Research Article

Deep Network Based on Stacked Orthogonal Convex Incremental ELM Autoencoders

Chao Wang, Jianhui Wang, and Shusheng Gu

College of Information Science and Engineering, Northeastern University, Shenyang 110819, China

Correspondence should be addressed to Chao Wang; supper_king1018@163.com

Received 1 March 2016; Revised 7 May 2016; Accepted 31 May 2016

Academic Editor: Lotfi Senhadji

Copyright © 2016 Chao Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Extreme learning machine (ELM) as an emerging technology has recently attracted many researchers' interest due to its fast learning speed and state-of-the-art generalization ability in the implementation. Meanwhile, the incremental extreme learning machine (I-ELM) based on incremental learning algorithm was proposed which outperforms many popular learning algorithms. However, the incremental algorithms with ELM do not recalculate the output weights of all the existing nodes when a new node is added and cannot obtain the least-squares solution of output weight vectors. In this paper, we propose orthogonal convex incremental learning machine (OCI-ELM) with Gram-Schmidt orthogonalization method and Barron's convex optimization learning method to solve the nonconvex optimization problem and least-squares solution problem, and then we give the rigorous proofs in theory. Moreover, in this paper, we propose a deep architecture based on stacked OCI-ELM autoencoders according to stacked generalization philosophy for solving large and complex data problems. The experimental results verified with both UCI datasets and large datasets demonstrate that the deep network based on stacked OCI-ELM autoencoders (DOC-IELM-AEs) outperforms the other methods mentioned in the paper with better performance on regression and classification problems.

1. Introduction

Extreme learning machine (ELM) proposed by Huang et al. [1, 2] is a specific type of single-hidden layer feedforward network (SLFN) with randomly generated additive or RBF hidden nodes and hidden node parameters, which has recently been extensively studied by many researchers in various areas of scientific research and engineering due to the excellent approximation capability. Wang et al. presented ASLGEM-ELM algorithm, which provides some useful guidelines for improving the generalization ability of SLFNs trained with ELM [3]. Alongside probing deeply into the research of theory and its application, ELM has become one of the leading trends for fast learning [4–7]. Recently, Huang et al. [8] have proposed an algorithm called incremental extreme learning machine (I-ELM) which randomly adds nodes to the hidden layer one by one and freezes the output weights of the existing hidden nodes when a new hidden node is added [9–12]. Then, Huang et al. [13] also showed its universal approximation capability for the case of fully complex hidden nodes. I-ELM is fully automatically implemented and in

theory no intervention is required for the learning process from users. But there still exist some issues to be tackled [14]:

- (1) The redundant nodes can be generated in I-ELM, which have a minor effect on the outputs of the network. Moreover, the existence of redundant nodes can eventually increase the complexity of the network.
- (2) The convergence rate of I-ELM is slower than ELM, and the number of hidden nodes in I-ELM is sometimes larger than the dimension of samples for the training.

In this paper, we propose a method called orthogonal convex extreme learning machine (OCI-ELM) to further settle the aforementioned problems of I-ELM. With the rigorous proofs in theory, we can obtain the least-squares solution of $H\beta = T$ and faster convergence rate by adopting the Gram-Schmidt orthogonalization method incorporated into CI-ELM [15]. The simulations on real-world datasets show that the proposed OCI-ELM algorithm can achieve faster convergence rates, more compact neural network, and

better generalization performance than both I-ELM and the improved I-ELM algorithms while keeping the simplicity and efficiency of ELM.

Recently, deep learning has attracted many research interests with its remarkable success in many applications [16–18]. Deep learning is an artificial neural network learning algorithm which has multilayer perceptrons. Deep learning has achieved an approximation of complex functions and alleviated the optimization difficulty associated with the deep models [19–21]. Motivated by the remarkable success of deep learning [22, 23], we propose a new stacked architecture to solve large and complex data problems using OCI-ELM autoencoder as the training algorithm in each layer, which incorporates the excellent performance of OCI-ELM with the ability of complex function approximation derived from deep architectures. We implemented OCI-ELM autoencoder in each iteration of deep orthogonal convex incremental extreme machine (DOC-IELM) to reconstruct the input data and estimate the errors of the prediction functions with the scheme of layer-by-layer architectures. Both the supervised and the unsupervised data all can be the pertaining input of the new proposed deep network. Moreover, the OCI-ELM autoencoder-based deep network (DOC-IELM-AEs) can suffice to achieve the efficiency improvement for generalization performance.

To show the effectiveness of DOC-IELM-AEs, we apply it to both the ordinary real-world datasets with UCI datasets and large datasets with MNIST, OCR Letters, NORB, and USPS datasets. The simulations show that the proposed deep model possesses better accuracy of testing and more compact network architecture than the aforementioned improved I-ELM and other deep models without incurring the out-of-memory problem.

This paper is organized as follows. Section 2 reviews the preliminary knowledge of incremental extreme learning machine (I-ELM). Section 3 describes OCI-ELM algorithm, the proposed model which adopts the Gram-Schmidt orthogonalization method into convex I-ELM (CI-ELM). Section 4 makes a comparison between OCI-ELM and other algorithms. Section 5 presents the details of DOC-IELM-AEs algorithm and compares the performance with deep architecture models. Section 6 applies the DOC-IELM-AEs algorithm into elongation prediction of strips. Finally, Section 7 concludes this paper.

2. Related Works

In this section, the main concepts and theory of the I-ELM [8] algorithm are shortly reviewed. For the sake of generality, we assume that the network has only one linear output node, and all the analysis can be easily extended into multilinear output nodes cases. Consider a training dataset $\mathcal{N} = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, 1 \leq i \leq N\}$; the SLFN with L additive hidden nodes and activation function $\sigma(x)$ can be represented by

$$f_n(x) = \sum_{i=1}^L \beta_i \sigma(a_i \cdot x_j + b_j), \quad j = 1, 2, \dots, N, \quad (1)$$

where $a_i = [a_{i1}, a_{i2}, \dots, a_{in}]^T$ is the weight vector connecting the input layer to the i th hidden node, $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$ is the weight connecting the i th hidden node to the output node, b_i is the threshold of the i th hidden node, and σ is the hidden node activation function.

The I-ELM proposed by Huang et al. is different from the conventional ELM algorithm; I-ELM is an automatic algorithm which can randomly add hidden nodes to the network one by one and freeze all the weights of the existing hidden nodes when a new hidden node is added, until the expected learning accuracy is obtained or the maximum number of hidden nodes is reached. Thus, I-ELM algorithm can be summarized in Algorithm 1.

Algorithm 1 (incremental extreme learning machine (I-ELM)). Given a training dataset $\mathcal{N} = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, 1 \leq i \leq N\}$, activation function $g(x)$, number of hidden nodes L , expected learning accuracy ε , and maximum number of hidden nodes $L_{\max}$, one has the following.

Step 1 (initialization). Let $L = 0$ and residual error $E = t$, where $t = [t_1, t_2, \dots, t_N]^T$.

Step 2 (learning step). While $L < L_{\max}$, $\|E\| > \varepsilon$,

- (a) increase the number of hidden nodes by one;
- (b) assign random input weight w and bias b for hidden nodes L ;
- (c) calculate the residual error after adding the new hidden node;
- (d) calculate the output weight β for the new hidden nodes: $\beta = \langle E, H_L \rangle / \|H_L\|^2$;
- (e) calculate the residual error: $E = E - \beta \cdot H_L$;

Endwhile.

3. The Proposed Orthogonal Convex Incremental Extreme Learning Machine (OCI-ELM)

The motivation for the work in this section comes from the important properties of basic ELM as follows:

- (1) The special solution $\hat{\beta} = H^\dagger T$ is one of the least-squares solutions of a general linear system $H\beta = T$, meaning that the smallest training error can be reached by this special solution: $\|H\hat{\beta} - T\| = \|HH^\dagger T - T\| = \min_{\beta} \|H\beta - T\|$.
- (2) The smallest norm of weights: the special solution $\hat{\beta} = H^\dagger T$ has the smallest norm among all the least-squares solutions of $H\beta = T$:

$$\|\hat{\beta}\| = \|H^\dagger T\| < \|\beta\| \quad (2)$$

$$\forall \beta \in \{\beta : \|H\beta - T\| \leq \|H\hat{\beta} - T\|, \forall \hat{\beta} \in R^{\tilde{N} \times N}\}.$$

- (3) The minimum norm least-squares solution of $H\beta = T$ is unique, which is $\hat{\beta} = H^\dagger T$.

In this section, we propose an improved I-ELM algorithm (OCI-ELM) based on Gram-Schmidt orthogonalization method combined with Barron's convex optimization learning method and prove the OCI-ELM algorithm in theory which can obtain the least-squares solution of $H\beta = T$. Meanwhile, OCI-ELM can achieve a more compact network architecture, faster convergence rate, and better generalization performance than other improved I-ELM algorithms while retaining the I-ELM's simplicity and efficiency.

Theorem 2. Gram-Schmidt orthogonalization process converts linearly independent vectors into orthogonal vectors [24]. Given a linearly independent vector set $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ in the inner product space $V \subseteq R^n$, the vector set $\beta_1, \beta_2, \dots, \beta_n$ for Gram-Schmidt orthogonalization process is as follows [25]:

$$\begin{aligned}\beta_1 &= \alpha_1, \\ \beta_2 &= \alpha_2 - \frac{\langle \beta_1, \alpha_2 \rangle}{\langle \beta_1, \beta_1 \rangle} \beta_1, \\ \beta_3 &= \alpha_3 - \frac{\langle \beta_1, \alpha_3 \rangle}{\langle \beta_1, \beta_1 \rangle} \beta_1 - \frac{\langle \beta_2, \alpha_3 \rangle}{\langle \beta_2, \beta_2 \rangle} \beta_2, \\ \beta_k &= \alpha_k - \frac{\langle \beta_1, \alpha_k \rangle}{\langle \beta_1, \beta_1 \rangle} \beta_1 - \frac{\langle \beta_2, \alpha_k \rangle}{\langle \beta_2, \beta_2 \rangle} \beta_2 - \dots \\ &\quad - \frac{\langle \beta_{k-1}, \alpha_k \rangle}{\langle \beta_{k-1}, \beta_{k-1} \rangle} \beta_{k-1},\end{aligned}\quad (3)$$

where $\alpha_1, \alpha_2, \dots, \alpha_n$ is the set of standardized vectors and $\beta_1, \beta_2, \dots, \beta_n$ form an orthogonal set with the same linear span. For each index $k = 1, 2, \dots, n$, $\text{span}(\beta_1, \beta_2, \dots, \beta_n) = \text{span}(\alpha_1, \alpha_2, \dots, \alpha_n)$.

Theorem 3. Given an orthogonal vector set $[v_1, v_2, \dots, v_{k-1}]$ in the inner product space $V \subseteq R^n$, if vector g_k can be expressed as a linear representation of $v_1, v_2, \dots, v_{k-1}$, one has

$$\begin{aligned}v_k &= g_k - \frac{\langle v_1, g_k \rangle}{\langle v_1, v_1 \rangle} v_1 - \frac{\langle v_2, g_k \rangle}{\langle v_2, v_2 \rangle} v_2 - \dots \\ &\quad - \frac{\langle v_{k-1}, g_k \rangle}{\langle v_{k-1}, v_{k-1} \rangle} v_{k-1} = 0.\end{aligned}\quad (4)$$

Proof. Given the vector set $v_1, v_2, \dots, v_{k-1}$ and vector g_k , suppose there exist scalars $c_1, c_2, \dots, c_{k-1}$; then the linear combination of those vectors with those scalars as coefficients is

$$g_k = c_1 v_1 + c_2 v_2 + \dots + c_{k-1} v_{k-1}. \quad (5)$$

Substituting (4) into (5), we have $v_k = 0$:

$$\beta_n = \frac{\langle e_{n-1}, g_n - f_{n-1} \rangle}{\|g_n - f_{n-1}\|^2}. \quad (6)$$

□

CI-ELM was originally proposed by Huang and Chen [15], which incorporates Barron's convex optimization learning method into I-ELM. By recalculating the output weights of the existing hidden nodes randomly generated after a new node is added, the CI-ELM can obtain better performance than I-ELM. Incorporated with Gram-Schmidt orthogonalization and Barron's convex optimization learning method, the process of OCI-ELM algorithm can be described in Algorithm 4.

Algorithm 4 (orthogonal convex incremental extreme learning machine (OCI-ELM)). Given a training dataset $\aleph = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1, 2, \dots, N\}$, where $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$ and $t_i = [t_{i1}, t_{i2}, \dots, t_{im}]$, and given activation function $\sigma(x)$, maximum number of iterations $K_{\max}$, and expected learning accuracy ε , one has the following.

Step 1 (initialization). Let the number of initial hidden nodes $L = 0$, the number of iterations $K = 0$, and residual error $E = t$, where $t = [t_1, t_2, \dots, t_N]^T$.

Step 2. This step consists of two steps as follows.

Orthogonalization Step. In this step, the following is carried out:

- Increase the number of hidden nodes L and K by one, respectively: $L = L + 1$ and $K = K + 1$.
- Randomly assign hidden node parameters (a_L, b_L) for new hidden node L and calculate the output g_k ,

$$\begin{aligned}v_k &= g_k - \frac{\langle v_1, g_k \rangle}{\langle v_1, v_1 \rangle} v_1 - \frac{\langle v_2, g_k \rangle}{\langle v_2, v_2 \rangle} v_2 - \dots \\ &\quad - \frac{\langle v_{k-1}, g_k \rangle}{\langle v_{k-1}, v_{k-1} \rangle} v_{k-1},\end{aligned}\quad (7)$$

and the hidden layer output matrix V ,

$$\begin{aligned}V &= \begin{bmatrix} V_1 [\sigma(w_1 \cdot x_1 + b_1)] & \dots & V_m [\sigma(w_m \cdot x_1 + b_m)] \\ \vdots & \dots & \vdots \\ V_1 [\sigma(w_1 \cdot x_n + b_1)] & \dots & V_m [\sigma(w_m \cdot x_n + b_m)] \end{bmatrix}_{n \times m}.\end{aligned}\quad (8)$$

Learning Step. While $L_1 < L_{\max}$, $\|E\| > \varepsilon$,

- calculate the output weight β_k for the newly added hidden node:

$$\beta_k = \frac{E \cdot [E - (F - V_k)]^T}{[E - (F - V_k)] \cdot [E - (F - V_k)]^T}; \quad (9)$$

- recalculate the output weight vectors of all existing hidden nodes if $L > 1$:

$$\beta_l = (1 - \beta_k) \beta_l, \quad l = 1, 2, \dots, L - 1; \quad (10)$$

- (e) calculate the residual error after adding the new hidden node L :

$$E = (1 - \beta_k) E + \beta_k (F - V_k); \quad (11)$$

Endwhile.

The rigorous proof on the conclusion is detailedly discussed where OCI-ELM can obtain the least-squares solution of $H\beta = T$.

Theorem 5. Given a training dataset $\aleph = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1, 2, \dots, N\}$ and number of hidden nodes L , where $x_i = [x_1, x_2, \dots, x_{in}]^T$ and $t_i = [t_1, t_2, \dots, t_{im}]^T$, the hidden layer output matrix is $G_L = [g_1, g_2, \dots, g_L] \in R^{n \times L}$, and the matrix of the output weights from the hidden nodes to the output nodes is $\beta_L = [\beta_1, \beta_2, \dots, \beta_L]^T \in R^L$. Let $e_n = e_{n-1} - \beta_n(g_n - f_{n-1})$ denote the residual error function, and $e_0 = T$, $\beta_k = \{\beta^* \mid \min \|e_k\|\} = \{\beta^* \mid \min \|f - ((1 - \beta_k)f_{k-1} + \beta_k g_k)\|$ holds with probability one if $e_L \perp \text{span}\{g_1, g_2, \dots, g_L\}$ and $\beta_k = \langle e_{k-1}, g_k - f_{k-1} \rangle / \|g_k - f_{k-1}\|^2$ for all $k = 1, 2, \dots, L$.

Proof. The proof consists of two steps:

- Firstly, we prove $e_L \perp \text{span}\{g_1, g_2, \dots, g_L\}$.
- And then, we further prove $\beta_k = \{\beta^* \mid \min \|f - ((1 - \beta_k)f_{k-1} + \beta_k g_k)\|$.

(a) According to the condition given above, we have the following:

- Here,

$$\langle e_1, g_1 \rangle = \langle T - \beta_1 g_1, g_1 \rangle = \langle T, g_1 \rangle - \beta_1 \langle g_1, g_1 \rangle = 0. \quad (12)$$

- When the output weight $\beta = \langle e_1, g_2 - f_1 \rangle / \|g_2 - f_1\|^2$, we have

$$\begin{aligned} \langle e_2, g_1 \rangle &= \langle f - f_2, g_1 \rangle = \langle e_1, \beta_2 (g_2 - f_1), g_1 \rangle \\ &= \langle e_1, g_1 \rangle \beta_2 \langle g_2 - f_1, g_1 \rangle \\ &= \langle e_1, g_1 \rangle - \frac{\langle e_1, g_2 - f_1 \rangle}{\|g_2 - f_1\|^2} \langle g_2 - f_1, g_1 \rangle \\ &= \langle e_1, g_1 \rangle - \langle e_1, g_1 \rangle = 0. \end{aligned} \quad (13)$$

- When the output weight $\beta = \langle e_1, g_2 - f_1 \rangle / \|g_2 - f_1\|^2$, we also have

$$\begin{aligned} \langle e_2, g_2 \rangle &= \langle f - f_2, g_2 \rangle = \langle e_1 - \beta_2 (g_2 - f_1), g_2 \rangle \\ &= \langle e_1, g_2 \rangle - \frac{\langle e_1, g_2 - f_1 \rangle}{\|g_2 - f_1\|^2} \langle g_2 - f_1, g_2 \rangle \\ &= \langle e_1, g_2 \rangle - \langle e_1, g_2 \rangle = 0. \end{aligned} \quad (14)$$

- When the output weight $\beta_k = \langle e_{k-1}, g_k - f_{k-1} \rangle / \|g_k - f_{k-1}\|^2$, suppose that, for all $k \leq l$, we have

$$\begin{aligned} \langle e_k, g_k \rangle &= \langle e_{k-1} - \beta_k (g_k - f_{k-1}), g_k \rangle \\ &= \langle e_{k-1}, g_k \rangle - \beta_k \langle g_k - f_{k-1}, g_k \rangle \\ &= \langle e_{k-1}, g_k \rangle \\ &\quad - \frac{\langle e_{k-1}, g_k - f_{k-1} \rangle}{\|g_k - f_{k-1}\|^2} \langle g_k - f_{k-1}, g_k \rangle \\ &= \langle e_{k-1}, g_k \rangle - \langle e_{k-1}, g_k \rangle = 0. \end{aligned} \quad (15)$$

- When the output weight $\beta_k = \langle e_{k-1}, g_k - f_{k-1} \rangle / \|g_k - f_{k-1}\|^2$, suppose that, for all $j \leq k-1$, we have

$$\begin{aligned} \langle e_k, g_j \rangle &= \langle e_{k-1} - \beta_k (g_k - f_{k-1}), g_j \rangle \\ &= \langle e_{k-1}, g_j \rangle - \beta_k \langle g_k - f_{k-1}, g_j \rangle \\ &= \langle e_{k-1}, g_j \rangle \\ &\quad - \frac{\langle e_{k-1}, g_k - f_{k-1} \rangle}{\|g_k - f_{k-1}\|^2} \langle g_k - f_{k-1}, g_j \rangle \\ &= \langle e_{k-1}, g_j \rangle - \langle e_{k-1}, g_j \rangle = 0. \end{aligned} \quad (16)$$

So, $e_k \perp \text{span}\{g_1, g_2, \dots, g_k\}$; that is, $e_L \perp \text{span}\{g_1, g_2, \dots, g_L\}$. Therefore,

$$G_L^T (T - G_L (1 - \beta_L) \beta^*) = 0. \quad (17)$$

- According to (17), we have $(\beta_L - \beta^*)^T G_L^T (T - G_L (1 - \beta_L) \beta^*) = 0$, where $\beta^* \in R^{L \times 1}$ is arbitrary; then, we have

$$\begin{aligned} &\|T - G_L (1 - \beta_L) \beta^*\|^2 \\ &\leq \|T - G_L (1 - \beta_L) \beta_L\|^2 + \|G_L (\beta_L - \beta^*)\|^2 \\ &= \|T - G_L (1 - \beta_L) \beta_L + G_L (\beta_L - \beta^*)\|^2 \\ &= \|T - G_L (\beta^* - \beta_L \cdot \beta_L)\|^2, \\ &T - G_L (1 - \beta_L) \beta^* = T - G_L (\beta^* - \beta_L \cdot \beta_L), \\ &\beta^* - \beta_L \cdot \beta^* = \beta^* - \beta_L \cdot \beta_L, \\ &\beta_L = \beta^*. \end{aligned} \quad (18)$$

And $\|T - G_L (1 - \beta_L) \beta^*\| = \|T - G_L (1 - \beta_L) \beta_L\|$ holds only if $\beta_L = \beta^*$. Therefore, $\beta_k = \{\beta^* \mid \min \|f - ((1 - \beta_k)f_{k-1} + \beta_k g_k)\|$. $\square$

4. Experiments and Analysis

In this section, we tested the generalization performance of the proposed OCI-ELM with other similar learning algorithms on ten UCI real-world datasets, including five regression and five classification problems, as shown in Table 1. The

TABLE 1: Specification of 10 benchmark problems.

Datasets	Training sample	Testing sample	Attributes	Type
Auto MPG	182	165	8	Regression
California Housing	8,800	8,260	8	Regression
Servo	92	60	4	Regression
Concrete Compressive Strength	975	830	9	Regression
Parkinsons	4,780	5,120	26	Regression
Delta Ailerons	5,080	4,600	6	Classification
Waveform II	3,300	2,800	40	Classification
Abalone	3,900	3,670	8	Classification
Breast Cancer	539	490	10	Classification
Energy Efficiency	740	600	8	Classification

simulations are conducted in MATLAB 2013a environment running on Windows 7 machine with 32 GB of memory and i7-990X (3.46 GHz) processor.

The experimental results between OCI-ELM and some other ELM algorithms on regression and classification problems are given in Tables 2 and 3. In Tables 2 and 3, the best results obtained by the OCI-ELM and the other 4 algorithms are italicized and shown in boldface. In Section 4.1, we compare the generalization performance of OCI-ELM with another six state-of-the-art algorithms on regression problems. In Section 4.2, we compare the generalization performance of OCI-ELM with the same six algorithms on classification problems. All these results in this section are obtained from thirty trials for all cases, and the mean results (mean), root-mean-square errors (RMSE), and standard deviations (Std.) are listed in the corresponding tables, respectively. The seven representative evolutionary algorithms are listed as follows:

- (i) Convex incremental extreme learning machine (CI-ELM) [15].
- (ii) Parallel chaos search based incremental extreme learning machine (PC-ELM) [26].
- (iii) Leave-one-out incremental extreme learning machine (LOO-IELM) [27].
- (iv) Sparse Bayesian extreme learning machine (SB-ELM) [28].
- (v) Improved incremental regularized extreme learning machine (II-RELM) [11].
- (vi) Enhancement incremental regularized extreme learning machine (EIR-ELM) [12].

4.1. Performance Comparison of Regression Problems. In this section, datasets Auto MPG, California Housing, Servo, CCS (Concrete Compressive Strength), and Parkinsons are conducted for the regression problems. Table 2 shows the RMSE of the training and testing with fixed hidden nodes obtained from OCI-ELM and another six algorithms, respectively. Meanwhile, hidden nodes and learning time with the same stop RMSE are also shown in Table 2. For California Housing dataset in the table, the OCI-ELM provides lower training and testing RMSE rate (0.1272 and 0.1263) than

CI-ELM (0.1601 and 0.1583), PC-ELM (0.1389 and 0.1377), LOO-IELM (0.1376 and 0.1374), SB-ELM (0.1363 and 0.1369), II-RELM (0.1341 and 0.1339), and EIR-ELM (0.1274 and 0.1268) with fixed nodes (20). For the stop criterion of RMSE 0.12, OCI-ELM also exhibits more compact network architecture with 127.15 nodes and faster speed with 0.9704 s, and the nodes and training speed of other algorithms are, respectively, 330.09 and 1.0051; 199.34 and 0.9810; 217.08 and 0.9766; and 192.33 and 0.9713. Where SB-ELM is the fixed ELM, thus there is a difficulty in finding the accurate stop criterion. Likewise, in the CCS dataset, the hidden node of SB-ELM is an approximate value; meanwhile, OCI-ELM shows better generalization performance than other algorithms in comparisons. Although, in the cases of Auto MPG, Servo, and Parkinsons, the learning time consumed by OCI-ELM shows that the presented algorithm is not the top spot, the average convergence rate of five regression problems consumed by OCI-ELM is still the fastest. Moreover, the average convergence rate demonstrates that the stability performance of OCI-ELM is better than other algorithms. The proposed algorithm can retain the simplicity and efficiency of incremental ELM and obtain the least-squares solution of $H\beta = T$ by incorporating the Gram-Schmidt orthogonalization method. The optimal solution obtained from $H\beta = T$ means that the best hidden node parameter leading to the largest residual error decreasing will be added to the existing network. Therefore, OCI-ELM can efficiently reduce the network complexity and meanwhile enhance the generalization performance of the algorithm.

4.2. Performance Comparison of Classification Problems. In this section, datasets Delta Ailerons, Waveform II, Abalone, Breast Cancer, and Energy Efficiency are conducted for the classification problems. Table 3 shows the comparisons of the classification performance conducted on 5 UCI datasets of classification problems. With the same fixed hidden nodes listed in Table 3, the results of comparisons obtained from OCI-ELM are better than those of the other algorithms. For Waveform II dataset in the table, the OCI-ELM proposed displays better training accuracy and standard deviation of 93.11% and 0.0083 than CI-ELM (84.47% and 0.0182), PC-ELM (89.81% and 0.0104), LOO-IELM (88.93% and 0.0097), SB-ELM (80.69% and 0.0181), II-RELM (90.64%

TABLE 2: The comparisons of training and testing on the regression cases.

Datasets	Approaches	RMSE (training & testing)			Hidden nodes & average time	
		Nodes (fixed)	Training	Testing	# nodes	Time (s)
Auto MPG (0.08)	CI-ELM ⁽²⁰⁰⁷⁾	20	0.1043	0.1035	66.29	0.1485
	PC-ELM ⁽²⁰¹²⁾	20	0.1014	0.1012	34.07	0.2783
	LOO-IELM ⁽²⁰¹⁴⁾	20	0.1106	0.1104	49.91	0.3183
	SB-ELM ⁽²⁰¹⁴⁾	20	0.1376	0.2307	≈130	0.0717
	II-RELM ⁽²⁰¹⁵⁾	20	0.0998	0.1005	44.17	0.3483
	EIR-ELM ⁽²⁰¹⁶⁾	20	0.0893	0.1005	31.05	0.3283
	OCI-ELM	20	0.0827	0.0823	23.62	0.2204
California Housing (0.12)	CI-ELM	150	0.1601	0.1583	330.09	1.0051
	PC-ELM	150	0.1389	0.1377	199.34	0.9810
	LOO-IELM	150	0.1376	0.1374	217.08	0.9766
	SB-ELM	150	0.1363	0.1369	—	—
	II-RELM	150	0.1341	0.1339	192.33	0.9713
	EIR-ELM	150	0.1274	0.1268	184.67	1.0017
	OCI-ELM	150	0.1272	0.1263	172.15	0.9704
Servo (0.115)	CI-ELM	100	0.1428	0.1419	182.63	0.0806
	PC-ELM	100	0.1373	0.1364	160.82	0.0701
	LOO-IELM	100	0.1371	0.1368	155.72	0.0765
	SB-ELM	100	0.1257	0.1254	≈127	0.0355
	II-RELM	100	0.1303	0.1307	157.12	0.0886
	EIR-ELM	100	0.1265	0.1264	147.80	0.0794
	OCI-ELM	100	0.1238	0.1232	143.56	0.0828
CCS (0.035)	CI-ELM	150	0.0611	0.0602	229.86	0.5893
	PC-ELM	150	0.0381	0.0365	162.79	1.1236
	LOO-IELM	150	0.0372	0.0369	159.04	0.9427
	SB-ELM	150	0.0366	0.0368	≈170	0.0872
	II-RELM	150	0.0361	0.0363	163.82	0.8341
	EIR-ELM	150	0.0348	0.0351	145.78	0.6305
	OCI-ELM	150	0.0332	0.0346	130.04	0.5835
Parkinsons (0.14)	CI-ELM	250	0.0913	0.0906	170.02	3.3403
	PC-ELM	250	0.0471	0.0463	63.55	4.7503
	LOO-IELM	250	0.0453	0.0462	59.78	4.4452
	SB-ELM	250	0.0388	0.0391	—	—
	II-RELM	250	0.0389	0.0383	77.19	4.4189
	EIR-ELM	250	0.0344	0.0347	48.92	3.9836
	OCI-ELM	250	0.0301	0.0283	39.92	3.0819

and 0.0112), and EIR-ELM (91.15% and 0.0096), thanks to the better classification ability of OCI-ELM. In addition, the hidden nodes 29.54 and the average time 3.0864 s are also less than others, which can demonstrate that OCI-ELM has more reasonable network structure than CI-ELM, PC-ELM, LOO-IELM, SB-ELM, II-RELM, and EIR-ELM,

which efficiently reduces the complexity of the network. Although SB-ELM shows the advantages in training speed as the fixed ELM, OCI-ELM generally produces better performance in comprehensive consideration of accuracy and speed for practical problems which need higher accuracy demand.

TABLE 3: The comparisons of training and testing on the classification cases.

Datasets	Approaches	Testing accuracy			Hidden nodes & average time	
		Nodes (fixed)	Mean (%)	Std.	# nodes	Time (s)
Delta Ailerons (0.035)	CI-ELM ⁽²⁰⁰⁷⁾	250	83.29	0.0036	369.32	1.3505
	PC-ELM ⁽²⁰¹²⁾	250	90.02	0.0016	35.19	0.6829
	LOO-IELM ⁽²⁰¹⁴⁾	250	91.17	0.0027	41.12	0.7761
	SB-ELM ⁽²⁰¹⁴⁾	250	91.66	0.0071	≈220	0.0556
	II-RELM ⁽²⁰¹⁵⁾	250	91.18	0.0042	51.16	0.7425
	EIR-ELM ⁽²⁰¹⁶⁾	250	92.03	0.0019	34.29	1.1304
	OCI-ELM	250	92.84	0.0012	31.73	0.7021
Waveform II (0.04)	CI-ELM	100	84.47	0.0182	200.11	3.0977
	PC-ELM	100	89.81	0.0104	47.63	3.0954
	LOO-IELM	250	88.93	0.0097	46.44	3.3437
	SB-ELM	250	80.69	0.0181	—	—
	II-RELM	250	90.64	0.0112	44.33	3.6603
	EIR-ELM	250	91.15	0.0096	38.91	3.2267
	OCI-ELM	100	93.11	0.0083	29.54	3.0864
Abalone (0.05)	CI-ELM	150	82.72	0.0022	150.37	0.4930
	PC-ELM	150	93.57	0.0016	24.62	0.6177
	LOO-IELM	250	90.51	0.0033	38.43	0.7102
	SB-ELM	250	86.03	0.0107	≈180	0.0413
	II-RELM	250	92.10	0.0034	35.74	0.6924
	EIR-ELM	250	93.91	0.0018	24.16	0.8533
	OCI-ELM	150	94.24	0.0015	21.41	0.6802
Breast Cancer (0.07)	CI-ELM	200	90.06	0.0145	88.30	0.0804
	PC-ELM	200	93.23	0.0082	34.79	0.0992
	LOO-IELM	250	93.07	0.0104	40.82	0.1102
	SB-ELM	250	94.41	0.0075	≈150	0.0275
	II-RELM	250	92.58	0.0095	55.27	0.1032
	EIR-ELM	250	94.76	0.0078	31.18	0.1174
	OCI-ELM	200	94.73	0.0067	34.58	0.1061
Energy Efficiency (0.055)	CI-ELM	150	91.78	0.0013	61.09	0.2966
	PC-ELM	150	96.53	0.0008	41.08	0.3617
	LOO-IELM	250	95.18	0.0024	27.94	0.3517
	SB-ELM	250	92.16	0.0033	≈150	0.0658
	II-RELM	250	95.29	0.0011	46.83	0.3826
	EIR-ELM	250	96.67	0.0012	22.42	0.4011
	OCI-ELM	150	97.25	0.0008	18.49	0.3979

In short, OCI-ELM can generally achieve better performance on these regression and classification problems in terms of training (and testing) RMSE for regression and testing accuracy for classification. Moreover, the compactness of network and convergence rate also display the good performance of OCI-ELM algorithm.

5. Deep Network Based on Stacked OCI-ELM Autoencoders (DOC-IELM-AEs)

5.1. OCI-ELM Autoencoder. As an artificial neural network model, the autoencoder is frequently applied in the deep architecture approaches. Autoencoder is a kind of unsupervised neural network, where the input of network is equal to the output. Kasun et al. [29] proposed an autoencoder based on ELM (ELM-AE). According to their ELM-AE theory, the model of ELM-AE is composed of input layer, hidden layer, and output layer. In addition, the weights and biases of the hidden nodes are randomly generated via orthogonalization, and the input data is projected to a different or equal dimension space [30]; the expressions are as follows:

$$\begin{aligned} h &= \sigma(a \cdot x + b), \quad a^T a = 1, \quad b^T b = 1, \\ h(x_i) V &= x_i^T, \quad i = 1, 2, \dots, N, \end{aligned} \quad (19)$$

where $a = [a_1, \dots, a_L]^T$ are the weights generated orthogonally randomly and $b = [b_1, \dots, b_L]^T$ are the biases generated orthogonally randomly between the input and hidden nodes. There are three calculation approaches to obtain the output weight β of ELM-AE:

- (1) For sparse ELM-AE representations, output weights β can be calculated as follows:

$$\beta = \left(\frac{I}{C} + H^T H \right)^{-1} H^T X. \quad (20)$$

- (2) For compressed ELM-AE representations, output weights β can be calculated as follows:

$$\beta = H^T \left(\frac{I}{C} + H^T H \right)^{-1} X. \quad (21)$$

- (3) For equal dimension ELM-AE representations, output weights β can be calculated as follows:

$$\beta = H^{-1} X. \quad (22)$$

In this section, we present the OCI-ELM, which is incorporated with Barron's convex optimization learning method and Gram-Schmidt orthogonalization method into I-ELM to achieve the optimal least-squares solution as the training algorithm for an autoencoder instead of conventional autoencoders, which apply backpropagation algorithm (BP) for training to obtain the identity function and normal ELM for training the autoencoder. Because of the adoption of incremental algorithm, there is no need to set the number of hidden nodes according to the experience. With the

initialization of the maximum value $L_{\max}$ of hidden nodes the, number of hidden nodes can be increased by more than one node each time, until the stop criterions are met; for example, residual error E is equal to the expected learning accuracy ε or the number of hidden nodes L achieves $L_{\max}$.

As shown in Figure 1, the model structure of OCI-ELM-AE can randomly control the number of the nodes without the computation accuracy. Given a training dataset $\mathcal{N} = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1, \dots, N\}$, where $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$ and $t_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T$, and given activation function $\sigma(x)$ and maximum number of hidden nodes in single layer $L_{\max}$, the input data is reconstructed at the output layer through the following function:

$$\sum_{i=1}^L \beta_i \sigma(a_i \cdot x_j + b_j) = x_j, \quad j = 1, 2, \dots, L. \quad (23)$$

The output weight can be obtained with the following:

$$\begin{aligned} \beta_{1(L)} &= \frac{E \cdot [E - (F - V_{1(L)})]^T}{[E - (F - V_{1(L)})] \cdot [E - (F - V_{1(L)})]^T}, \\ \beta_{1(i)} &= (1 - \beta_{1(L)}) \beta_{1(i)}, \quad i = 1, 2, \dots, L - 1, \end{aligned} \quad (24)$$

where $a_i = [a_{i1}, a_{i2}, \dots, a_{im}]^T$ is the input weight generated randomly and $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$ are the input and output of the OCI-ELM-AE.

5.2. Implementation of Stacked OCI-ELM Autoencoders in Deep Network. In 2006, Hinton et al. [31] presented the concept of deep learning to solve the problems of unsupervised data. Deep belief nets (DBNs) are probabilistic generative models which are first trained only with unlabeled data and then fine-tuned in a supervised mode. And then, another kind of deep network based on Restricted Boltzmann Machine (RBM) [32], deep Boltzmann machine (DBM) [33], was introduced by Salakhutdinov and Larochelle. The base building block of DBN is RBM. The ML-ELM was presented by Kasun et al. in 2013 [29]. There is no difference between the ML-ELM and the other deep learning models; ML-ELM performs layerwise unsupervised learning to train the parameters with the hidden layer weights which are initialized with ELM-AE, and the ML-ELM does not need to be fine-tuned. The AE-S-ELMs was proposed by Zhou et al. in 2014 [34]. The network consists of multiple ELMs with a small number of hidden nodes in each layer to substitute a single ELM with a large number of hidden nodes, and it implements ELM autoencoder in each iteration of S-ELMs algorithm to further improve the testing accuracy, especially for the unstructured large data without properly selected features.

Algorithm 6 (deep network based on stacked orthogonal convex incremental ELM autoencoders (DOC-IELM-AEs)). Given a training dataset $\mathcal{N} = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1, 2, \dots, N\}$, where $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$ and $t_i = [t_{i1}, t_{im}, \dots, t_{im}]^T$, and given activation function $\sigma(x)$, maximum number of hidden nodes in single layer $L_{\max}$, maximum

FIGURE 1: Network diagram of OCI-ELM-AE.

number of iterations $K_{\max}$, and expected learning accuracy ε , one has the following.

Step 1 (initialization). Let the number of initial hidden nodes $L_{1 \rightarrow \tilde{N}} = 0$, the number of iterations $K_{1 \rightarrow \tilde{N}} = 0$, and residual error $E = t$, where $t = [t_1, t_2, \dots, t_N]^T$.

Step 2 (orthogonal convex I-ELM autoencoder on layer 1). This step consists of two steps as follows.

Orthogonalization Step. In this step, the following is carried out:

- Increase the number of hidden nodes L_1 and K_1 by one, respectively: $L_1 = L_1 + 1$ and $K_1 = K_1 + 1$.
- Randomly assign hidden node parameters (a_L, b_L) for new hidden node L and calculate the output g_k ,

$$v_k = g_k - \frac{\langle v_1, g_k \rangle}{v_1, v_1} v_1 - \frac{\langle v_2, g_k \rangle}{v_2, v_2} v_2 - \dots - \frac{\langle v_{k-1}, g_k \rangle}{v_{k-1}, v_{k-1}} v_{k-1}, \quad (25)$$

and the hidden layer output matrix,

$$V = \begin{bmatrix} V_1 [\sigma(a_1 \cdot x_1 + b_1)] & \dots & V_m [\sigma(a_m \cdot x_1 + b_m)] \\ \vdots & \dots & \vdots \\ V_1 [\sigma(a_1 \cdot x_n + b_1)] & \dots & V_m [\sigma(a_m \cdot x_n + b_m)] \end{bmatrix}_{n \times m}. \quad (26)$$

Learning Step. While $L_1 < L_{\max}$, $\|E\| > \varepsilon$,

- calculate the output weight $\beta_{1(L)}$ for the newly added hidden node:

$$\beta_{1(L)} = \frac{E \cdot [E - (F - V_{1(L)})]^T}{[E - (F - V_{1(L)})] \cdot [E - (F - V_{1(L)})]^T}; \quad (27)$$

- recalculate the output weight vectors of all existing hidden nodes if $L_1 > 1$:

$$\beta_{1(i)} = (1 - \beta_{1(L)}) \beta_{1(i)}, \quad i = 1, 2, \dots, L-1; \quad (28)$$

- calculate the residual error after adding the new hidden node L_1 :

$$E = (1 - \beta_{1(L)}) E + \beta_{1(L)} (F - V_{1(L)}). \quad (29)$$

Step 3 (orthogonal convex I-ELM autoencoder on layer 2 $\rightarrow \tilde{N}$). This step is carried out as follows.

Learning Step. While $L_{2 \rightarrow \tilde{N}} < L_{\max}$, $\|E\| > \varepsilon$,

- calculate the output weight $\beta_{1(L)}$ for the newly added hidden node with the hidden layer output matrix V :

$$\beta_{2 \rightarrow \tilde{N}(L)} = \frac{E \cdot [E - (F - V_{2 \rightarrow \tilde{N}(L)})]^T}{[E - (F - V_{2 \rightarrow \tilde{N}(L)})] \cdot [E - (F - V_{2 \rightarrow \tilde{N}(L)})]^T}; \quad (30)$$

- recalculate the output weight vectors of all existing hidden nodes if $L_{2 \rightarrow \tilde{N}} > 1$:

$$\beta_{2 \rightarrow \tilde{N}(i)} = (1 - \beta_{2 \rightarrow \tilde{N}(L)}) \beta_{2 \rightarrow \tilde{N}(i)}, \quad i = 1, 2, \dots, L-1; \quad (31)$$

- calculate the residual error after adding the new hidden node $L_{2 \rightarrow \tilde{N}}$:

$$E = (1 - \beta_{2 \rightarrow \tilde{N}(L)}) E + \beta_{1(L)} (F - V_{2 \rightarrow \tilde{N}(L)}); \quad (32)$$

Endwhile.

The DOC-IELM-AEs algorithm inherits the advantages of incremental constructive feedforward networks model and deep learning algorithms on exactly capturing higher-level abstractions and characterizing the data representations. The implementation of autoencoders for the unsupervised pretraining of data exhibits super-duper performance on regression and classification problems. The improved method utilizes the OCI-ELM-AE as a base building layer to construct the whole deep architecture. As shown in Figure 2, the data is mapped to OCI-ELM feature space; in each layer, OCI-ELM-AE output weights with respect to input data are the weight of the first layer; for the same reason, the output weights of OCI-ELM-AE, with respect to hidden layer output, are the layer weights of DOC-IELM-AEs. The detailed algorithm of DOC-IELM-AEs is shown in Algorithm 6.

FIGURE 2: The model structure of DOC-IELM-AEs.

5.2.1. Performance Comparison of Regression Problems Based on DOC-IELM-AEs. In this section, we mainly test the regression performance of the proposed OCI-ELM and DOC-IELM-AEs on three UCI real-world datasets, Parkinsons, California Housing, and CCS (Concrete Compressive Strength) data, and two large datasets, BlogFeedback and Online News Popularity data. The simulations are conducted in MATLAB 2013a environment running on Windows 7 machine with 128 GB of memory and Intel Xeon E5-2620V2 (2.1 GHz) processor.

The regression performance comparisons of the proposed algorithms OCI-ELM and DOC-IELM-AEs with the baseline methods including SVM [35], single ELM, ML-ELM, AE-S-ELMs, DBN, ErrCor [36], and PC-ELM are shown in Table 5. The specific analyses on the results of regression capability and effectiveness are as follows:

- (1) OCI-ELM compared with SVM, ELM, ErrCor, and PC-ELM: we perform the regression testing on the datasets described in Table 4. The simulations are obtained by the average of 50 trails; we can observe from Table 4 that the testing accuracies of OCI-ELM on UCI datasets and large datasets are both better than SVM, ELM, ErrCor, and PC-ELM. For BlogFeedback dataset, the training accuracy of OCI-ELM is 91.76%, and those of SVM, ELM, ErrCor, and PC-ELM are 89.75%, 90.12%, 90.39%, and 90.54%, respectively. Meanwhile, OCI-ELM also obtains better testing accuracy of 91.82% than other algorithms. Although the OCI-ELM is an iterative learning

algorithm, the compact neural network makes the convergence rate faster than PC-ELM and ErrCor, merely slower than SVM and ELM. Thus, the training time consumed on OCI-ELM learning is acceptable.

- (2) DOC-IELM-AEs compared with DBN, ML-ELM, and AE-S-ELMs: the testing accuracy on UCI datasets can show that the performance of DOC-IELM-AEs outperforms the OCI-ELM. With the aforementioned comparisons between OCI-ELM and the other algorithms (SVM, ELM, ErrCor, and PC-ELM), evidenced by the same token, the DOC-IELM-AEs can achieve better testing accuracy than SVM, ELM, ErrCor, and PC-ELM; this result can also be seen in Table 5. For the large-scale datasets (BlogFeedback and Online News Popularity), the DOC-IELM-AEs obtained accuracies of 93.16%, 93.27% and 93.69%, 93.84% for training and testing with the network structures 281-1000-1000-2000-10 and 61-700-700-10000-26, respectively. The simulations in Table 5 show that DOC-IELM-AEs can produce better results than DBN, ML-ELM, and AE-S-ELMs. Furthermore, DOC-IELM-AEs enjoys the advantage over the DBN and ML-ELM on training speed. Thus, with the better regression performance, DOC-IELM-AEs would provide the state-of-the-art method for large-scale unstructured data problems.

5.2.2. Performance Comparison of Classification Problems Based on DOC-IELM-AEs. The classification performance

TABLE 4: Specification of 10 benchmark problems.

Datasets	Training sample	Testing sample	Attributes	Type
Parkinsons	4,780	5,120	26	Regression
California Housing	8,800	8,260	8	Regression
Concrete Compressive Strength	975	830	9	Regression
BlogFeedback	58,000	23,000	281	Regression
Online News Popularity	36,800	10,000	61	Regression
Delta Ailerons	5,080	4,600	6	Classification
Waveform II	3,300	2,800	40	Classification
MNIST	35,000	6,000	784	Classification
OCR Letters	40,000	12,000	128	Classification
NORB	20,000	5,000	2,048	Classification

comparisons of the proposed algorithms OCI-ELM and DOC-IELM-AEs with the baseline methods including SVM, single ELM, ML-ELM, AE-S-ELMs, DBN, ErrCor, and PC-ELM are shown in Table 6. The specific comparisons are as follows:

- (1) OCI-ELM compared with SVM, ELM, ErrCor, and PC-ELM: the simulation results are obtained by the average of 50 trails on datasets in Table 4 (from Delta Ailerons to NORB data). For the BlogFeedback, the training and testing accuracies are 91.76% and 91.82%, respectively, listed in Table 6; we can see that OCI-ELM achieves better classification accuracy than SVM, ELM, ErrCor, and PC-ELM. And the speed of learning is faster than other improved ELM algorithms, notwithstanding behind the SVM and single ELM due to the process of iteration learning.
- (2) DOC-IELM-AEs compared with DBN, ML-ELM, and AE-S-ELMs: to test these anticipated effects, we used UCI datasets and large-scale datasets to acquire the results. From the experimental results, we can see that the classification accuracies of DOC-IELM-AEs are better than others obviously. Focusing on NORB, the network structure used by DOC-IELM-AEs is 2048-800-800-3000-5; the DOC-IELM-AEs obtained the best accuracies of 93.16%, 93.27% and 93.69%, 93.84% for training and testing, respectively, in all algorithms, including SVM, single ELM, ML-ELM, AE-S-ELMs, DBN, ErrCor, PC-ELM OCI-ELM, and DOC-IELM-AEs. Furthermore, the simulation results of other datasets also display the outstanding performance of DOC-IELM-AEs. Thus, with the better accuracy and faster speed of training, DOC-IELM-AEs can be applied in the vast majority of classification problems.

6. Case Study on Elongation Prediction of Strips

In this section, all of the experimental results for the elongation of strips prediction are presented. The annealing treatment is considered the most important process to cold rolled

FIGURE 3: General scheme of strip steel in annealing process.

strips. In this process, the cold working hardening and internal stress of strips can be eliminated; the hardness of strips can be reduced; moreover, the ability of plastic deformation, stamping, and mechanical technique can be improved. Figure 3 shows the process of continuous annealing. In the furnace, the strips will pass five temperature sections, that is, preheating section (PHS), heating section (HS), slow cooling section (SS), rapid cooling section (RCS), and equalising section (ES), and three tension sections, that is, SS tension section, RCS tension section, and HS tension section. Therefore, the strips will extend or shorten with the changes of temperature and tension. Meanwhile, the surface friction coefficient and the rotational speed of the tension rolls also affect the elongation of strips, rendering the weld position unable to be tracked accurately, having a great influence on the rate of finished product and the safety of air-knife. Thus, the proposed method DOC-IELM-AEs is applied in the annealing of strips process to obtain the position information of welds. The annealing process has 12 continuous process measurements and 10 manipulated variables according to the experience and mechanism analysis.

We collect the historical records in the last 16 months which can affect the position of the welding seam, including the temperature data of 5 sections, the tension data of 3 sections, and the speed data of 11 sections. We use data of 10 months for training and the following data of 6 months for testing. The comparison results of elongation of strips prediction are shown in Figure 4. From the figures, we can see that the prediction results in the 6 months obtained based on 6 algorithms can all approximate the measured values. Although there are only little differences in the experimental results, the prediction based on DOC-IELM-AEs consistently outperforms the other methods in the comparisons.

TABLE 5: The comparisons of training and testing on the regression cases.

Datasets	Algorithms	Training accuracy (%)	Training time (s)	Testing accuracy (%)	Testing deviation (%)
Parkinsons	SVM	95.42	282.26	95.43	0.01
	ELM	95.75	240.68	95.73	0.29
	ML-ELM	97.93	482.7	97.96	0.08
	AE-S-ELMs	98.07	492.21	98.05	0.11
	DBN	97.51	5013	97.53	0.04
	ErrCor	96.25	344.82	96.19	0.32
	PC-ELM	97.16	362.05	97.18	0.09
	OCI-ELM	97.59	327.2	97.62	0.05
	DOC-IELM-AEs	98.44	440.39	98.41	0.04
California Housing	SVM	96.56	327.17	96.56	0
	ELM	96.79	289.24	96.72	0.16
	ML-ELM	98.12	462.31	98.14	0.12
	AE-S-ELMs	98.26	402.49	98.23	0.09
	DBN	98.03	3989	98.04	0.08
	ErrCor	97.01	401.66	97.11	0.25
	PC-ELM	97.34	396.52	97.31	0.28
	OCI-ELM	97.92	386.63	97.90	0.11
	DOC-IELM-AEs	98.71	447.2	98.72	0.07
Concrete Compressive Strength	SVM	95.72	32.32	95.78	0.04
	ELM	95.91	28.29	96.02	0.06
	ML-ELM	97.26	53.4	97.33	0.18
	AE-S-ELMs	97.49	50.66	97.41	0.12
	DBN	96.64	320.04	96.58	0.05
	ErrCor	96.32	47.45	96.35	0.25
	PC-ELM	96.55	43.97	96.67	0.09
	OCI-ELM	97.04	44.33	97.15	0.06
	DOC-IELM-AEs	98.39	51.58	98.37	0.03
BlogFeedback	SVM	89.75	3906	89.82	0.03
	ELM	90.12	2405	90.14	0.22
	ML-ELM	91.86	5175	91.82	0.12
	AE-S-ELMs	91.83	5247	91.79	0.13
	DBN	90.51	19766	90.60	0.09
	ErrCor	90.39	4889	90.44	0.09
	PC-ELM	90.54	4308	90.58	0.12
	OCI-ELM	91.76	4223	91.82	0.09
	DOC-IELM-AEs	93.16	5271	93.27	0.07
Online News Popularity	SVM	91.75	311.83	91.72	0.04
	ELM	91.68	131.34	91.69	0.34
	ML-ELM	92.62	684.98	92.71	0.17
	AE-S-ELMs	92.59	685.62	95.72	0.12
	DBN	92.23	7062	92.26	0.05
	ErrCor	91.61	541.14	91.77	0.29
	PC-ELM	92.29	576.01	92.35	0.15
	OCI-ELM	92.52	521.24	92.54	0.12
	DOC-IELM-AEs	93.69	634.09	93.84	0.11

TABLE 6: The comparisons of training and testing on the classification cases.

Datasets	Algorithms	Training accuracy (%)	Training time (s)	Testing accuracy (%)	Testing deviation (%)
Delta Ailerons	SVM	95.41	219.54	95.42	0
	ELM	95.78	43.88	95.81	0.22
	ML-ELM	97.32	317.83	97.34	0.19
	AE-S-ELMs	97.75	279.32	97.71	0.27
	DBN	98.34	4432	97.35	0.05
	ErrCor	96.42	234.27	96.48	0.12
	PC-ELM	96.56	262.69	96.61	0.08
	OCI-ELM	96.73	249.89	96.69	0.07
	DOC-IELM-AEs	98.89	350.17	98.90	0.05
Waveform II	SVM	95.45	247.11	95.44	0.01
	ELM	95.71	107.14	95.73	0.16
	ML-ELM	97.83	304.28	97.86	0.11
	AE-S-ELMs	98.09	365.83	98.11	0.15
	DBN	97.59	4101	97.52	0.04
	ErrCor	96.54	261.44	96.58	0.06
	PC-ELM	96.46	293.85	96.47	0.05
	OCI-ELM	97.33	202.08	97.36	0.08
	DOC-IELM-AEs	98.86	332.58	98.85	0.05
MNIST	SVM	95.01	2108	95.03	0.03
	ELM	95.33	1114	95.32	0.31
	ML-ELM	96.78	3793	96.82	0.17
	AE-S-ELMs	96.93	3772	96.94	0.21
	DBN	96.67	14117	96.72	0.03
	ErrCor	96.17	4065	96.26	0.13
	PC-ELM	96.21	3049	96.24	0.09
	OCI-ELM	96.38	3092	96.39	0.09
	DOC-IELM-AEs	97.89	3985	97.94	0.06
OCR Letters	SVM	88.27	770.24	88.41	0.03
	ELM	89.02	265.01	89.03	0.22
	ML-ELM	89.19	923.23	89.17	0.14
	AE-S-ELMs	89.44	960.06	89.46	0.19
	DBN	88.98	15212	89.02	0.04
	ErrCor	88.81	1014	88.82	0.10
	PC-ELM	89.17	979.62	89.16	0.05
	OCI-ELM	89.36	992.08	89.39	0.08
	DOC-IELM-AEs	90.85	1175	90.86	0.06
NORB	SVM	91.13	723.81	91.14	0.04
	ELM	91.42	66.07	91.45	0.14
	ML-ELM	92.67	1107	92.70	0.17
	AE-S-ELMs	92.81	698.21	92.82	0.12
	DBN	92.58	44502	92.59	0.03
	ErrCor	91.77	1287	91.81	0.08
	PC-ELM	91.80	896.63	91.83	0.06
	OCI-ELM	92.61	839.48	92.65	0.05
	DOC-IELM-AEs	94.53	1480	94.56	0.03

FIGURE 4: The comparisons of prediction on 6-month strip-elongation data.

FIGURE 5: Different prediction errors of 5 models on 6-month data.

For further investigation on the prediction capabilities of DOC-IELM-AEs, the performances of algorithms are evaluated in terms of four criteria, that is, the mean absolute percentage error (MAPE), the mean square error (MSE), the relative root-mean-square error (rRMSE), and the absolute fraction of variance (R^2). During testing of DOC-IELM-AEs and other algorithms, they are also defined by using the following equations:

$$\begin{aligned}
 \text{MAPE} &= \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - y'_i}{y_i} \right| \times 100\%, \\
 \text{MSE} &= \frac{1}{n} \sum_{i=1}^n (y_i - y'_i)^2, \\
 \text{rRMSE} &= \sqrt{\frac{n \sum_{i=1}^n (y_i - y'_i)^2}{(n-1) \sum_{i=1}^n (y_i)^2}}, \\
 R^2 &= 1 - \left(\sqrt{\frac{\sum_{i=1}^n (y_i - y'_i)^2}{\sum_{i=1}^n (y'_i)^2}} \right),
 \end{aligned} \tag{33}$$

where y_i and y'_i are measured value and predicted value, respectively, and n is the number of testing data. The smaller MAPE, MSE, and rRMSE and the larger R^2 are indicative of better generalization performance of algorithm.

MAPE in Figure 5(a) evaluates the effect on measured values, generated by the disparity between the elongation values of steel-strips and the predicted values. Meanwhile, MSE and rRMSE in Figures 5(b) and 5(c), respectively, also reflect the dispersion of models, where MSE are more sensitive to the large errors compared with rRMSE because the squared errors amplify the large errors further. R^2 in Figure 5(d) is the distance between errors and predicted values. R^2 closer to 1 means that the algorithms have better performances. By analyzing the comparisons, it has become apparent that the results for evaluation criteria based on arbitrary 6-month testing data show better generalization performances of DOC-IELM-AEs than other algorithms in experiments of comparison. Accordingly, there is important practical significance in the prediction of elongation of steel-strips using DOC-IELM-AEs.

In order to demonstrate the effectiveness of the algorithm proposed in practical engineering, we have selected the successive data of 12 months from the whole data (16-month

FIGURE 6: The comparisons of prediction accuracy of annual statistical data.

data) to conduct the comparisons, and we obtained the prediction accuracies of every month and one year. The comparisons of prediction accuracy shown in Figure 6 indicate that the performance of the algorithm proposed is the best overall, with mean accuracy of 96.795% in 12 months (all the year), compared with those of 92.49%, 92.71%, 94.47%, 94.62%, 94.43%, 93.18%, 93.22%, and 94.50% obtained from SVM, ELM, ML-ELM, AE-S-ELMs, DBN, ErrCor, PC-ELM, and OCI-ELM methods, respectively. DOC-IELM-AEs has the best accuracy among the nine methods, which indicates the predictive stability and performance of the method. Therefore, we can get the conclusion that DOC-IELM-AEs has the best prediction performance in the testing, and the algorithm proposed is a very effective method.

7. Conclusions

In this paper, we proposed a stacked architecture with OCI-ELM algorithm based on deep representation learning and added the OCI-ELM autoencoder into each layer of OCI-ELM, called DOC-IELM-AEs. The experiment results have demonstrated strongly that DOC-IELM-AEs can be suitable for solving regression and classification problems; simulations showed that, (1) compared with CI-ELM, EI-ELM, ECI-ELM, PC-ELM, and OCI-ELM, DOC-IELM-AEs can achieve the best testing accuracy with the same network size, even less hidden nodes; meanwhile, the speed of learning is also faster than other algorithms. Moreover, DOC-IELM-AEs has better performance than OCI-ELM algorithm; (2) compared with SVM, ELM, ML-ELM, AE-S-ELMs, DBN, ErrCor, PC-ELM, and OCI-ELM, DOC-IELM-AEs can also obtain the best testing accuracy with consuming more time in a certain range for the large datasets; (3) compared with SVM, ELM, ML-ELM, AE-S-ELMs, DBN, ErrCor, PC-ELM, and OCI-ELM, the DOC-IELM-AEs applied in the case of strips-elongation prediction can enhance the performance of prediction; demonstrated with the production data, the prediction accuracy based on the algorithm we proposed

outperforms other algorithms. For these reasons, the OCI-ELM and DOC-IELM-AEs can further be implemented in practical engineering and have the potential for solving more complicated big data problems with further study.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61102124) and Liaoning Key Industry Programme (JH2/101).

References

- [1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006.
- [2] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107-122, 2011.
- [3] X.-Z. Wang, Q.-Y. Shao, Q. Miao, and J.-H. Zhai, "Architecture selection for networks trained with extreme learning machine using localized generalization error model," *Neurocomputing*, vol. 102, pp. 3-9, 2013.
- [4] A. M. Fu, C. R. Dong, and L. S. Wang, "An experimental study on stability and generalization of extreme learning machines," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 1, pp. 129-135, 2015.
- [5] X.-Z. Wang, R. A. R. Ashfaq, and A.-M. Fu, "Fuzziness based sample categorization for classifier performance improvement," *Journal of Intelligent and Fuzzy Systems*, vol. 29, no. 3, pp. 1185-1196, 2015.
- [6] J. Wu, S. T. Wang, and F.-L. Chung, "Positive and negative fuzzy rule system, extreme learning machine and image classification," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 4, pp. 261-271, 2011.
- [7] S. Lu, X. Wang, G. Zhang, and X. Zhou, "Effective algorithms of the Moore-Penrose inverse matrices for extreme learning machine," *Intelligent Data Analysis*, vol. 19, no. 4, pp. 743-760, 2015.
- [8] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.
- [9] J. Zhang, S. Ding, N. Zhang, and Z. Shi, "Incremental extreme learning machine based on deep feature embedded," *International Journal of Machine Learning and Cybernetics*, vol. 7, no. 1, pp. 111-120, 2016.
- [10] Y. Ye and Y. Qin, "QR factorization based Incremental Extreme Learning Machine with growth of hidden nodes," *Pattern Recognition Letters*, vol. 65, pp. 177-183, 2015.
- [11] J.-L. Ding, F. Wang, H. Sun, and L. Shang, "Improved incremental regularized extreme learning machine algorithm and its application in two-motor decoupling control," *Neurocomputing*, vol. 149, pp. 215-223, 2015.
- [12] Z. Xu, M. Yao, Z. Wu, and W. Dai, "Incremental regularized extreme learning machine and its enhancement," *Neurocomputing*, vol. 174, pp. 134-142, 2016.

- [13] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, no. 4–6, pp. 576–583, 2008.
- [14] Y. Li, "Orthogonal incremental extreme learning machine for regression and multiclass classification," *Neural Computing & Applications*, vol. 27, no. 1, pp. 111–120, 2016.
- [15] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16–18, pp. 3056–3062, 2007.
- [16] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [17] S. Ding, N. Zhang, X. Xu, L. Guo, and J. Zhang, "Deep extreme learning machine and its application in EEG classification," *Mathematical Problems in Engineering*, vol. 2015, Article ID 129021, 11 pages, 2015.
- [18] O. Vinyals, Y. Jia, L. Deng, and T. Darrell, "Learning with recursive perceptual representations," in *Advances in Neural Information Processing Systems*, pp. 2825–2833, 2012.
- [19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 2, no. 25, MIT Press, 2012.
- [20] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, pp. 151–161, Association for Computational Linguistics, July 2011.
- [21] Y. Bengio and O. Delalleau, "On the expressive power of deep architectures," in *Algorithmic Learning Theory*, J. Kivinen, C. Szepesvári, E. Ukkonen, and T. Zeugmann, Eds., vol. 6925 of *Lecture Notes in Computer Science*, pp. 18–36, Springer, New York, NY, USA, 2011.
- [22] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009.
- [23] Y. Bengio and Y. Lecun, "Scaling learning algorithms towards AI," *Large-Scale Kernel Machines*, vol. 2007, no. 34, pp. 1–41, 2007.
- [24] T. S. Shores, *Applied Linear Algebra and Matrix Analysis*, Springer, Berlin, Germany, 2007.
- [25] G. Taguchi and R. Jugulum, *The Mahalanobis Taguchi Strategy: A Pattern Technology System*, John Wiley & Sons, Hoboken, NJ, USA, 2002.
- [26] Y. M. Yang, Y. N. Wang, and X. F. Yuan, "Parallel chaos search based incremental extreme learning machine," *Neural Processing Letters*, vol. 37, no. 3, pp. 277–301, 2013.
- [27] Q. Yu, Y. Miche, E. Séverin, and A. Lendasse, "Bankruptcy prediction using Extreme Learning Machine and financial expertise," *Neurocomputing*, vol. 128, pp. 296–302, 2014.
- [28] K. I. Wong, M. V. Chi, P. K. Wong et al., "Sparse Bayesian extreme learning machine and its application to biofuel engine performance prediction," *Neurocomputing*, vol. 2015, no. 149, pp. 397–404, 2015.
- [29] L. L. C. Kasun, H. Zhou, G. B. Huang, and C. M. Vong, "Representational learning with extreme learning machine," *IEEE Intelligent Systems*, vol. 6, no. 28, pp. 31–34, 2013.
- [30] W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz maps into a Hilbert space," *Modern Analysis and Probability*, vol. 189, no. 26, pp. 189–206, 1984.
- [31] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [32] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," *Momentum*, vol. 1, no. 9, pp. 599–619, 2010.
- [33] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep Boltzmann machines," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS '10)*, vol. 9 of *JMLR: Workshop and Conference Proceedings*, pp. 693–700, 2010.
- [34] H. Zhou, G. B. Huang, Z. Lin et al., "Stacked extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 2, no. 2, pp. 1–13, 2014.
- [35] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18–28, 1998.
- [36] H. Yu, P. D. Reiner, T. Xie, T. Bartczak, and B. M. Wilamowski, "An incremental design of radial basis function networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 10, pp. 1793–1803, 2014.

Hindawi

Submit your manuscripts at
<http://www.hindawi.com>