout training function is *pseudo–inverse*, if 0, it is *ridge regress*;
- $c_{16}^i$—Leak rate ($\alpha$), varying between [0.1, 1];
- $c_{17}^i$—Regularization parameter ($\lambda$), varying between $[10^{-8}, 10^{-1}]$;
- $c_{18}^i$—If 1, augmented states are used (ESN-AS), if 0, natural states are used (classical ESN);
- $c_{19}^i$—reserved for future use;

- $c^i_{20}$—reserved for future use;
- $c^i_{21}$ ... $c^i_{(N^2+3N+20)}$ are the weights of **W**, $\mathbf{w^{in}}$, $\mathbf{w^{back}}$ and $\mathbf{w^{bias}}$, varying between $[-1, 1]$.

The pseudo-code used by RCDESIGN in its evolutionary process of searching for the best global parameters, topology and reservoir weights at the same time is presented in Ferreira, Ludermir and Aquino [15].

The operator of elitism is nothing more than the replication of the genetic trait of individuals with the best skills between consecutive generations. The number of new individuals produced by elitism is defined according to the elite number parameter.

An adaptation of the *uniform crossover* for populations of individuals with different sizes of genomes is used by the crossover operator in RCDESIGN-AS. For a given couple of parents, the individual who has the largest genome defines the vector $\mathbf{c^a}$, and the smallest one, $\mathbf{c^b}$. The genome of child $\mathbf{c^f}$ is created by crossing the parents' genomes based on a mask, **m**, that indicates which $\mathbf{c^f}$ characteristics will be inherited from $\mathbf{c^a}$ or $\mathbf{c^b}$. The process begins by crossing the first 20 genes of the parents (which represent the main characteristics of the ESN). Next, a mask (vector with 20 elements) of randomly generated binary numbers is created, indicating which genes will be inherited by the child, as shown in Figure 3.

| | | | | | |
|---|---|---|---|---|---|
| $\boldsymbol{c^a}$ | $c^a_1$ | $c^a_2$ | ... | $c^a_{19}$ | $c^a_{20}$ |
| $\boldsymbol{c^b}$ | $c^b_1$ | $c^b_2$ | ... | $c^b_{19}$ | $c^b_{20}$ |
| $\boldsymbol{m}$ | 0 | 1 | ... | 1 | 0 |
| $\boldsymbol{c^f}$ | $c^b_1$ | $c^a_2$ | ... | $c^a_{19}$ | $c^b_{20}$ |

**Figure 3.** Operator crossing in the first part of the genome. Adapted from Ferreira, Ludermir and Aquino [15].

Then, the crossing is performed to define the second part of the genome (which corresponds to the weights of the matrix **W**). If $c^f_1 = c^b_1$, create mask **m** with size $(c^b_1)^2$ and the crossover operator will combine $\mathbf{c^a}$ and $\mathbf{c^b}$ from gene 21 to gene $(c^b_1)^2 + 20$. If $c^f_1 = c^a_1$, a mask with size $(c^a_1)^2$ is created, with the first $(c^b_1)^2$ elements being random binary values and with the remaining elements equal to 1. That is, the crossing operator will combine $\mathbf{c^a}$ and $\mathbf{c^b}$ from gene 21 to gene $(c^b_1)^2 + 20$ and repeat the genes from $c^a$ from positions $(c^b_1)^2 + 21$ to $(c^a_1)^2 + 20$. The crossing operator acts identically to the parts of the genome corresponding to the weights of $\mathbf{w^{in}}$, $\mathbf{w^{back}}$ and $\mathbf{w^{bias}}$. A scheme of this procedure applied to the second part of the genome is shown in Figure 4.

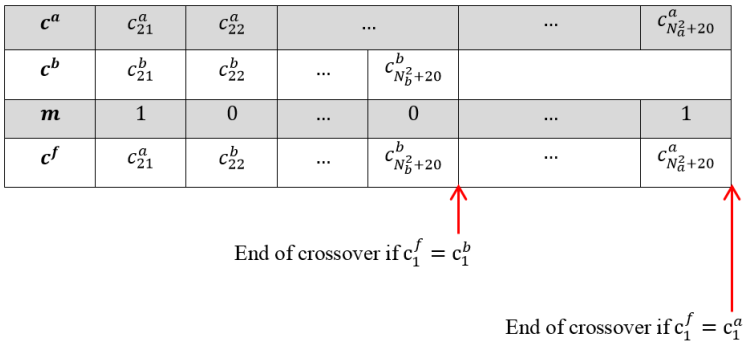


**Figure 4.** Operator crossing in the second part of the genome. Adapted from Ferreira, Ludermir and Aquino [15].

As individuals in the population have genomes with different sizes, the mutation operator used by RCDESIGN-AS is also adapted. For each future child susceptible to genetic mutation with a parent $\mathbf{c^i}$, the gene $c^i_1$ is copied to $c^f_1$ (which means that the number

of neurons in the reservoir represented by the child's first gene is identical to that of one of their parent's). The mutation can occur with a probability defined by the parameter $M_T$ in any gene of the child, except for the first one.

Randomized experiments were conducted to define the parameters of genetic algorithm used by RCDESIGN-AS. The configuration parameters that showed a lower average error in the experimental tests were chosen. In our experiments, the maximum number of generations is 10, the population size is 60, the number of elite individuals is 2, the crossover fraction is 0.8, and the remaining individuals are the mutation children (with a probability of mutation equal to 0.05).

## 4. Persistence Method

The persistence forecasting method was used to compare the results obtained by the forecasting models created with ESN and ESN-AS. This method is premised on the fact that the conditions on the forecast horizon will not change. The persistence forecasting method tends to be used as a benchmark to serve as a reference for comparison with all other forecasting models [26].

## 5. Data Sets

The experiments covered in this article were based on the application of the *Persistence Method* and the RCDESIGN-AS to carry out forecasts of average hourly wind speeds for five cities in northeastern Brazil: Triunfo and Belo Jardim data are from the project SONDA [27]; Macau, Mossoro and Natal data are from National Meteorological Institute (INMET) automatic surface stations [28].

In Table 1, we present the periods and the total samples of the average hourly wind speed series used in this article. The minimum, maximum, means, standard deviations and measurement heights of the wind speed series are shown in Table 2.

**Table 1.** Wind Speed Series.

| Data Set | Initials | Source | Period | Size |
|---|---|---|---|---|
| Triunfo | TRI | SONDA | 2004 to 2005 | 13,176 |
| Belo Jardim | BJD | SONDA | 2004 to 2006 | 21,936 |
| Macau | MAC | INMET | 2008 to 2009 | 16,800 |
| Mossoro | MOS | INMET | 2008 to 2009 | 13,200 |
| Natal | NAT | INMET | 2008 to 2010 | 13,152 |

**Table 2.** Wind Speed Series Statistics.

| Initials | $v_{min}$ (m/s) | $v_{max}$ (m/s) | $v$ (m/s) | $\sigma$ (m/s) | *Height* (m) |
|---|---|---|---|---|---|
| BJD | 0 | 12.95 | 5.1 | 1.79 | 50 |
| TRI | 0 | 37.0 | 11.97 | 5.13 | 50 |
| MAC | 0.2 | 11.6 | 4.48 | 1.99 | 10 |
| MOS | 0.1 | 9.4 | 3.32 | 1.95 | 10 |
| NAT | 0.09 | 10.50 | 4.70 | 1.81 | 10 |

Before creating the models, the time series were normalized for the interval [0, 1] according to:

$$d_{norm}(t) = \frac{d(t) - d_{min}}{d_{max} - d_{min}}, \tag{9}$$

where $d_{norm}(t)$ is the normalized value of $d(t)$; $d_{min}$ and $d_{max}$ are the minimum and maximum values of the time series **d**, respectively.

We used 75% of the normalized data for training and validation, and the remaining 25% percent was used for the tests. To perform the *10-fold cross-validation*, the data were

split into 10 chunks. Each one was used separately as a validation set, while the rest was used for training the readout.

It is worth mentioning that the test set for each series was exactly the same in order to allow a correct comparison between the performance of the different forecasting models.

## 6. Results

In this section, we present the results of different models for wind speed forecasting. We used the *Persistence Method* as a reference for comparison with the ESN (as presented in [15]) and ESN-AS (created with RCDESIGN-AS) models. There are also comparisons with results from the WTESN model, proposed by Gouveia [29].

The simulations were performed on a notebook with a 64-bit Intel (R) Core (TM) 2 Duo CPU T6600 @ 2.20 GHz-2.20 GHz processor, 4 GB of RAM and 64-bit Windows 7 operating system, using the software MATLAB® in version 8.6.0.267246 (R2015b).

We used MAE (mean absolute error, defined in (8) previously), MSE (mean squared error), NMSE (normalized mean squared error), NRMSE (normalized root mean squared error) and MAPE (mean absolute percentage error), to evaluate the performance of forecasters, according to:

$$\mathrm{MSE} = \frac{1}{D}\sum_{i=1}^{D}(y(i) - y_{target}(i))^2, \tag{10}$$

$$\mathrm{NMSE} = \frac{\mathrm{MSE}}{\sigma^2_{y_{target}}}, \tag{11}$$

$$\mathrm{NRMSE} = \sqrt{\mathrm{NMSE}}, \tag{12}$$

$$\mathrm{MAPE} = \frac{100}{D}\sum_{i=1}^{D}\frac{|y(i) - y_{target}(i)|}{y(i)}, \tag{13}$$

where $D$ is the total number of data in the set; $y(i)$ and $y_{target}(i)$ are the actual and desired (target) outputs of the network, respectively; $\sigma^2_{y_{target}}$ is the variance of the values in the target data set.

When proposing a new forecasting method, it is very important to highlight and quantify the gains obtained in relation to the reference methods [30]. The expression used to calculate these percentage gains is the following:

$$\mathrm{G}_{\mathrm{ref,EC}} = 100\left(\frac{\mathrm{EC}_{\mathrm{ref}} - \mathrm{EC}}{\mathrm{EC}_{\mathrm{ref}}}\right), \tag{14}$$

where $\mathrm{EC}_{\mathrm{ref}}$ and EC are the evaluation criteria for the reference and proposed method, respectively.

Due to the random characteristics of the RCDESIGN-AS method, thirty initializations of the method were performed for each of the time series used in this paper, with the average performance for a given series corresponding to the average of the performances in the thirty initializations.

The average test performances of persistence, ESN and ESN-AS for TRI and BJD are presented in Tables 3 and 4, respectively, in which the values in bold text indicate the best performance for the corresponding criterion.

**Table 3.** Average test performance—TRI.

| Criterion | Persistence | ESN | ESN-AS |
|---|---|---|---|
| NRMSE | 0.64501 | 0.43601 | **0.1944** |
| NMSE | 0.41604 | 0.19011 | **0.069111** |
| MAPE (%) | 22.46 | 9.86 | **7.77** |
| MAE (m/s) | 1.92 | 0.87 | **0.84** |
| Generations | - | 4.5 | **2.9** |
| Spectral radius | - | 1.59 | **0.95** |
| Reservoir size | - | 86 | **43** |

**Table 4.** Average test performance—BJD.

| Criterion | Persistence | ESN | ESN-AS |
|---|---|---|---|
| NRMSE | 0.92295 | 0.4923 | **0.38854** |
| NMSE | 0.85183 | **0.24237** | 0.25798 |
| MAPE (%) | 23.99 | **12.08** | 13.08 |
| MAE (m/s) | 1.22 | **0.62** | 0.64 |
| Generations | - | 7.4 | **3.5** |
| Spectral radius | - | 1.53 | **1.02** |
| Reservoir size | - | 79 | **43** |

The average test performances of persistence, WTESN and RCDESIGN-AS for MAC, MOS and NAT are presented in Tables 5, 6 and 7, respectively.

**Table 5.** Average test performance—MAC.

| Criterion | Persistence | WTESN | ESN-AS |
|---|---|---|---|
| MSE ($m^2/s^2$) | 2.4315 | 1.6476 | **0.8322** |
| MAE (m/s) | 1.195 | 0.9934 | **0.7144** |
| Generations | - | - | **2.9** |
| Spectral radius | - | 0.9 | **0.99** |
| Reservoir size | - | 1000 | **44** |

**Table 6.** Average test performance—MOS.

| Criterion | Persistence | WTESN | ESN-AS |
|---|---|---|---|
| MSE ($m^2/s^2$) | 1.2315 | 0.8017 | **0.4314** |
| MAE (m/s) | 0.8275 | 0.6708 | **0.5336** |
| Generations | - | - | **2.8** |
| Spectral radius | - | 0.9 | **0.99** |
| Reservoir size | - | 1000 | **44** |

**Table 7.** Average test performance—NAT.

| Criterion | Persistence | WTESN | ESN-AS |
|---|---|---|---|
| MSE ($m^2/s^2$) | 1.34 | 0.9080 | **0.2906** |
| MAE (m/s) | 0.883 | 0.7437 | **0.4749** |
| Generations | - | - | **4.7** |
| Spectral radius | - | 0.9 | **1.01** |
| Reservoir size | - | 1000 | **43** |

As can be seen in Tables 3–7, the performances of RCDESIGN-AS are far superior to that of the *Persistence Method*, which is confirmed in Table 8, which presents the gains (14)

for TRI and BJD based on the MAE and MSE criteria. The same is valid for MAC, MOS and NAT, according to Table 9.

**Table 8.** Gains (%) over Persistence—TRI and BJD.

| | BJD | | TRI | |
|---|---|---|---|---|
| **error** | **ESN** | **ESN-AS** | **ESN** | **ESN-AS** |
| MSE | **71.5** | 69.6 | 83.2 | **83.8** |
| MAE | **49.2** | 47.5 | 54.7 | **56.3** |

**Table 9.** Gains (%) over Persistence—MAC, MOS and NAT.

| | MAC | | MOS | | NAT | |
|---|---|---|---|---|---|---|
| **error** | **WTESN** | **ESN-AS** | **WTESN** | **ESN-AS** | **WTESN** | **ESN-AS** |
| MSE | 32.2 | **65.8** | 34.9 | **65.0** | 32.2 | **78.3** |
| MAE | 16.9 | **40.2** | 18.9 | **35.5** | 15.8 | **46.2** |

The main parameters of the best solutions for the test sets of all evaluated series with the application of RCDESIGN-AS are presented in Table 10, in which the average time (shown in the last line) refers to the time needed to carry out the training of a single ESN-AS.

An overview of the eigenvalues of **W** as well as the actual versus forecasted values for the last 240 wind speeds of the test set for TRI, is presented in Figures 5 and 6, respectively. Due to the restriction on the total number of pages in this article, we do not present the equivalent figures for the other evaluated series, whose details can be checked in Gouveia [16].

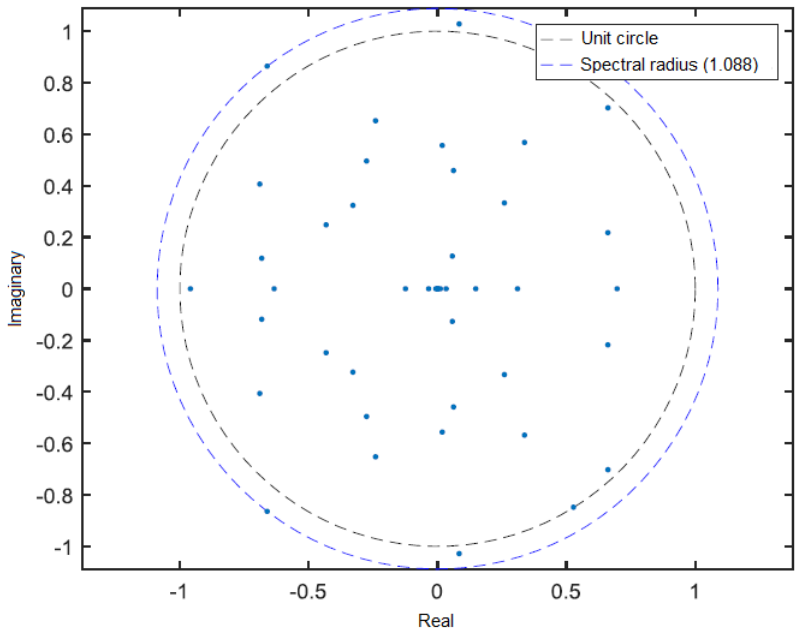


**Figure 5.** Eigenvalues of **W**. Adapted from Gouveia [16].

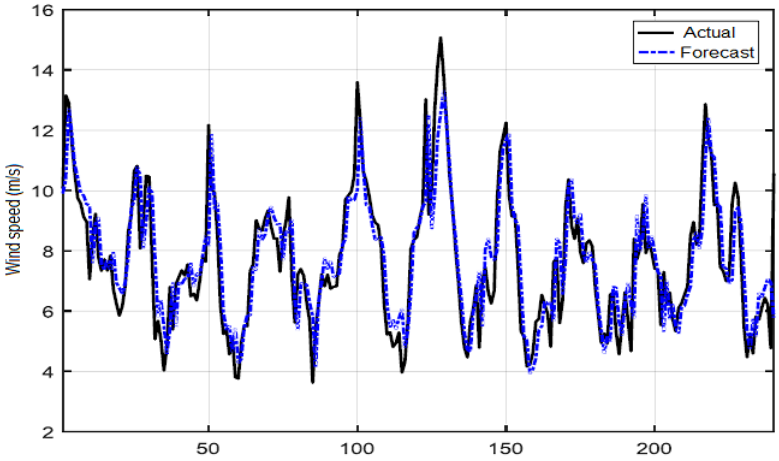


**Figure 6.** Sample of forecasts for TRI. Adapted from Gouveia [16].

**Table 10.** Main characteristics of the best ESN-AS.

| Parameter | TRI | BJD | MAC | MOS | NAT |
|---|---|---|---|---|---|
| Generations | 1 | 4 | 2 | 2 | 5 |
| Reservoir size | 45 | 46 | 44 | 50 | 46 |
| Sparsity of **W** | 91.6% | 95.8% | 90.8% | 93.4% | 91.7% |
| Spectral radius | 1.088 | 1.015 | 1.362 | 1.156 | 1.127 |
| Input scaling | 9.463 | 6.926 | 1.222 | 7.710 | 3.465 |
| Leak rate | 0.477 | 0.332 | 0.436 | 0.715 | 0.501 |
| Average time (s) | 2.93 | 0.82 | 0.79 | 0.64 | 0.69 |

## 7. Conclusions

In this paper, we present models created for wind forecasting within a 24-h ahead forecasting horizon using real data of wind speeds from five cities in the Brazilian Northeast.

We used the *Persistence Method* as a reference for comparing the performances of the models developed with the RCDESIGN-AS, which combines the advantages of an evolutionary strategy with RC in order to generate an automatic process for creating and training ESNs. Through this method, it is possible to investigate different topologies, as well as evaluate many parameter combinations. The use of ESN-AS enables the reduction of computational effort to train the readout, because augmented states allow reductions in the reservoir size without a loss in performance. Another interesting point is that all the reservoir and readout activation functions can be linear (*identity*).

We verified that models created with ESN or ESN-AS overcame the *Persistence Method* for wind speed forecasting in all time series. Although models created with ESN and ESN-AS both presented good performances, ESN-AS performed better than ESN for all except for one of the five series evaluated.

**Author Contributions:** Conceptualization, H.T.V.G., A.A.F. and R.R.B.d.A.; methodology, H.T.V.G., M.A.S., A.A.F. and R.R.B.d.A.; software, H.T.V.G., M.A.S. and J.C.d.A.; validation, M.M.d.S.L. and O.N.N.; formal analysis, R.R.B.d.A. and A.A.F.; investigation, H.T.V.G. and M.A.S.; resources, R.R.B.d.A., M.M.d.S.L. and O.N.N.; data curation, H.T.V.G., M.A.S. and J.C.d.A.; writing—original draft preparation, H.T.V.G. and M.A.S.; writing—review and editing, H.T.V.G., M.A.S., J.C.d.A., M.M.d.S.L. and O.N.N.; visualization, H.T.V.G. and M.A.S.; supervision, R.R.B.d.A. and A.A.F.; project administration, R.R.B.d.A.; funding acquisition, R.R.B.d.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was financed, in part, by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001. Implemented at Federal University of Pernambuco (UFPE), nº 054735/2022-11, through the PROPG nº 06/2022 announcement.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Venayagamoorthy, G.K.; Rohrig, K.; Erlich, I. One Step Ahead: Short-Term Wind Power Forecasting and Intelligent Predictive Control Based on Data Analytics. *IEEE Power Energy Mag.* **2012**, *10*, 70–78. [CrossRef]
2. Milligan, M.; Kirby, B.; Acker, T.; Ahlstrom, M.; Frew, B.; Goggin, M.; Lasher, W.; Marquis, M.; Osborn, D. *Review and Status of Wind Integration and Transmission in the United States. Key Issues and Lessons Learned*; Technical Report; National Renewable Energy Lab. (NREL): Golden, CO, USA, 2015.
3. Lei, M.; Shiyan, L.; Chuanwen, J.; Hongling, L.; Yan, Z. A review on the forecasting of wind speed and generated power. *Renew. Sustain. Energy Rev.* **2009**, *13*, 915–920. [CrossRef]
4. Jung, J.; Broadwater, R.P. Current status and future advances for wind speed and power forecasting. *Renew. Sustain. Energy Rev.* **2014**, *31*, 762–777. [CrossRef]
5. Fortuna, L.; Nunnari, S.; Guariso, G. Fractal order evidences in wind speed time series. In Proceedings of the ICFDA'14 International Conference on Fractional Differentiation and Its Applications 2014, Catania, Italy, 23–25 June 2014; pp. 1–6.
6. Shen, Z.; Ritter, M. Forecasting volatility of wind power production. *Appl. Energy* **2016**, *176*, 295–308. [CrossRef]
7. Fazelpour, F.; Tarashkar, N.; Rosen, M.A. Short-term wind speed forecasting using artificial neural networks for Tehran, Iran. *Int. J. Energy Environ. Eng.* **2016**, *7*, 377–390. [CrossRef]
8. Kadhem, A.A.; Wahab, N.I.A.; Aris, I.; Jasni, J.; Abdalla, A.N. Advanced Wind Speed Prediction Model Based on a Combination of Weibull Distribution and an Artificial Neural Network. *Energies* **2017**, *10*, 1744. [CrossRef]
9. Zheng, D.; Shi, M.; Wang, Y.; Eseye, A.; Zhang, J. Day-Ahead Wind Power Forecasting Using a Two-Stage Hybrid Modeling Approach Based on SCADA and Meteorological Information, and Evaluating the Impact of Input-Data Dependency on Forecasting Accuracy. *Energies* **2017**, *10*, 1988. [CrossRef]
10. Wu, Z.; Luo, G.; Yang, Z.; Guo, Y.; Li, K.; Xue, Y. A comprehensive review on deep learning approaches in wind forecasting applications. *CAAI Trans. Intell. Technol.* **2022**, *7*, 129–143. [CrossRef]
11. Chen, Q.; Folly, K.A. Short-Term Wind Power Forecasting Using Mixed Input Feature-Based Cascade-connected Artificial Neural Networks. *Front. Energy Res.* **2021**, *9*, 411. [CrossRef]
12. He, B.; Ye, L.; Pei, M.; Lu, P.; Dai, B.; Li, Z.; Wang, K. A combined model for short-term wind power forecasting based on the analysis of numerical weather prediction data. *Energy Rep.* **2022**, *8*, 929–939. [CrossRef]
13. Hallas, M.; Dorffner, G. *A Comparative Study on Feedforward and Recurrent Neural Networks in Time Series Prediction Using Gradient Descent Learning*; Vienna University of Economics and Business Administration: Wien, Austria, 1998.
14. Jaeger, H. Adaptive Nonlinear System Identification with Echo State Networks. In *Advances in Neural Information Processing Systems 15*; Becker, S., Thrun, S., Obermayer, K., Eds.; MIT Press: Cambridge, MA, USA, 2003; pp. 609–616.
15. Ferreira, A.A.; Ludermir, T.B.; de Aquino, R.R.B. An approach to reservoir computing design and training. *Expert Syst. Appl.* **2013**, *40*, 4172–4182. [CrossRef]
16. Gouveia, H.T.V. Sistema de previsão de Geração eólica Baseado em Ferramentas de Inteligência Artificial. Ph.D. Thesis, Universidade Federal de Pernambuco, Recife, Brazil, 2018.
17. Jaeger, H. *The "Echo State" Approach to Analysing and Training Recurrent Neural Networks—With an Erratum Note1*; German National Research Center for Information Technology GMD: Bonn, Germany, 2010.
18. Lukosevicius, M.; Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **2009**, *3*, 127–149. [CrossRef]
19. Maass, W.; Natschläger, T.; Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **2002**, *14*, 2531–2560. [CrossRef] [PubMed]
20. Jaeger, H. *Short Term Memory in Echo State Networks*; German National Research Center for Information Technology GMD: Bonn, Germany, 2002.
21. Jaeger, H. *Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the Echo State Network Approach*; GMD-Report; GMD-Forschungszentrum Informationstechnik: Bonn, Germany, 2002.
22. Racca, A.; Magri, L. Robust Optimization and Validation of Echo State Networks for learning chaotic dynamics. *Neural Netw.* **2021**, *142*, 252–268. [CrossRef] [PubMed]
23. Liu, K.; Zhang, J. Nonlinear process modelling using echo state networks optimised by covariance matrix adaption evolutionary strategy. *Comput. Chem. Eng.* **2020**, *135*, 106730. [CrossRef]
24. Taylor, C.E. Complex Adaptive Systems. In *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Complex Adaptive Systems*; Holland, J.H., Ed.; MIT Press: Cambridge, MA, USA, 1994.
25. Baker, J.E. Reducing bias and inefficiency in the selection algorithm. In Proceedings of the Second International Conference on Genetic Algorithms, Cambridge, MA, USA, 28–31 July 1987; Volume 206, pp. 14–21.
26. Walson, S. Fresh forecasts [wind power forecasting]. *Power Eng.* **2005**, *19*, 36–38. [CrossRef]
27. SONDA—Sistema Nacional de Organização de Dados Ambientais. Available online: http://sonda.ccst.inpe.br/ (accessed on 17 January 2020).
28. INMET—Instituto Nacional de Meteorologia. Available online: http://www.inmet.gov.br (accessed on 17 January 2020).

29. Gouveia, H.T.V.; De Aquino, R.R.B.; Ferreira, A.A. Enhancing Short-Term Wind Power Forecasting through Multiresolution Analysis and Echo State Networks. *Energies* **2018**, *11*, 824. [CrossRef]
30. Madsen, H.; Pinson, P.; Kariniotakis, G.; Nielsen, H.A.; Nielsen, T.S. A protocol for standardizing the performance evaluation of short-term wind power prediction models. *Wind Eng.* **2005**, *29*, 475–489. [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.