Evolving Deep Echo State Networks for Intelligent Fault Diagnosis

Jianyu Long, Shaohui Zhang, and Chuan Li

***Abstract*—Echo state network (ESN) is a fast recurrent neural network with remarkable generalization performance for intelligent diagnosis of machinery faults. When dealing with high-dimensional signals mixed with much noise, however, the performance of a deep ESN is still highly affected by the random selection of input weights and reservoir weights, resulting in the optimal design of the deep ESN architecture, which is an open issue. For this reason, a hybrid evolutionary algorithm featuring a competitive swarm optimizer combined with a local search is proposed in this article. An indirect encoding method is designed based on the network characteristics of ESN to make the evolutionary process computationally economical. A layerwise optimization strategy is subsequently introduced for evolving deep ESNs. The results of two experimental cases show that the proposed approach has promising performance in identifying different faults reliably and accurately by comparing with other intelligent fault diagnosis approaches.**

***Index Terms*—Classification, echo state network (ESN), evolutionary algorithm, evolving neural network, fault diagnosis, neuroevolution.**

I. Introduction

WITH the rapid development of sensing, communication, control, and actuation, modern mechanical equipment become more and more complicated and functional. Data-driven fault diagnosis approaches have been applied for health management of key mechanical systems [1], [2]. Among various data-driven approaches, support vector machine (SVM) [3], [4], artificial neural network [5], [6], Bayesian estimation [7], [8], fuzzy clustering [9], and *k*-nearest neighbor [10] have been employed commonly in this field.

Deep learning (DL) algorithms realized by deep neural networks (DNNs) were developed recently dealing with complicated fault diagnosis tasks. Stacked autoencoders (SAEs) [11], convolutional neural networks (CNNs) [12], deep belief networks (DBNs) [13], and recurrent neural networks (RNNs) [14] were applied for intelligent fault diagnosis. The performance of the DL algorithm depends greatly on its parameter values and hyperparameter (e.g., number of layers, number of neurons, and coefficient values) settings [15]. For hyperparameter setting, the grid search method performed by exhaustively generating candidates from a grid of parameter values was always employed. Obviously, it is inefficient when there are many hyperparameters, especially when they are continuous numbers. Gradient-based algorithms, such as back-propagation algorithm, stochastic gradient descent, adaptive gradient algorithm, adaptive learning rate, and root mean square prop, are commonly used for parameter values optimization [15]. Although gradient-based information can reduce optimization time, these algorithms are still difficult to guarantee the global optimization of parameters for a DNN, and gradient vanishing and gradient exploding are also easy to encounter in the training process. Echo state network (ESN) [16], consisting of an input layer, a hidden layer (reservoir), and an output layer, was proposed by Jaeger as a powerful RNN branch. Although ESN is a fast learning network with remarkable generalization performance, its performance is still highly affected by the random selection of weights and hyperparameters. Seriously, this impact will expand for deep ESNs.

To optimize parameter values and hyperparameter settings globally, evolutionary algorithms, such as genetic algorithm (GA) [17], particle swarm algorithm (PSO) [18], and artificial bee colony [19], have been widely used for the connection weights optimization and the architecture setting of DNNs. These studies are generally referred to as neuroevolution. Since evolutionary algorithms are gradient-free and insensitive to local optima, neuroevolution is a promising field to further improve the performance of DNNs. Due to the particularity of ESN, evolutionary algorithms for evolving deep ESNs are different from other DNNs in coding and evolutionary strategy design. Zhong *et al*. [20] proposed a double-reservoir ESN for multiregime time series prediction, and employed GA to optimize four hyperparameters (two internal unit numbers and two spectral radii). Obviously, in their study, the input weights and reservoir weights are still determined using the classical method. Similarly, Wang and Yan [21] employed a binary PSO to only determine the optimal connection structures for output weights of ESNs, without optimizing the input weights and reservoir weights. Wang *et al*. [22] proposed a neuroevolution model named ESN-DE for electricity energy consumption forecasting, where the differential evolution (DE) algorithm is used to search optimal values of the three hyperparameters (scale of the reservoir, connectivity rate, and spectral radius) of ESN. In addition

Manuscript received May 27, 2019; revised July 30, 2019; accepted August 22, 2019. Date of publication September 9, 2019; date of current version March 17, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 71801046, Grant 51775112, and Grant 51975121, and in part by the Natural Science Foundation of Guangdong Province under Grant 2018A030310029. Paper no. TII-19-2051. *(Corresponding author: Chuan Li.)*

The authors are with the School of Mechanical Engineering, Dongguan University of Technology, Dongguan 523808, China (e-mail: longjy@dgut.edu.cn; zhangsh@dgut.edu.cn; chuanli@dgut.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2938884

to using evolutionary algorithms to optimize hyperparameters, some studies also attempt to optimize weight parameters. Basterrech *et al*. [23] presented a supervised learning approach by using PSO to find initial reservoir weights of the ESN model. However, to reduce the calculation costs, only a subset of the reservoir weights was optimized, which may cause a defect and result on local minima. Chouikhi *et al*. [24] employed a PSO to pretrain some fixed weight values within the ESN model, and subsequently performed the normal training process. In this article, the architecture setting of ESN is not optimized by PSO. Moreover, some hyperparameters of PSO are introduced to determine which weight values need to be selected from the input weights and reservoir weights for pretraining, and their values are determined empirically. The existing works act as pilots for DNN neuroevolution. However, the optimal design of parameter values and hyperparameter settings in the deep ESN architecture is still an open issue. To the best of our knowledge, there is no research on evolving deep ESNs for complicated intelligent classification or fault diagnosis problems.

In this article, a hybrid evolutionary algorithm (denoted as CSOLS) featuring a competitive swarm optimizer (CSO) combined with a local search (LS) is proposed for evolving ESNs for dealing with fault diagnosis tasks. A layerwise optimization strategy based on CSOLS is subsequently presented for evolving deep ESNs. The motivations and solutions of this article can be concluded as follows.

1) Since classical PSO is prone to stagnation or premature convergence in dealing with high-dimensional optimization problems, CSO, a novel variant of PSO, was proposed to overcome the "the curse of dimensionality" [25]. Simultaneous optimizations of parameter values and hyperparameter settings in a deep ESN must be a high-dimensional optimization problem because a large reservoir with numerous neurons is always constructed for an effective ESN. Therefore, CSO is employed as the main optimization framework of CSOLS.
2) Indirect encoding is always used when a large number of parameters need to be optimized in neuroevolution. Based on the network characteristics of ESNs, a novel indirect encoding method is introduced in CSOLS according to the idea of the encoding method designed for evolving unsupervised DNNs in [15]. Unlike the indirect encoding method proposed in [24] for ESNs, our method does not need additional parameters to select which weights need to be optimized.
3) In the classical training method of ESNs, echo state and separability are two important properties that need to be guaranteed in the random generation process of reservoir weights [16]. In the proposed CSOLS, an LS strategy is designed based on the two properties to accelerate the local optimization. Therefore, CSOLS combines the exploration breadth of swarm-based global search and the aggressive-improvement capability of neighborhood-based LS to ensure its effectiveness and efficiency.
4) Compared with ESNs, the deep ESN inevitably needs to optimize more parameters and hyperparameters. To avoid optimizing too many parameters at one time, a novel type of deep ESN is constructed first and a layerwise optimization strategy is subsequently proposed based on the proposed CSOLS for evolving the deep ESN.

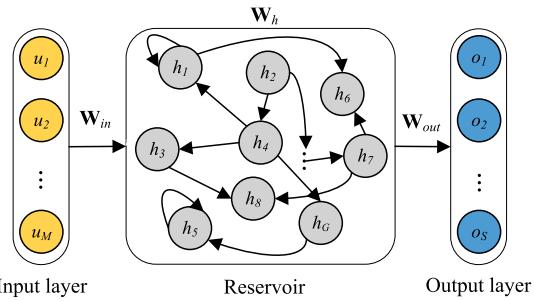


Fig. 1. Architecture of an ESN.

In summary, the main contributions of this article can be concluded as follows.

1) Simultaneous optimization of parameter values and hyperparameter settings for ESNs was realized by the CSOLS, leading to improved performance for intelligent fault diagnosis.
2) On the basis of the CSOLS designed with a novel indirect encoding method and LS strategy, a layerwise deep ESN optimization method was proposed to speed up the evolving procedure.
3) The present approach features a relatively performance-excellent yet computation-economical deep network evolving method, which may be applied for other DL applications.

The rest of this article is organized as follows. The background of ESN and CSO is described in Section II. Section III describes the details of the proposed CSOLS and the layerwise optimization strategy for evolving the deep ESN. Case studies of fault diagnosis using the proposed approach are reported in Section IV, and, finally, Section V concludes this article.

II. Preliminaries

A. Echo State Network

The architecture of an ESN, as shown in Fig. 1, is a three-layer network consisting of an input layer $\mathbf{u} = \{u_1, u_2, \ldots, u_M\}$, a reservoir $\mathbf{h} = \{h_1, h_2, \ldots, h_G\}$, and an output layer $\mathbf{o} = \{o_1, o_2, \ldots, o_S\}$. The number of neurons in the input layer, the reservoir, and the output layer is M, G, and S, respectively. M is determined by the data dimension of sample, S is determined by the number of categories, whereas G is a hyperparameter of ESN. In the network, $\mathbf{W}_{\text{in}} \in R^{M\times G}$, $\mathbf{W}_h \in R^{G\times G}$, and $\mathrm{W}_{\text{out}} \in R^{G\times S}$ denote the input weight matrix, reservoir weight matrix, and output weight matrix, respectively. The network is governed by

$$\mathbf{h}(\mathbf{t}+\mathbf{1}) = f_1(\mathbf{W}_{\text{in}}\mathbf{u}(\mathbf{t}+\mathbf{1}) + \mathbf{W}_h\mathbf{h}(\mathbf{t})) \quad (1)$$

$$\mathbf{o}(\mathbf{t}+\mathbf{1}) = f_2(\mathbf{W}_{\text{out}}\mathbf{h}(\mathbf{t}+\mathbf{1})) \quad (2)$$

where $\mathbf{t}$ denotes the index of sample, and f_1 and f_2 represent the activation functions used in the network. Activation function selection is also a hyperparameter of ESN.

The three weight matrices should be determined in the training process. The classical training method can be regarded as a heuristic parameter optimization method, which is performed by first randomly generating $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_h$, and then optimally calculating $\mathbf{W}_{\text{out}}$. To ensure the performance of ESN, randomly generated $\mathbf{W}_h$ needs to satisfy two properties named echo state and separability. Echo state property refers to that the reservoir will asymptotically wash out any information from initial conditions, which is empirically guaranteed by adjusting the spectral radius of reservoir weight matrix less than 1. Separability property means that different states can be generated from different inputs, which is empirically guaranteed by determining a sparse matrix using a small connectivity rate for reservoir weights. Therefore, the spectral radius and the connectivity rate are the other two hyperparameters of ESN. Once $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_h$ are achieved, $\mathbf{W}_{\text{out}}$ can be optimized by solving an optimization problem, whose objective function is the mean square error (MSE) between the desired label sequence and the predicted one. The optimization problem is formulated as follows:

$$\mathbf{W}_{\text{out}} = \arg\min_{\mathbf{W}_{\text{out}}} \text{MSE}(\mathbf{o}^{\text{desired}}, \mathbf{o}) \tag{3}$$

$$\text{MSE}(\mathbf{o}^{\text{desired}}, \mathbf{o}) = \frac{1}{S}\frac{1}{T}\sum_{s=1}^{S}\sum_{t=1}^{T}\left[o_s^{\text{desired}}(t) - o_s(t)\right]^2 \tag{4}$$

where $o_s^{\text{desired}}(t)$ is the desired label of the sth output of sample t. If sample t belongs to fault category s, $o_s^{\text{desired}}(t) = 1$, otherwise, $o_s^{\text{desired}}(t) = 0$. T denotes the number of samples in the training data. $o_s(t)$ is the sth output of ESN for sample t. The range of values for each $o_s(t)$ $(s \in \{1, 2, \ldots, S\})$ is [0,1], and the predicted label of ESN for sample t is the category s with the largest $o_s(t)$ value.

B. Competitive Swarm Optimizer

As a new variant of PSO, CSO is proposed recently to deal with high-dimensional optimization problems [25]. In PSO or CSO, a swarm is initialized with N particles, where each particle denotes a candidate solution for an optimization problem. The particles will be updated in the process of swarm iteration to search for the optimal solution. Each particle p has its individual position and velocity, which are represented by $\mathbf{X}_p = \{x_p^1, x_p^2, \ldots, x_p^D\}$ and $\mathbf{V}_p = \{v_p^1, v_p^2, \ldots, v_p^D\}$, respectively. D refers to the number of decision variables in the optimization problem. CSO mainly differs from PSO in its powerful particle update mechanism.

To detail the learning process between two iterations in CSO, $P(k) = \{\mathbf{X}_1(k), \ldots, \mathbf{X}_p(k), \ldots \mathbf{X}_N(k)\}$ $(\mathbf{X}_p(k) = \{x_p^1(k), \ldots, x_p^D(k)\})$ is used to denote the swarm at iteration k. After the fitness values of all particles in $P(k)$ are calculated, the swarm is randomly divided into two equal groups. A pairwise competition is performed between two particles, one randomly selected from the first group and the other randomly selected from the second group. Then, the particle with a better fitness value is called the winner, and it goes directly into $P(k+1)$. On the contrary, the other particle called the loser is updated based on the winner, and then is transferred to $P(k+1)$. For ease of description, the positions and velocities of the winner and loser are represented as $\mathbf{X}_{p,w}(k)$, $\mathbf{V}_{p,w}(k)$, $\mathbf{X}_{p,l}(k)$, and $\mathbf{V}_{p,l}(k)$, respectively. Now, the update procedure for the loser can be formulated as

$$\begin{aligned}\mathbf{V}_{p,l}(k+1) = {} & r_1(p,k)\mathbf{V}_{p,l}(k) + r_2(p,k)(\mathbf{X}_{p,w}(k) - \mathbf{X}_{p,l}(k)) \\ & + \lambda r_3(p,k)(\bar{\mathbf{X}}_p(k) - \mathbf{X}_{p,l}(k))\end{aligned} \tag{5}$$

$$\mathbf{X}_{p,l}(k+1) = \mathbf{X}_{p,l}(k) + \mathbf{V}_{p,l}(k+1) \tag{6}$$

where coefficient values $r_1(p,k)$, $r_2(p,k)$, and $r_3(p,k)$ are randomly generated from [0, 1], $\bar{\mathbf{X}}_p(k)$ is the mean position of the swarm, and λ is the only parameter used to control the influence of $\bar{\mathbf{X}}_p(k)$.

III. Evolving Deep ESNs

A. Proposed CSOLS

As stated before, the classical method used to train an ESN is a heuristic parameter optimization approach, which makes it difficult to obtain the optimal combination of parameters. Moreover, several hyperparameters (e.g., spectral radius and connectivity rate) used to control the values of connection weights, activation function selection, and network architecture setting are all determined empirically, which may affect the generalization performance of ESNs.

To optimize parameter values and hyperparameter settings simultaneously, we propose a hybrid evolutionary algorithm that combines the CSO with LS (CSOLS) for evolving ESNs. The procedure of CSOLS is shown in Algorithm 1.

Details of the proposed Algorithm 1 are described as follows.

1) Indirect Encoding and Decoding: Since a large number of parameters need to be optimized in ESNs, an indirect encoding method is proposed to avoid excessive length of the particle. The indirect encoding method we propose is based on the encoding method introduced by Sun *et al.* [15], but it includes some modifications considering the network characteristics of ESNs.

With respect to the parameter optimization, only the value information of weight matrices $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_h$ need to be encoded because $\mathbf{W}_{\text{out}}$ can be optimally calculated with fixed $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_h$. The dimension of $\mathbf{W}_{\text{in}}$ is $M \times G$, and the dimension of $\mathbf{W}_h$ is $G \times G$. In CSOLS, G denotes the maximum number of neurons set for the reservoir. Therefore, G is generally greater than M. Based on the theory of linear algebra, we know that a set of orthogonal vectors $a_i \in R^G$ $(i = 1, 2, \ldots G)$ is sufficient to span the G-dimensional space R^G. Moreover, when one basis vector a_1 is computed, as shown in (7), the other $(G-1)$ G-dimensional basis vectors can be obtained from the singular value decomposition to find the null space. Therefore, the encoding of length G can efficiently model the problem with G^2 parameters. Based on the above analysis, in this article, a set of bases consisting of G linear independent G-dimensional vectors $Z = [z_1, z_2, \ldots, z_G]$ $(z_i \in R^G)$ is given first. For simplicity, we use the G-dimensional unit matrix as Z. Subsequently, through linearly combining with the bases in Z, two vectors of coefficients, $b_{\text{in}} \in R^G$ and $b_h \in R^G$, are encoded to compute the vectors a_1^{in} and a_1^h for W_{in} and $\mathbf{W}_h$, respectively. Then, the orthogonal complements $\{a_2^{\text{in}}, a_3^{\text{in}}, \ldots, a_G^{\text{in}}\}$ of a_1^{in}

Algorithm 1: CSOLS for Evolving ESNs.

Input: swarm size N, control parameter λ, maximum number of neurons in reservoir G, maximum number of fitness evaluations *MAX_F*
Output: The final best particle p^* and its fitness value *Fitness*(p^*)

Set a counter $cf = 0$;
Initialize a swarm P with N particles;
Calculate the fitness value of each particle in the swarm by constructing, training and validating a ESN model based on the particle information;
$cf += N$;
The best particle of the swarm is p^*, and its fitness value is *Fitness*(p^*);
While $cf \leq MAX\_F$ **do**
Set $\Psi = P$ and $P = \emptyset$;
While $\Psi \neq \emptyset$ **do**
Randomly select two particles from Ψ, where the better particle is denoted as $\mathbf{X}_{p,w}$, and the other one is denotedas $\mathbf{X}_{p,l}$;
$P = P \cup \mathbf{X}_{p,w}$;
Update $\mathbf{X}_{p,l}$ using Equation (5) and Equation (6);
Calculate the new fitness value $Fitness(\mathbf{X}_{p,l})$;
$cf += 1$;
If a new particle $\mathbf{X'}_{p,l}$ is generated using the LS strategy on $\mathbf{X}_{p,l}$ **then**
Calculate the fitness value $Fitness(\mathbf{X'}_{p,l})$;
If $Fitness(\mathbf{X'}_{p,l}) < Fitness(\mathbf{X}_{p,l})$ **then**
$\mathbf{X}_{p,l} = \mathbf{X'}_{p,l}$ and $Fitness(\mathbf{X}_{p,l}) = Fitness(\mathbf{X'}_{p,l})$;
End
$cf += 1$;
End
$P = P \cup \mathbf{X}_{p,l}$;
If $Fitness(\mathbf{X}_{p,l}) < Fitness(p^*)$ **then**
$p^* = \mathbf{X}_{p,l}$ and $Fitness(p^*) = Fitness(\mathbf{X}_{p,l})$;
End
Delete particles $\mathbf{X}_{p,w}$ and $\mathbf{X}_{p,l}$ from Ψ;
End While
End While

and $\{a_2^h, a_3^h, \ldots, a_G^h\}$ of a_1^h can be computed by (7). Now, both $A^{\text{in}} = \{a_1^{\text{in}}, a_2^{\text{in}}, \ldots, a_G^{\text{in}}\}$ and $A^h = \{a_1^h, a_2^h, \ldots, a_G^h\}$ are able to span a G-dimensional space. Since the dimension of $\mathbf{W}_{\text{in}}$ is $M \times G$, a binary encoded vector e_{in} with M one values and $G - M$ zero values is employed to select a submatrix $\tilde{A}^{\text{in}}$ $(M \times G)$ from A^{in} for $\mathbf{W}_{\text{in}}$

$$\text{null space}(a_1) = \{x \in R^G | x a_1 = 0\}. \tag{7}$$

Since the values of $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_h$ are determined by the encoding, these hyperparameters (e.g., spectral radius and connectivity rate) used to control the values of connection weights do not need to be encoded. As for the activation function selection, a list of candidate activation functions is first given, and their indexes are encoded to decide which one will be used.

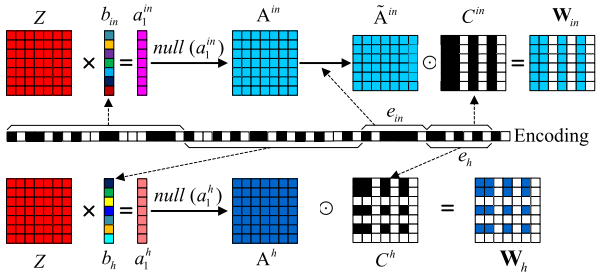


Fig. 2. Flowchart of the proposed encoding and decoding method. The encoding contains 20*G*+2 b. The decoding procedure is as follows: the *G*-dimensional unit matrix *Z* is given first; a set of coefficients b_{in}/b_h is generated according to the encoding information, and the vector a_1^{in}/a_1^h is then computed by multiplying *Z* by b_{in}/b_h; the orthogonal complements $\{a_2^{\text{in}}, \ldots, a_G^{\text{in}}\}/\{a_2^h, \ldots, a_G^h\}$ of a_1^{in}/a_1^h are subsequently computed to obtain a *G*-dimensional square matrix A^{in}/A^h; a submatrix $\tilde{A}^{\text{in}}$ $(M \times G)$ is then selected from A^{in} based on the encoding information of e_{in}; two coefficient matrices C^{in} $(M \times G)$ and C^h $(G \times G)$ are computed based on the encoding information of e_h; finally, $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_h$ are achieved by computing the Hadamard product $C^{\text{in}} \odot \tilde{A}^{\text{in}}$ and $C^h \odot A^h$, respectively.

Network architecture setting, namely the number of neurons in the reservoir, also needs to be optimized. Note that the maximum number of neurons set for the reservoir is G. Then, a binary encoded vector e_h with G values is used to determine which neurons are available, which is somewhat similar to the idea of dropout for DNNs. Based on e_h, two coefficient matrices C^{in} $(M \times G)$ and C^h $(G \times G)$ can be computed using (8) and (9), respectively. Finally, (10) and (11) are used to determine $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_h$, where $\odot$ represents the Hadamard product of matrices

$$C_{ij}^{\text{in}} = \begin{cases} 0 & \text{if } e_h(k) = 0 \text{ and } j = k \\ 1 & \text{otherwise} \end{cases}, \quad i = \{1, 2, \ldots, M\}$$
$$j = \{1, 2, \ldots, G\}, \quad k = \{1, 2, \ldots, G\} \tag{8}$$

$$C_{ij}^h = \begin{cases} 0 & \text{if } e_h(k) = 0 \text{ and } (i = k \text{ or } j = k) \\ 1 & \text{otherwise} \end{cases}$$
$$i = \{1, 2, \ldots, G\},\ j = \{1, 2, \ldots, G\},\ k = \{1, 2, \ldots, G\} \tag{9}$$

$$\mathbf{W}_{\text{in}} = C^{\text{in}} \odot \tilde{A}^{\text{in}} \tag{10}$$

$$\mathbf{W}_h = C^h \odot A^h. \tag{11}$$

Fig. 2 shows the flowchart of the proposed encoding and decoding method to intuitively illustrate its philosophy. For computational efficiency, the binary encoding method is employed here to encode the above information. Each coefficient of b_{in} and b_h is encoded with 9 b, where the leftmost bit represents its positive or negative sign. For example, the encoding (110110100) refers to the float value $-2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} = -0.703125$. In total, $18G$ bits need to be used to encode b_{in} and b_h. The coded range of each coefficient is $[2^{-8} - 1,\ 1 - 2^{-8}]$, which is sufficient to cover their ideal range $[-1,\ 1]$. Obviously, encoding e_{in} and e_h requires $2G$ bits. Besides, 2 b are employed to encode the selection of activation functions because the three commonly used activation functions (identity, tanh, and sigmoid) for ESNs are considered.

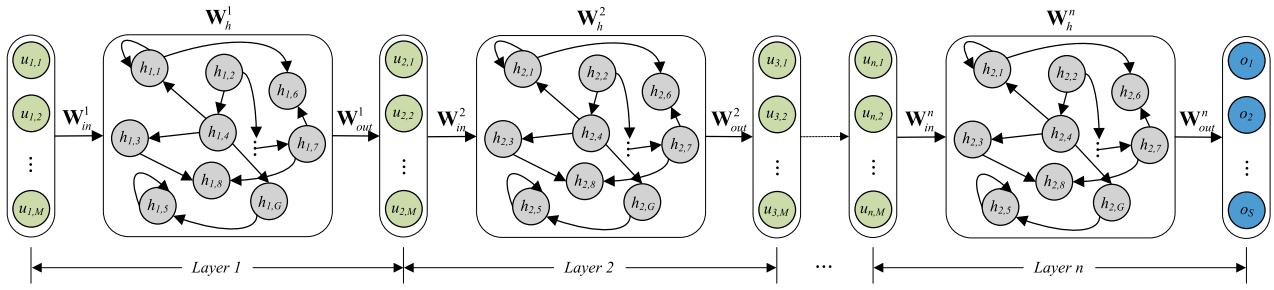


Fig. 3. Architecture of deep ESNs.

Therefore, in total, the length of position or velocity for a particle is $D = 20G + 2$.

2) Fitness Value Calculation: Calculating fitness value is used to evaluate the quality of each particle in the swarm. To avoid overfitting, a small proportion of data from the training data is randomly selected as the validation data. For each particle, a corresponding ESN can be constructed based on its decoding information, and it can be subsequently trained using the training data. Then, its fitness value is calculated as

$$\text{Accuracy} = \frac{\sum_{t=1}^{\tilde{T}} \sum_{s=1}^{S} f(t,s)o(t,s)}{\tilde{T}} \tag{12}$$

where $\tilde{T}$ denotes the number of samples in the validation set, S denotes the number of fault classes, $f(t,s)$ is equal to 1 if and only if the sample t belongs to fault class s, and $o(t,s)$ is equal to 1 if and only if the predicted fault class of sample t is s.

3) LS Procedure: As stated before, echo state and separability are two important properties to ensure the performance of ESNs in the classical training method. In the proposed CSOLS, since the input weights and reservoir weights are all optimized, those hyperparameters used to control the values of connection weights to reach the two properties do not need to be encoded. However, we can design an LS strategy based on these two properties to further improve the quality of the particle in the searching process. More specifically, when the spectral radius of reservoir weight matrix $\mathbf{W}_h$ of the corresponding ESN model for a particle is greater than 1, it can be decreased by iteratively reducing the absolute value of randomly selected coefficients of b_h to zero for searching a better particle. Note that all the random processes in LS procedure are performed by the uniform random distribution. The iteration process stops until the value of the spectral radius is less than 1. Since the binary encoding method is employed for each coefficient of b_h, all of the corresponding gene values need to be set to zero for a selected coefficient. With respect to the separability, if a particle has high prediction accuracy on the training set but low accuracy on the validation set, we can increase the sparsity of $\mathbf{W}_h$ by randomly changing some coefficients in b_h from nonzero value to zero because it may be overfitting. On the contrary, if a particle has low accuracy on the training set, we can randomly alter some coefficients in b_h from zero to a nonzero value generated randomly from $[-1, 1]$ to increase its nonlinear fitting ability. To realize this operation for a selected coefficient, it can be realized by randomly alter some of its gene values from zero to one.

B. Layerwise Optimization Method for Evolving Deep ESNs

Most of the existing deep ESNs are constructed by integrating multiple layers of reservoir [26], [27], which makes it very difficult to evolve a deep ESN because adding one more reservoir layer will add a lot of decision variables in the evolutionary algorithm. To overcome this problem, a layerwise optimization method is designed based on the proposed CSOLS. First, a novel deep ESN structure shown in Fig. 3 is constructed by integrating multiple ESNs rather than multiple layers of reservoir. Each layer of the deep ESN is an ESN consisting of an input sublayer, a reservoir sublayer, and an output sublayer. Moreover, the output sublayer of the former ESN acts as the input sublayer of the latter ESN. For any ESNs except the last one, its task is to learn meaningful representations (features) of input data through multiple nonlinear transformations, whereas the last ESN is used for the fault recognition.

The layerwise optimization method refers to the sequential optimization of each ESN in the deep ESN by means of the proposed CSOLS. Obviously, the method of optimizing the last ESN with CSOLS is the same as that of optimizing a separate ESN described in the last section. However, for any of the previous ESNs, since the label information cannot be used directly, the key problem is how to train the output weight matrix when the input weight matrix and reservoir weight matrix are determined in CSOLS. More specifically, for the ith ESN in the deep architecture, its output weight matrix $\mathbf{W}_{\text{out}}^{i}$ is optimized by solving the following optimization problem:

$$\mathbf{W}_{\text{out}}^{i} = \arg\min_{\mathbf{W}_{\text{out}}^{i}} \text{MSE}(\mathbf{u}_{i+1}^{\text{desired}}, \mathbf{u}_{i+1}) \tag{13}$$

$$\text{MSE}(\mathbf{u}_{i+1}^{\text{desired}}, \mathbf{u}_{i+1}) = \frac{1}{M}\frac{1}{T}\sum_{m=1}^{M}\sum_{t=1}^{T} \times \left[u_{i+1,m}^{\text{desired}}(t) - u_{i+1,m}(t)\right]^2 \tag{14}$$

where $\mathbf{u}_{i+1}^{\text{desired}}$ and $\mathbf{u}_{i+1}$ denote the desired and predicted fault feature sequences, respectively. $\mathbf{u}_{i+1}$ can be calculated easily because $\mathbf{W}_{\text{in}}^{i}$ and $\mathbf{W}_{h}^{i}$ are determined in CSOLS. Therefore, the only remaining problem is to determine the value of $\mathbf{u}_{i+1}^{\text{desired}}$.

Due to the high dimensionality, nonlinearity, and high data noise of real fault diagnosis tasks, samples of the same class are always scattered and are not easily distinguished from others. Therefore, improving the clustering performance of features

Algorithm 2: Layerwise Optimization Method for Evolving Deep ESNs.

Input: maximum number of layers n, parameters of CSOLS

Output: the diagnosis accuracy $Accuracy^*$ and the number of layers n^* of deep ESNs with optimal performance

```
Set a layer counter i = 1, Accuracy* = 0;
Initialize the input features u according to the training data;
While i ≤ n do
    Evolve the ith ESN using the proposed CSOLS based on the current input features u (the output weight matrix is optimized by Eqs. (3) and (4));
    Calculate the accuracy Accuracy(DESN_i) of the current evolved deep ESN with i layers on the validation set;
    If Accuracy* < Accuracy(DESN_i) do
        Accuracy* = Accuracy(DESN_i), n* = i;
    End
    If i < n do
        Evolve the ith ESN using CSOLS based on the current input features u (the output weight matrix is optimized according to Eqs. (13) and (14));
        Update u with the output features of the ith ESN;
    End
    i = i + 1;
End
```

is helpful to distinguish different classes. Based on this fact, to improve the clustering performance of features obtained from the previous ESN, the center of features for each class is calculated first by using the label information as

$$\mathbf{fc}_i(s) = \frac{1}{|\Phi_s|} \sum\nolimits_{t \in \Phi_s} \mathrm{sigm}(\mathbf{u}_i(t)) \tag{15}$$

where $\mathbf{fc}_i(s)$ is the feature center of the sth class, Φ_s is the sample set of the sth class, $|\Phi_s|$ is the number of samples in Φ_s, $\mathbf{u}_i(t)$ is a feature obtained from the previous ESN for sample t, and $\mathrm{sigm}(\cdot)$ denotes the sigmoid activation function. After calculating the feature centers of all classes, as shown in (16), the desired fault feature sequences for all the samples in the same class are set to their feature center directly

$$\mathbf{u}_{i+1}^{\mathrm{desired}}(t) = \mathbf{fc}_i(s), \quad t \in \Phi_s \tag{16}$$

Based on the above discussion, the procedure of the layerwise optimization method for evolving deep ESNs is presented in Algorithm 2.

IV. Experiments and Discussion

A. Experimental Case I: Fault Diagnosis for a Three-Dimensional (3-D) Printer

In this case, intelligent fault diagnosis experiments for a 3-D printer were carried out to test the performance of the proposed approach. The attitude data (three-axial vibratory acceleration signals, three-axial angular velocity ones, and three-axial magnetic field intensity ones) of the delta 3-D printer in different condition patterns were collected first on the test-rig shown in Fig. 4(a). Note that a cheap attitude sensor (BWT901) was used in the experiment to reduce the cost of fault diagnosis. Moreover, adopting a cheap attitude sensor will increase the noise of the data, which is helpful to test the present algorithm.

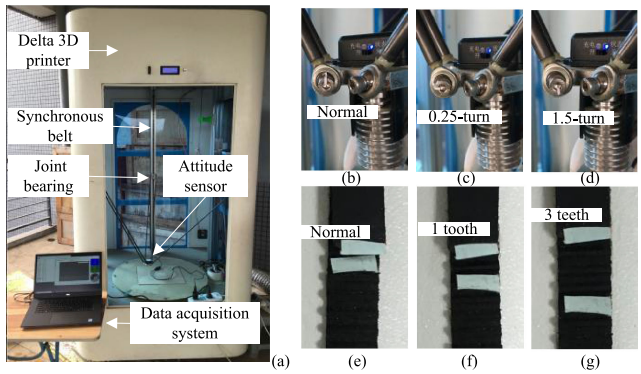


Fig. 4. Fault diagnosis experiments on a 3-D printer. (a) Test-rig. (b) NC of joint bearing. (c) Slight fault of joint bearing. (d) Serious fault of joint bearing. (e) NC of synchronous belt. (f) Slight fault of synchronous belt. (g) Serious fault of synchronous belt.

In the experiments, 15 fault patterns were simulated by loosening the screw of each joint bearing and relaxing the length of teeth of each synchronous belt. More specifically, the screw (0.7 mm pitch) of each joint bearing was loosened a 0.25-turn and 1.5-turn to simulate a slight fault (i.e., 0.175 mm clearance) and a serious fault (i.e., 1.05 mm clearance each), respectively. Similarly, the length of one tooth (1.5 mm) and three teeth (4.5 mm) is relaxed to simulate a slight fault and a serious fault, respectively. Two datasets (3-D Dataset_1 and 3-D Dataset_2), where 3-D Dataset_1 contains all the slight faults and 3-D Dataset_2 includes all the serious faults, were collected for the experiments. Therefore, each dataset contains 16 condition patterns, i.e., 15 fault patterns and 1 normal condition (NC). Fig. 4(b)–(g) shows the NC, slight fault, and serious fault of a joint bearing and a synchronous belt. For each condition pattern, 93 000 raw time signal points were collected by setting the sampling frequency as 100 Hz. The original signals were divided into 1500 samples, each of which contains 62 raw time signal points. Therefore, we finally obtained $1500 \times 16 = 24\,000$ samples for each dataset, and the data dimension of each sample is $62 \times 9 = 558$. In the experiments, 80% of the total 24 000 samples were randomly taken for training and the rests for testing.

Since the core of the optimization method designed for evolving deep ESNs is the CSOLS algorithm, the effect of parameter setting on its performance is investigated first in this section. As shown in Algorithm 1, there are four parameters in the proposed CSOLS. For each parameter, five levels were designed in the experiment as follows.

1) The levels of N are 10, 20, 30, 40, and 50.
2) The levels of λ are 0, 0.2, 0.4, 0.6, and 0.8.

TABLE I
INFORMATION OF PARAMETER CONFIGURATIONS

ID	N	λ	G	MAX_FS	ID	N	λ	G	MAX_FS
1	20	0.2	900	250	14	40	0.4	600	250
2	50	0	700	250	15	10	0.4	900	400
3	40	0	800	300	16	10	0	600	200
4	10	0.2	1000	300	17	30	0.2	800	200
5	30	0	900	350	18	50	0.8	800	400
6	20	0.4	800	350	19	10	0.6	800	250
7	40	0.2	700	400	20	30	0.8	1000	250
8	50	0.2	600	350	21	20	0.8	600	300
9	50	0.6	900	300	22	40	0.8	900	200
10	40	0.6	1000	350	23	50	0.4	1000	200
11	30	0.6	600	400	24	10	0.8	700	350
12	20	0.6	700	200	25	30	0.4	700	300
13	20	0	1000	400					

TABLE II
RESULTS OF THE TESTS OF BETWEEN-SUBJECTS EFFECTS

Source (Parameter)	Type III Sum of Squares	df	Mean Square	F-value	p-value
Model	0.006	16	0.000	3.250	0.048
Intercept	22.568	1	22.568	205692.555	0.000
N	0.004	4	0.001	9.566	0.004
λ	0.000	4	0.000	0.972	0.473
G	0.001	4	0.000	1.159	0.396
MAX_F	0.001	4	0.000	1.304	0.346
Error	0.001	8	0.000		
Total	22.574	25			

3) The levels of G are 600, 700, 800, 900, and 1000 because G needs to be greater than the data dimension of each sample.
4) The levels of $\mathrm{MAX\_}F$ are 200, 250, 300, 350, and 400.

Through employing the orthogonal experimental design method in the SPSS Statistics 23.0, 25 parameter configurations listed in Table I were generated. The diagnosis accuracy on test dataset was used as the response variable to evaluate the performance of the parameter configurations. Note that ten trials were carried out for each parameter configuration experiment to reduce the impacts of the randomness.

Following Long *et al*. [28], the experimental results of the parameter configurations were analyzed by employing the analysis of variance (ANOVA) in the SPSS Statistics 23.0. Taking 3-D Dataset_1 as an example, the tests of between-subjects effects are listed in Table II.

In Table II, p-value refers to the probability of obtaining the observed statistic. On the basis of the principle of hypothesis testing, only if the p-value is less than 0.05, the factor has a significant statistical effect on the objective. One can find that only parameter N has a significant effect on the performance because its p-value is less than 0.05. Therefore, according to its profile plot shown in Fig. 5, N should be set as 40. The setting of parameter λ has little effect on the performance, which is the same as the conclusion in [25]. Thus, λ is set as 0.2 in the experiments. Since the p-values of G and $\mathrm{MAX\_}F$ are both much greater than 0.05, we set $G = 600$ and $\mathrm{MAX\_}F = 200$ in the experiments by considering the computational complexity. G has no significant effect on the results, to some extent, which

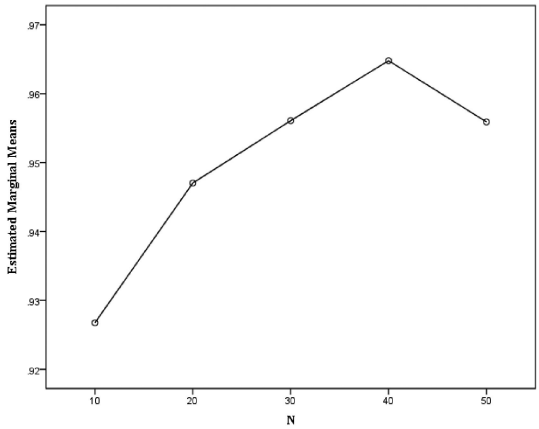


Fig. 5. Profile plot describing estimated marginal mean for each level of N.

demonstrates that the number of actually available neurons in the reservoir can be automatically optimized in the proposed CSOLS.

The performance of our proposed fault diagnosis method is then compared with six other fault diagnosis approaches, which are the original ESN, SVM, SAE, DBN, one-dimensional (1-D) CNN, and CSO (the proposed CSOLS without the LS procedure). To make a fair comparison, the abovementioned parameter tuning method used for the CSOLS algorithm was also employed to obtain the optimal parameter settings of these algorithms on each dataset. Table III lists parameter settings of all the algorithms in the experiments. Comparative results of ten trials are described in Fig. 6. Note that the results of the evolved deep ESN with different layers (at most four layers) are reported for a more comprehensive comparison. Thus, CSOLS-1 denotes the deep ESN with one layer, CSOLS-2 denotes the deep ESN with two layers, and so on.

From the results shown in Fig. 6, one can find that almost all the algorithms on 3-D Dataset_2 tend to obtain higher accuracy than 3-D Dataset_1, which indicates that slight faults are more difficult to detect. It is clear that the highest accuracy is always obtained by one of the proposed CSOLS-series algorithms. With respect to the results on 3-D Dataset_1, SVM always obtains much lower accuracy (69.01%) than the other algorithms, which verifies that the shallow model is difficult to achieve good results without effective features. The CSOLS-1 can obtain about 96.41% prediction accuracy, which exceeds that of the original ESN (81.96%) by a considerable margin. It demonstrates that evolving ESN using the proposed CSOLS can greatly improve the performance of the original ESN. The three DL algorithms (SAE, DBN, and CNN) have satisfactory performance with the accuracy more than 90%, which demonstrates the effectiveness of deep features. Through comparing the accuracies of CSO with that of CSOLS-1, we can find that combining the LS procedure can improve the performance of the algorithm. The highest accuracy (97.11%) on 3-D Dataset_1 is achieved by the CSOLS-2, which performs best with accuracy boost 4.60%, 6.25%, 2.01%, and 2.48% than SAE, DBN, CNN, and CSO, respectively. Similar comparisons also occur on

TABLE III
PARAMETER SETTINGS OF ALL THE COMPARED ALGORITHMS FOR FAULT DIAGNOSIS OF 3-D PRINTER

Algorithm	Parameter settings	
	3D Dataset_1	3D Dataset_2
ESN	Reservoir nodes= 450; spectral radius=0.3; connectivity rate=0.01	Reservoir nodes=400; spectral radius=0.3; connectivity rate=0.01
SVM	Kernel function=RBF; gamma=1.0; penalty coefficient=0.1	Kernel function=RBF; gamma=1.0; penalty coefficient=0.1
SAE	Hidden layer structure=[150-100]; learning rate=0.1; iterations=100; batchsize=10;	Hidden layer structure=[120-50]; learning rate=0.1; iterations=100; batchsize=10;
DBN	Hidden layer structure=[100]; learning rate=0.1; iterations=200; batchsize=10;	Hidden layer structure=[150]; learning rate=0.1; iterations=250; batchsize=10;
CNN	Hidden layer structure= [conv(8)-pool(2)-conv(10)-pool(3)-conv(10)-pool(3)]; kernel size=5; learning rate=0.1; iterations=150; batchsize=10;	Hidden layer structure= [conv(8)-pool(2)-conv(10)-pool(3)-conv(10)-pool(3)]; kernel size=5; learning rate=0.1; iterations=150; batchsize=10;
CSO	N =40; λ =0.2; G =600; MAX_F =280	N =50; λ =0.2; G =600; MAX_F =300
CSOLS	N =40; λ =0.2; G =600; MAX_F =200	N =50; λ =0.2; G =600; MAX_F =200

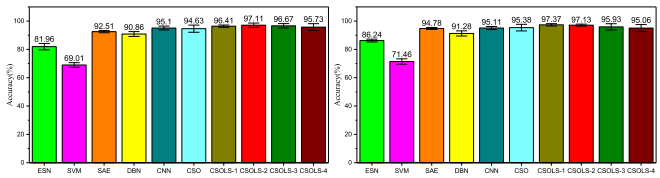


Fig. 6. Comparative results. (a) Diagnosis accuracy of all the algorithms on 3-D Dataset_1. (b) Diagnosis accuracy of all the algorithms on 3-D Dataset_2.

TABLE IV
TRAINING TIME OF ALL THE ALGORITHMS ON 3-D DATASETS

Algorithm	3D Dataset_1 (s)	3D Dataset_2 (s)
ESN	18.676	16.378
SVM	15.476	15.376
SAE	615.878	588.907
DBN	485.283	631.251
CNN	6936.351	7046.681
CSO	6327.276	6621.671
CSOLS-1	4576. 119	4484.872
CSOLS-2	8628.286	8487.387
CSOLS-3	12538.287	12287.276
CSOLS-4	16438.586	16178.976

the 3-D Dataset_2. Since the serious faults are easier to be diagnosed, the prediction accuracy of most algorithms has been improved slightly. The CSOLS-1 obtains the highest prediction accuracy (97.37%) on 3-D Dataset_2, which demonstrates the superiority and robustness of our proposed algorithm. As for the performance of the CSOLS-series algorithms on both two datasets, an interesting phenomenon can be found that the performance of evolved deep ESNs is not improved with the increase of the number of layers. The most likely reason is that some evolved deep ESN models with many layers may lead to overfitting. However, the proposed layerwise optimization method for evolving deep ESNs can solve this problem well because it does not choose the overfitting models as its final output.

Table IV lists the training times of all the compared algorithms. Note that the testing times for all algorithms are not provided because they are negligible. In general, neuroevolution is always time-consuming because a large number of model training operations need to be performed for evaluating the effect

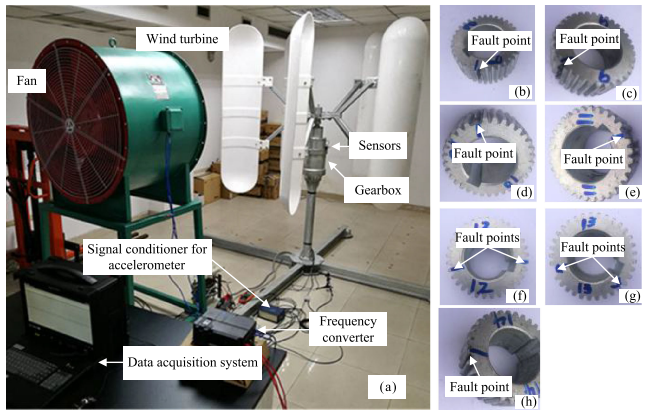
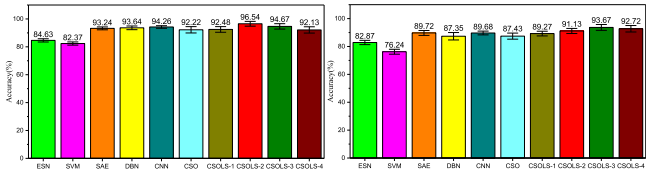


Fig. 7. Fault diagnosis of wind turbine gearbox. (a) Test-rig. (b) Slight single-point pitting. (c) Serious single-point pitting. (d) Half broken tooth. (e) Broken tooth. (f) Double-point pitting. (g) Single-point pitting and half broken tooth. (h) Groove.

of evolution. However, as listed in Table IV, the training times of the CSOLS-series algorithms are still acceptable in practical applications. Obviously, the first reason is that training an ESN model in the fitness calculation process is very efficient. Besides that, both the indirect encoding used for reducing the number of decision variables and the LS used for accelerating local optimization enable the CSOLS algorithm to converge quickly. In addition, the layerwise optimization method designed for evolving deep ESN also makes a good tradeoff between prediction accuracy and computational efficiency.

B. Experimental Case II: Fault Diagnosis for a Gearbox

To testify the robustness of the proposed approach, fault diagnosis for wind turbine gearbox was conducted on the test-rig shown in Fig. 7(a). A triaxial accelerometer sensor, a three-phase current sensor, an acoustic sensor, and an acoustic emission sensor were used to collect gear vibration signals, current signals, acoustic signals, and acoustic emission signals under different condition patterns of the sun gear. As shown in Fig. 7(b)–(h), seven fault conditions were simulated in the experiments, i.e., (b) slight single-point pitting, (c) serious single-point pitting, (d) half broken tooth, (e) broken tooth, (f) double-point pitting, (g) single-point pitting + half broken



(a) (b)

Fig. 8. Comparative results. (a) Diagnosis accuracy of all the algorithms on WT Dataset_1. (b) Diagnosis accuracy of all the algorithms on WT Dataset_2.

TABLE V
TRAINING TIME OF ALL THE ALGORITHMS ON WT DATASETS

Algorithm	WT Dataset_1 (s)	WT Dataset_2 (s)
ESN	15.267	20.783
SVM	16.016	16.129
SAE	522.403	723.763
DBN	565.257	762.213
CNN	6826.762	6982.265
CSO	6214.382	6682.926
CSOLS-1	4436. 135	4512.356
CSOLS-2	8581.430	8439.264
CSOLS-3	12498.176	12392.972
CSOLS-4	16501.902	16228.071

tooth, and (h) groove. Therefore, considering the NC, we have eight condition patterns in total. In total, 4000 samples are collected under each condition pattern. To exploit the structural information among multiple condition patterns, two datasets (WT Dataset_1 and WT Dataset_2) are designed by combining fault types. WT Dataset_1 consists of six patterns (A, C, E, F, G, and NC), which is used to test the ability of algorithms to diagnose different types of faults. However, WT Dataset_2 contains all the eight patterns, which is more complicated because it contains not only different types of fault data but also different degrees of fault data under the same fault type. Therefore, $4000 \times 6 = 24\,000$ samples are gathered in the WT Dataset_1, whereas $4000 \times 8 = 32\,000$ samples are acquired in the WT Dataset_2. These data points were transformed into frequency domain using the fast Fourier transformation, and 80% of the samples in the dataset were randomly taken for training and the remaining for test.

The parameter setting of each algorithm in the experiments was also determined by employing the parameter tuning method in the previous section. In this case, for ESN, the number of reservoir nodes was changed to 300 and 600 for WT Dataset_1 and WT Dataset_2, respectively. For SAE, the hidden layer structure was set as [200] and [150-80] for WT Dataset_1 and WT Dataset_2, respectively. Moreover, the number of iterations of SAE was altered as 200. The hidden layer structure of DBN for WT Dataset_2 was changed to [80-40]. In CNN, the hidden layer structure was set as [conv(8)-pool(3)-conv(10)-pool(4)-conv(8)-pool(4)] and [conv(10)-pool(3)-conv(10)-pool(4)-conv(6)-pool(4)] for WT Dataset_1 and WT Dataset_2, respectively. Diagnosis accuracy results using all the compared algorithms are shown in Fig. 8, and their training times are reported in Table V. The results show that the fault diagnosis data of Case II is more difficult to process than Case I. However, similar comparative effect with Case I can still be gained. This validates the effectiveness and the robustness of our proposed method. Note that the best performer and the second best performer on WT Dataset_2 are CSOLS-3 and CSOLS-4, respectively, which shows the superiority of evolving deep ESN in dealing with complicated fault diagnosis problems to some extent.

V. CONCLUSION

In this article, a CSO combined with an LS (CSOLS) was reported for optimizing parameter values and hyperparameter settings of ESNs simultaneously. An indirect encoding approach was introduced in CSOLS to reduce the number of decision variables, and the LS designed based on the echo state property and separability property was used to accelerate the local optimization. Based on the proposed CSOLS, a layerwise optimization strategy was subsequently designed for evolving a novel type of deep ESN. Four datasets related to the fault diagnosis of 3-D printer and wind turbine gearbox were collected and used to test the performance of the addressed approach. Through comparing with the ESN, SVM, SAE, DBN, CNN, and CSO, the evolved deep ESNs can always obtain the highest accuracy (97.11%, 97.37%, 96.54%, and 93.67%) on each dataset, which demonstrates that the proposed approach used for evolving deep ESNs has promising performance for complicated fault diagnosis problems.

REFERENCES

[1] S. Yin, S. X. Ding, X. C. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6418–6428, Nov. 2014.

[2] C. Li, J. V. de Oliveira, M. Lozada, D. Cabrera, R. V. Sanchez, and G. Zurita, "A systematic review of fuzzy formalisms for bearing fault diagnosis," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 7, pp. 1362–1382, Jul. 2019.

[3] A. Widodo and B. S. Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *Mech. Syst. Signal Process.*, vol. 21, pp. 2560–2574, 2007.

[4] I. Bandyopadhyay, P. Purkait, and C. Koley, "Performance of a classifier based on time-domain features for incipient fault detection in inverter drives," *IEEE Trans. Ind. Inform.*, vol. 15, no. 1, pp. 3–14, Jan. 2019.

[5] N. Saravanan and K. I. Ramachandran, "Incipient gear box fault diagnosis using discrete wavelet transform (DWT) for feature extraction and classification using artificial neural network (ANN)," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4168–4181, 2010.

[6] J. Ben Ali, N. Fnaiech, L. Saidi, B. Chebel-Morello, and F. Fnaiech, "Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals," *Appl. Acoust.*, vol. 89, pp. 16–27, 2015.

[7] B. Zhang, C. Sconyers, C. Byington, R. Patrick, M. E. Orchard, and G. Vachtsevanos, "A probabilistic fault detection approach: Application to bearing fault detection," *IEEE Trans. Ind. Electron.*, vol. 58, no. 5, pp. 2011–2018, May 2011.

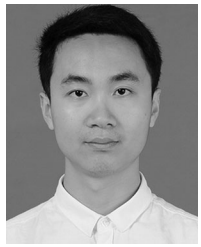
[8] Y. L. He, R. Wang, S. Kwong, and X. Z. Wang, "Bayesian classifiers based on probability density estimation and their applications to simultaneous fault diagnosis," *Inf. Sci.*, vol. 259, pp. 252–268, 2014.

[9] C. Li *et al.*, "A comparison of fuzzy clustering algorithms for bearing fault diagnosis," *J. Intell. Fuzzy Syst.*, vol. 34, pp. 3565–3580, 2018.

[10] J. Tian, C. Morillo, M. H. Azarian, and M. Pecht, "Motor bearing fault detection using spectral kurtosis-based feature extraction coupled with K-nearest neighbor distance analysis," *IEEE Trans. Ind. Electron.*, vol. 63, no. 3, pp. 1793–1803, Mar. 2016.

[11] W. J. Sun, S. Y. Shao, R. Zhao, R. Q. Yan, X. W. Zhang, and X. F. Chen, "A sparse auto-encoder-based deep neural network approach for induction motor faults classification," *Measurement*, vol. 89, pp. 171–178, 2016.

[12] S. Lu, J. Feng, H. Zhang, J. Liu, and Z. Wu, "An estimation method of defect size from MFL image using visual transformation convolutional neural network," *IEEE Trans. Ind. Inform.*, vol. 15, no. 1, pp. 213–224, Jan. 2019.
[13] F. Xu and P. W. Tse, "Combined deep belief network in deep learning with affinity propagation clustering algorithm for roller bearings fault diagnosis without data label," *J. Vib. Control*, vol. 25, pp. 473–482, 2019.
[14] H. Liu, J. Z. Zhou, Y. Zheng, W. Jiang, and Y. C. Zhang, "Fault diagnosis of rolling bearings with recurrent neural network based autoencoders," *ISA Trans.*, vol. 77, pp. 167–178, 2018.
[15] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 89–103, Feb. 2019.
[16] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks," German Nat. Res. Center Inf. Technol., Bonn, Germany, Tech. Rep. GMD 148, 2001.
[17] A. Sharma and R. Rani, "An optimized framework for cancer classification using deep learning and genetic algorithm," *J. Med. Imag. Health Inform.*, vol. 7, pp. 1851–1856, 2017.
[18] Y. Y. Zhang, X. Y. Li, L. Gao, and P. G. Li, "A new subset based deep feature learning method for intelligent fault diagnosis of bearing," *Expert Syst. Appl.*, vol. 110, pp. 125–142, 2018.
[19] H. Badem, A. Basturk, A. Caliskan, and M. E. Yuksel, "A new efficient training strategy for deep neural networks by hybridization of artificial bee colony and limited-memory BFGS optimization algorithms," *Neurocomputing*, vol. 266, pp. 506–526, 2017.
[20] S. Zhong, X. Xie, L. Lin, and F. Wang, "Genetic algorithm optimized double-reservoir echo state network for multi-regime time series prediction," *Neurocomputing*, vol. 238, pp. 191–204, 2017.
[21] H. S. Wang and X. F. Yan, "Optimizing the echo state network with a binary particle swarm optimization algorithm," *Knowl.-Based Syst.*, vol. 86, pp. 182–193, 2015.
[22] L. Wang, H. L. Hu, X. Y. Ai, and H. Liu, "Effective electricity energy consumption forecasting using echo state network improved by differential evolution algorithm," *Energy*, vol. 153, pp. 801–815, 2018.
[23] S. Basterrech, E. Alba, and V. Snasel, "An experimental analysis of the echo state network initialization using the particle swarm optimization," in *Proc. IEEE 6th World Congr. Nature Biol. Inspired Comput.*, 2014, pp. 13–22.
[24] 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.
[25] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.
[26] Z. K. Malik, A. Hussain, and Q. J. Wu, "Multilayered echo state machine: A novel architecture and algorithm," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 946–959, Apr. 2017.
[27] S. Zhang, Z. Sun, M. Wang, J. Long, Y. Bai, and C. Li, "Deep fuzzy echo state networks for machinery fault diagnosis," *IEEE Trans. Fuzzy Syst.*, to be published, doi: 10.1109/TFUZZ.2019.2914617.
[28] J. Long, Z. Sun, P. M. Pardalos, Y. Hong, S. Zhang, and C. Li, "A hybrid multi-objective genetic local search algorithm for the prize-collecting vehicle routing problem," *Inf. Sci.*, vol. 478, pp. 40–61, 2019.



Jianyu Long received the B.E. and Ph.D. degrees in metallurgical engineering from Chongqing University, Chongqing, China, in 2012 and 2017, respectively.

He is currently a Teacher of Industrial Engineering with the School of Mechanical Engineering, Dongguan University of Technology, Dongguan, China. From 2015 to 2016, he was a Visiting Scholar with the Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA. His research interests include production scheduling, and prognostics and system health management.

Shaohui Zhang received the B.E. degree in vehicle engineering from Huaqiao University, Quanzhou, China, in 2009, and the Ph.D. degree in vehicle engineering from the South China University of Technology, Guangzhou, China, in 2014.

He is currently a Teacher of Mechanical Engineering with the School of Mechanical Engineering, Dongguan University of Technology, Dongguan, China. His research interests include machine learning, fault diagnosis, and remaining useful life prediction.

Chuan Li received the Ph.D. degree in mechatronics engineering from Chongqing University, Chongqing, China, in 2007.

He was a Postdoctoral Fellow with the University of Ottawa, Ottawa, ON, Canada, a Research Professor with Korea University, Seoul, South Korea, and a Senior Research Associate with the City University of Hong Kong, Hong Kong. He is currently a Professor of Mechanical Engineering with the School of Mechanical Engineering, Dongguan University of Technology, Dongguan, China. His research interests include prognostics and health management, and intelligent systems.