Deep Extreme Learning Machine and Its Application in EEG Classification

Article *in* Mathematical Problems in Engineering · May 2015

DOI: 10.1155/2015/129021

5 authors, including:

Shifei Ding
China University of Mining Technology
193 PUBLICATIONS **1,213** CITATIONS
SEE PROFILE

Xinzheng Xu
China University of Mining Technology
36 PUBLICATIONS **345** CITATIONS
SEE PROFILE

Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 129021, 11 pages
http://dx.doi.org/10.1155/2015/129021

Hindawi

Research Article

Deep Extreme Learning Machine and Its Application in EEG Classification

Shifei Ding,[1,2] Nan Zhang,[1,2] Xinzheng Xu,[1,2] Lili Guo,[1,2] and Jian Zhang[1,2]

[1] *School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China*
[2] *Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*

Correspondence should be addressed to Shifei Ding; dingsf@cumt.edu.cn

Received 26 August 2014; Revised 4 November 2014; Accepted 12 November 2014

Academic Editor: Amaury Lendasse

Recently, deep learning has aroused wide interest in machine learning fields. Deep learning is a multilayer perceptron artificial neural network algorithm. Deep learning has the advantage of approximating the complicated function and alleviating the optimization difficulty associated with deep models. Multilayer extreme learning machine (MLELM) is a learning algorithm of an artificial neural network which takes advantages of deep learning and extreme learning machine. Not only does MLELM approximate the complicated function but it also does not need to iterate during the training process. We combining with MLELM and extreme learning machine with kernel (KELM) put forward deep extreme learning machine (DELM) and apply it to EEG classification in this paper. This paper focuses on the application of DELM in the classification of the visual feedback experiment, using MATLAB and the second brain-computer interface (BCI) competition datasets. By simulating and analyzing the results of the experiments, effectiveness of the application of DELM in EEG classification is confirmed.

1. Introduction

Brain-computer interface (BCI) is a kind of technology that enables people to communicate with a computer or to control devices with EEG signals [1]. The core technologies of BCI are to extract the feature of preprocessed EEG and classify ready-processed EEG, and this paper is mainly about classification analysis. In recent years, BCI has gotten a great advance with the rapid development of computer technology. BCI has been applied to many fields, such as medicine and military [2–4]. Currently, many different methods have been proposed for EEG classification, including decision trees, local backpropagation (BP) algorithm, Bayes classifier, K-nearest neighbors (KNN), support vector machine (SVM), batch incremental support vector machine (BISVM), and ELM [5–8]. However, most of them are shallow neural network algorithms in which the capabilities achieve approximating the complex functions that are subject to certain restrictions, and there is no such restriction in deep learning.

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 [9–11]. In 2006, the concept of deep learning was first proposed by Hinton and Salakhutdinov who presented deep structure of multilayer autoencoder [12]. Deep belief network was proposed by Hinton [13]. LeCun et al. put forward the first real deep learning algorithm—convolutional neural networks (CNNs) [14]. More and more people put forward some new algorithms based on deep learning. Then convolutional deep belief network was put forward [15]. In 2013, the model of multilayer extreme learning machine (MLELM) was proposed by Kasun et al. [16], and DELM takes advantages of deep learning and extreme learning machine. Extreme learning machine (ELM) proposed by Huang et al. is a simple and efficient learning algorithm of single layer feed-forward neural networks (SLFNs) [17, 18]. In addition, some people put forward some deformation algorithms based on ELM, such as regularized extreme learning machine (RELM) [19], extreme learning machine with kernel (KELM) [20], optimally pruned extreme learning machine (OP-ELM) [21], and evolving fuzzy optimally pruned extreme learning machine (eF-OP-ELM) [22].

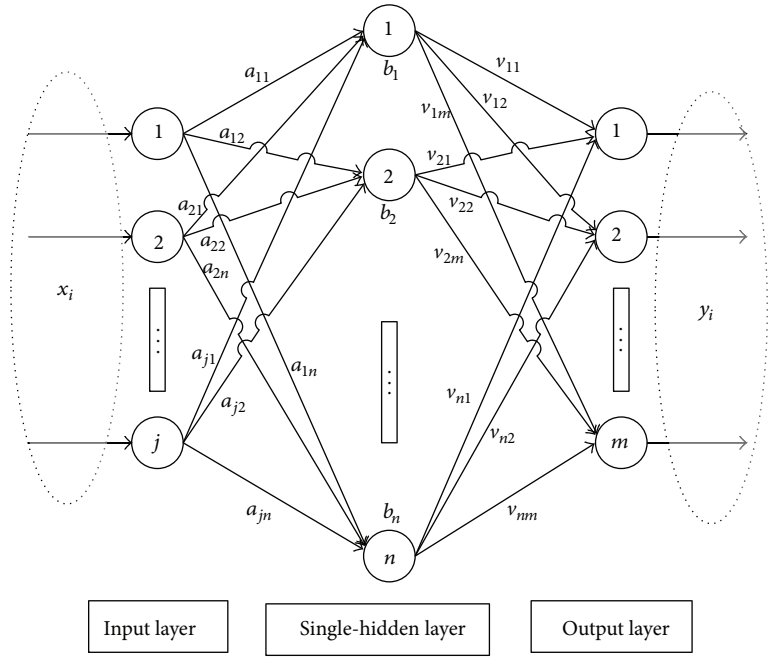


Figure 1: The model structure of ELM.

We combining with multilayer extreme learning machine (MLELM) and extreme learning machine with kernel (KELM) put forward deep extreme learning machine (DELM) and apply it to EEG classification, and the paper is organized as follows: Section 2 gives the model of ELM, RELM, and KELM. Section 3 describes the model structure of MLELM. Section 4 details the model structure of DELM. Section 5 first evaluates the usefulness of DELM on UCI datasets and then applies DELM to EEG classification. In Section 6, the conclusion is gotten.

2. Extreme Learning Machine (ELM)

2.1. Basic Extreme Learning Machine (Basic ELM). ELM proposed by Huang et al. is a simple and efficient learning algorithm of SLFNs. The model of ELM constituted input layer, single-hidden layer, and output layer. The model structure of ELM is shown in Figure 1, with j input layer nodes, n hidden layer nodes, m output layer nodes, and the hidden layer activation function $g(x)$.

For N distinct samples $x_i \in R_N \times R_j$, $y_i \in R_N \times R_m$ $(i = 1, 2, \ldots, N)$, the outputs of the hidden layer can be expressed as (1), and the numerical relationship between output of the hidden layer and output of the output layer can be expressed as (2):

$$h = g(ax + b), \tag{1}$$

$$h(x_i) V = y_i, \quad i = 1, 2, \ldots, N. \tag{2}$$

The above equation can be written compactly as

$$HV = Y, \tag{3}$$

where

$$H = \begin{bmatrix} g(\vec{a}_1, b_1, \vec{x}_1) & g(\vec{a}_1, b_1, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \\ g(\vec{a}_2, b_2, \vec{x}_1) & g(\vec{a}_2, b_2, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ g(\vec{a}_n, b_n, \vec{x}_1) & g(\vec{a}_n, b_n, \vec{x}_2) & \cdots & g(\vec{a}_n, b_n, \vec{x}_N) \end{bmatrix}^T, \tag{4}$$

$$V = \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix}_{n \times m}, \qquad Y = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_N^T \end{bmatrix}_{N \times m}, \tag{5}$$

where $a_i = [a_{i1}, a_{i2}, \ldots, a_{in}]^T$ are the weights connecting the ith input nodes and hidden layer, b_j is the bias of the jth hidden node, and $v_j = [v_{j1}, v_{j2}, \ldots, v_{jn}]^T$ are the weights connecting the jth hidden node and the output layer. H is output matrix of the neural network. We need to set input weights a_{ij} and the bias of the hidden layer b_j; the output weights V can be obtained by a series of linear equations transformations.

In conclusion, using ELM to obtain the output weights V can be divided into three steps.

Step 1. Randomly select numerical values between 0 and 1 to set input weights a_{ij} and the bias of the hidden layer b_j.

Step 2. Calculate the output matrix H.

Step 3. Calculate the output weights V:

$$V = H^{\dagger} Y, \tag{6}$$

where $H^{\dagger}$ represents the generalized inverse matrix of the output matrix H.

2.2. Regularized Extreme Learning Machine (RELM). ELM has the advantage of fast training speed and high generalization performance, but ELM also has the disadvantage of bad robustness. Deng et al. combining with experiential risk and structural risk put forward regularized extreme learning machine (RELM) which has better robustness, and RELM aims to solve the output weights by minimizing the regularized cost function of least squares estimate regularization, which leads to the following formulation:

$$\min L_{\mathrm{RELM}} = \frac{1}{2}\|V\|^2 + \frac{C}{2}\|Y - HV\|^2, \tag{7}$$

where C is a scale parameter which adjusts experiential risk and structural risk.

By setting the gradient of L_{RELM} with respect to $\mathbf{V}$ to zero, we have

$$V + CH^T(Y - HV) = 0. \tag{8}$$

When the number of training samples is more than the number of hidden layer nodes, the output weight matrix $\mathbf{V}$ in RELM can be expressed as

$$V = \left(\frac{I}{C} + H^T H\right)^{-1} H^T Y. \tag{9}$$

When the number of training samples is less than the number of hidden layer nodes, the output weight matrix $\mathbf{V}$ in RELM can be expressed as

$$V = H^T\left(\frac{I}{C} + HH^T\right)^{-1} Y. \tag{10}$$

2.3. Extreme Learning Machine with Kernel (KELM). Huang et al. combining with the kernel method and extreme learning machine put forward extreme learning machine with kernel (KELM). The outputs of the hidden layer of ELM can be regarded as the nonlinear mapping of samples. When the mapping is an unknown, we can construct the kernel function instead of HH^T:

$$HH^T = \Omega_{\mathrm{ELM}} = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_N) \\ \vdots & \ddots & \vdots \\ K(x_N, x_1) & \cdots & K(x_N, x_N) \end{bmatrix},$$
$$h(x)H^T = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T. \tag{11}$$

The most popular kernel of KELM in use is the Gaussian kernel $K(x_i, x_j) = \exp(-\|x_i - x_j\|/\gamma)$, where γ is the kernel parameter.

Thus, the output weight matrix $\mathbf{V}$ in KELM can be expressed as (12) and the Classification of formula of KELM can be expressed as (13):

$$V = \left(\frac{I}{C} + \Omega_{\mathrm{ELM}}\right)^{-1} Y, \tag{12}$$

$$f(x) = h(x)H^T V = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T \left(\frac{I}{C} + \Omega_{\mathrm{ELM}}\right)^{-1} Y. \tag{13}$$

3. Multilayer Extreme Learning Machine (MLELM)

3.1. Extreme Learning Machine-Autoencoder (ELM-AE). Autoencoder is an artificial neural network model which is commonly used in deep learning. Autoencoder is an unsupervised neural network, the outputs of autoencoder are the same as the inputs of autoencoder, and autoencoder is a kind of neural networks which reproduces the input signal as much as possible. ELM-AE proposed by Kasun et al. is a new method of neural network which can reproduce the input signal as well as autoencoder.

The model of ELM-AE constituted input layer, single-hidden layer, and output layer. The model structure of ELM-AE is shown in Figure 2, with j input layer nodes, n hidden layer nodes, j output layer nodes, and the hidden layer activation function $g(x)$. According to the output of the hidden layer representing the input signal, ELM-AE can be divided into three different representations as follows.

$j > n$: Compressed Representation: this represents features from a higher dimensional input signal space to a lower dimensional feature space.

$j = n$: Equal Dimension Representation: this represents features from an input signal space dimension equal to feature space dimension.

$j < n$: Sparse Representation: this represents features from a lower dimensional input signal space to a higher dimensional feature space.

There are two differences between ELM-AE and traditional ELM. Firstly, ELM is a supervised neural network and the output of ELM is label, but ELM-AE is an unsupervised neural network and the output of ELM-AE is the same as the input of ELM-AE. Secondly, the input weights of ELM-AE are orthogonal and the bias of hidden layer of ELM-AE is also orthogonal, but ELM is not so. For N distinct samples, $x_i \in R_n \times R_j$, $(i = 1, 2, \ldots, N)$, the outputs of ELM-AE hidden layer can be expressed as (14), and the numerical relationship between the outputs of the hidden layer and the outputs of the output layer can be expressed as (15):

$$h = g(ax + b), \quad \text{where } a^T a = I,\ b^T b = 1, \tag{14}$$

$$h(x_i)V = x_i^T, \quad i = 1, 2, \ldots, N. \tag{15}$$

Using ELM-AE to obtain the output weights V can be also divided into three steps, but the calculation method of the output weights V of ELM-AE in Step 3 is different from the calculation method of the output weights V of ELM.

For sparse and compressed ELM-AE representations, output weights $\mathbf{V}$ are calculated by (16) and (17).

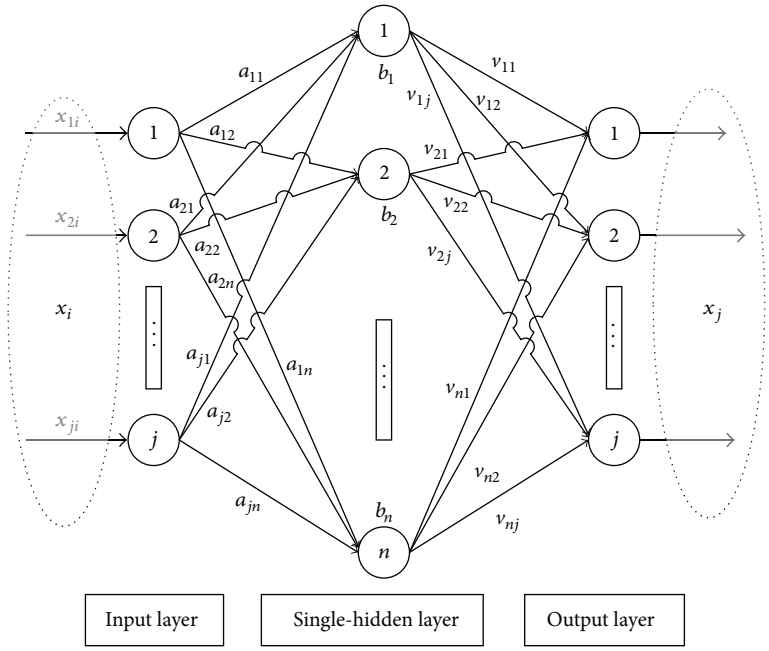


Figure 2: The model structure of ELM-AE.

When the number of training samples is more than the number of hidden layer nodes,

$$V = \left(\frac{I}{C} + H^T H\right)^{-1} H^T X. \tag{16}$$

When the number of training samples is less than the number of hidden layer nodes,

$$V = H^T \left(\frac{I}{C} + HH^T\right)^{-1} X. \tag{17}$$

For equal dimension ELM-AE representation, output weights V are calculated by

$$V = H^{-1} X. \tag{18}$$

3.2. Multilayer Extreme Learning Machine (MLELM). In 2006, Hinton et al. put forward an effective method of establishing a multilayer neural network on the unsupervised data. In the new method, first the parameters in each layer are obtained by unsupervised training, and then the network is fine-tuned by supervised learning. In 2013, MLELM was proposed by Kasun et al. Like other deep learning models, MLELM makes use of unsupervised learning to train the parameters in each layer, but the difference is that MLELM does not need to fine-tune the network. Thus, compared with other deep learning algorithms, MLELM does not need to spend a long time on the network training.

MLELM makes use of ELM-AE to train the parameters in each layer, and MLELM hidden layer activation functions can be either linear or nonlinear piecewise. If the activation function of the MLELM ith hidden layer is $g(x)$, then the parameters between the MLELM ith hidden layer and the MLELM $(i-1)$ hidden layer (if $i-1=0$, this layer is the input layer) are trained by ELM-AE, and the activation function should be $g(x)$, too. The numerical relationship between the outputs of MLELM ith hidden layer and the outputs of MLELM $(i-1)$ hidden layer can be expressed as

$$H_i = g\left(\left(v_i\right)^T H_{i-1}\right), \tag{19}$$

where H_i represents the outputs of MLELM ith hidden layer (if $i-1=0$, this layer is the input layer, and H_{i-1} represents the inputs of MLELM). The model of MLELM is shown in Figure 3, v_i represents the output weights of ELM-AE, the input of ELM-AE is H_{i-1}, and the number of ELM-AE hidden layer nodes is identical to the number of MLELM ith hidden nodes when the parameters between the MLELM ith hidden layer and the MLELM $(i-1)$ hidden layer are trained by ELM-AE. The output of the connections between the last hidden layer and the output layer can be analytically calculated using regularized least squares.

4. Deep Extreme Learning Machine (DELM)

MLELM makes use of ELM-AE to train the parameters in each layer, and ML-ELM hidden layer activation functions can be either linear or nonlinear piecewise, and the mapping of MLELM is linear or nonlinear. When the mapping is an unknown, we can add one hidden layer and construct the kernel function. In other words, at last the outputs of MLELM hidden layer H_k (the matrix size is $n_k * N$) are the inputs of KELM, and we can construct the kernel function instead

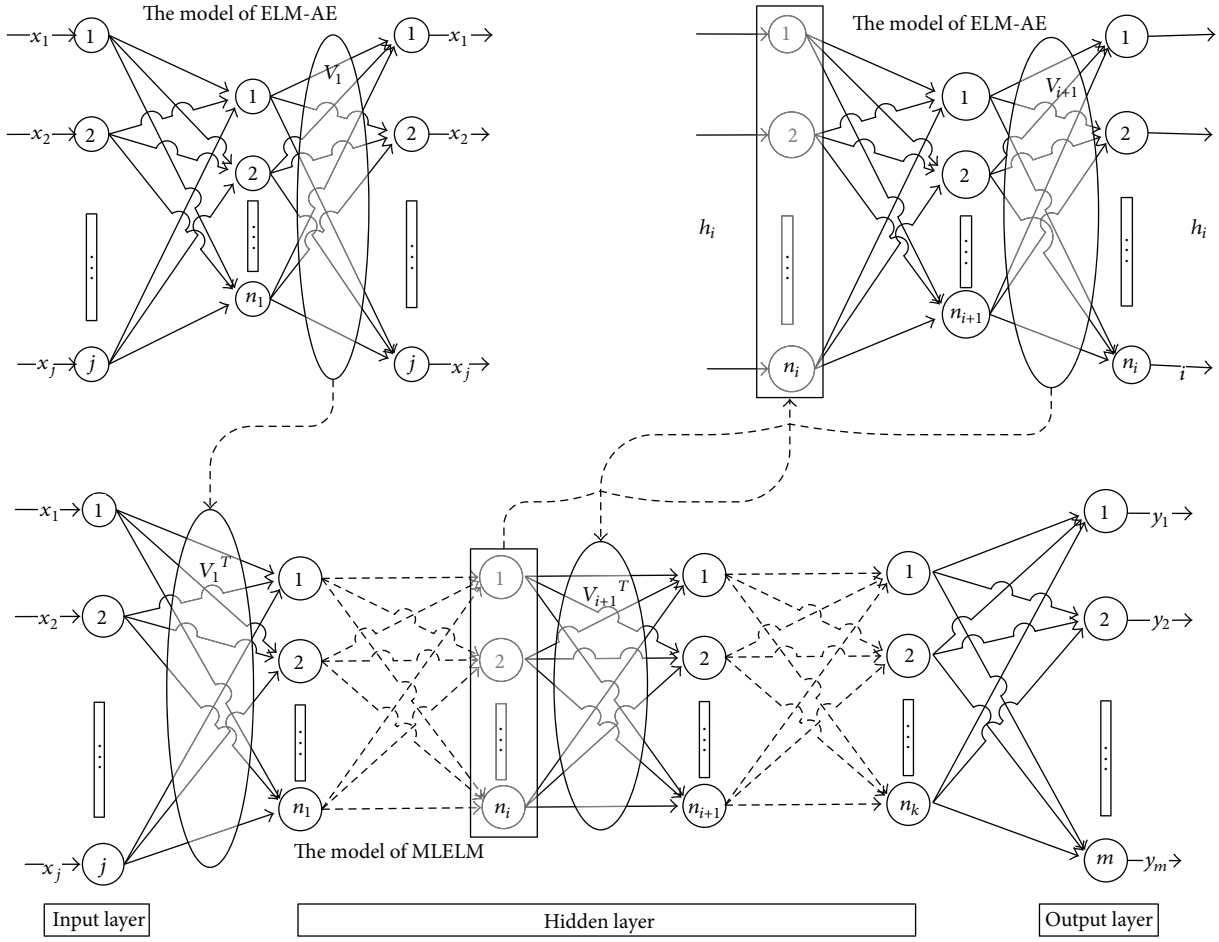


Figure 3: The model structure of MLELM.

of $H_{k+1}H_{k+1}^T$. This algorithm combining with MLELM and KELM is called deep extreme learning machine (DELM):

$$H_{k+1}H_{k+1}^T = \Omega_{\text{DELM}} = \begin{bmatrix} K(H_k(:,1), H_k(:,1)) & \cdots & K(H_k(:,1), H_k(:,N)) \\ \vdots & \ddots & \vdots \\ K(H_k(:,N), H_k(:,1)) & \cdots & K(H_k(:,N), H_k(:,N)) \end{bmatrix},$$

$$h_{k+1}(h_k(x))H_{k+1}^T = \begin{bmatrix} K(h_k(x), H_k(:,1)) \\ \vdots \\ K(h_k(x), H_k(:,N)) \end{bmatrix}^T. \tag{20}$$

The model of DELM is shown in Figure 4, v_i ($i \in [1, \ldots, k]$) represents the output weights of ELM-AE, the input of ELM-AE is H_{i-1}, and the number of ELM-AE hidden layer nodes is identical to the number of DELM ith hidden nodes when the parameters between the DELM ith hidden layer and the MLELM $(i-1)$ hidden layer are trained by ELM-AE. And we can construct the kernel function instead of $H_{k+1}H_{k+1}^T$; thus the output weight matrix $\mathbf{V}$ in DELM can be expressed as (21) and the classification of formula of KELM can be expressed as (22):

$$V = \left(\frac{I}{C} + \Omega_{\text{DELM}}\right)^{-1} Y, \tag{21}$$

$$f(x) = h_{k+1}(h_k(x))H_{k+1}^T V = \begin{bmatrix} K(h_k(x), H_k(:,1)) \\ \vdots \\ K(h_k(x), H_k(:,N)) \end{bmatrix}^T \left(\frac{I}{C} + \Omega_{\text{DELM}}\right)^{-1} Y. \tag{22}$$

5. Experiments and Analysis

The execution environment of experiments is MATLAB 2012B. All activation functions of ELM, MLELM, and DELM select sigmoid function and the kernel functions of KELM and DELM are Gaussian kernel. ELM, MLELM, and DELM were executed100 times, and the average values and the best values are reported.

5.1. UCI Datasets Classification. In this part, the UCI datasets were used to test the performances of DELM, and the details

Table 1: The details of UCI datasets.

Dataset	Number of samples		Attributes information		Number of labels
	Training samples	Testing samples	Number of attributes	Attribute characteristics	
Ionosphere	200	151	34	Continuous attributes	2
Diabetes	576	192	8	Categorical, integer	2

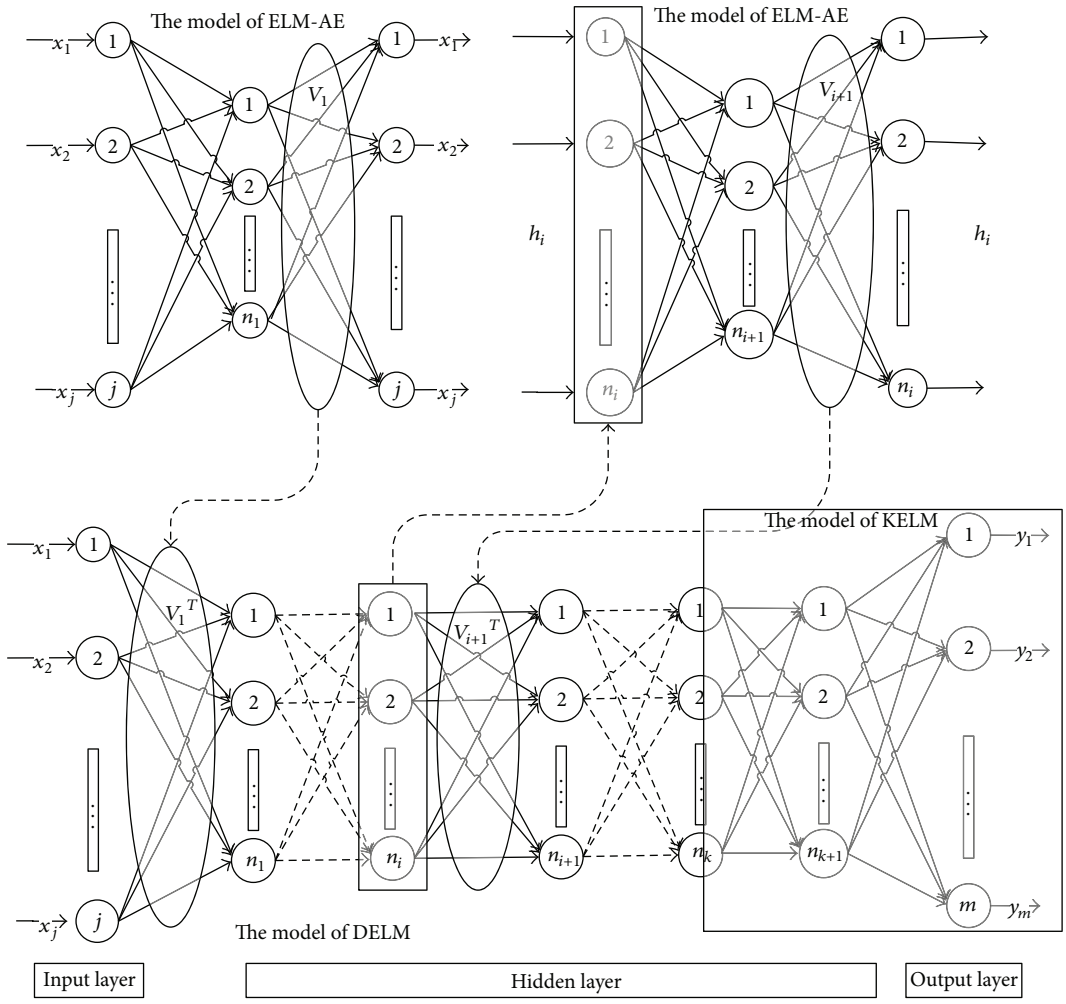


Figure 4: The model structure of DELM.

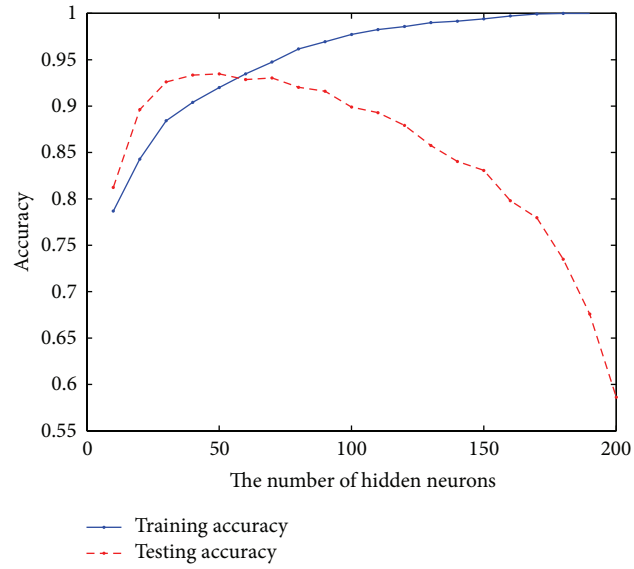
of UCI dataset are presented in Table 1, including ionosphere dataset and diabetes dataset.

As shown in Figure 5, we can make choices that the numbers of ELM hidden layer nodes on ionosphere dataset and diabetes dataset are 50 and 40, the regularized parameter C and the kernel parameter γ of KELM on ionosphere dataset are 10^3 and 10^2, and the regularized parameter C and the kernel parameter γ of KELM on diabetes dataset are 10^2 and 10^1. The structure of MLELM on ionosphere dataset is 34-30-30-50-2, where the parameter C for layer 34-30 is 10^3, the parameter C for layer 30-50 is 10^{-2}, and the parameter C for layer 50-2 is 10^8. And the structure of MLELM on diabetes dataset is 8-10-10-40-2, where the parameter C for layer 8-10 is 10^6, the parameter C for layer 10-40 is 10^8, and the parameter C for layer 40-2 is 10^5. The structure of DELM on ionosphere dataset is 34-30-30-L-2, where the parameter C for layer 34-30 is 10^1, the parameter C for layer L-2 is 10^3, and the kernel parameter γ is 10^2. And the structure of DELM on diabetes dataset is 8-10-10-L-2, where the parameter C for layer 34-30 is 10^1, the parameter C for layer L-2 is 10^2, and the kernel parameter γ is 10^1.

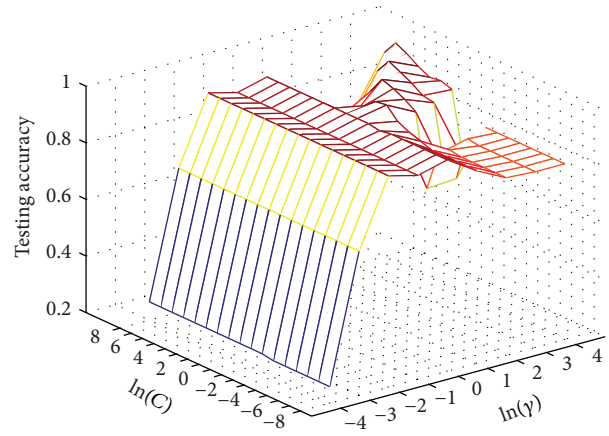
The performance comparison of DELM with ELM, KELM, and MLELM on UCI datasets is shown in Table 2. It is clearly observed that DELM testing accuracy is higher than MLELM, either the average or the maximum, and the best values of DELM testing accuracy are higher than ELM and

Table 2: Performance comparison of DELM with ELM, KELM, and MLELM on UCI datasets.

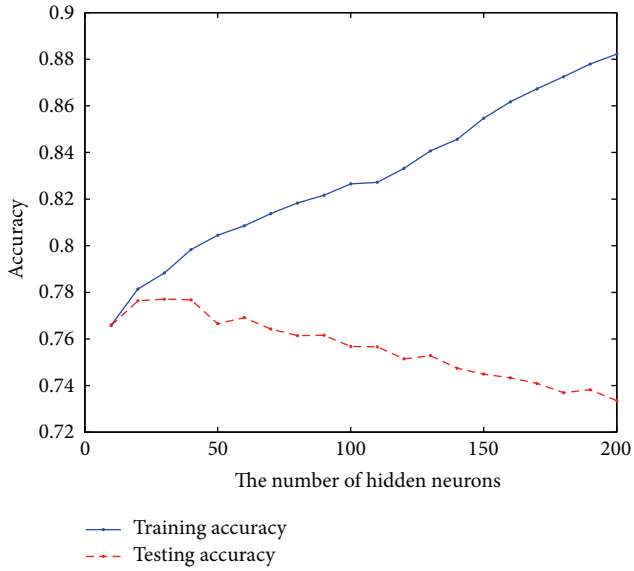
Dataset	Algorithm	#	Training accuracy	Testing accuracy	Training time (s)	Testing time (s)
Ionosphere	Basic ELM	Average	0.9207 ± 0.0151	0.9342 ± 0.0222	0.0044 ± 0.0074	0.0013 ± 0.0048
		Best	0.9500	0.9735	—	—
	KELM	—	0.9900	0.9735	0.0064	0.0017
	ML-ELM	Average	0.9112 ± 0.0159	0.9447 ± 0.0216	0.0115 ± 0.0116	0.0014 ± 0.0050
		Best	0.9500	0.9801	—	—
	DELM	Average	0.9503 ± 0.0111	0.9474 ± 0.0292	0.0164 ± 0.0079	0.0045 ± 0.0064
		Best	0.9750	0.9934	—	—
Diabetes	Basic ELM	Average	0.7983 ± 0.0062	0.7725 ± 0.0129	0.0072 ± 0.0132	0.0034 ± 0.0098
		Best	0.8125	0.8021	—	—
	KELM	—	0.7899	0.7917	0.6818	0.0314
	ML-ELM	Average	0.7666 ± 0.0207	0.7522 ± 0.0324	0.0091 ± 0.0143	0.0044 ± 0.0109
		Best	0.7951	0.8177	—	—
	DELM	Average	0.7871 ± 0.0079	0.7580 ± 0.0422	0.0641 ± 0.0143	0.0112 ± 0.0086
		Best	0.8038	0.8229		



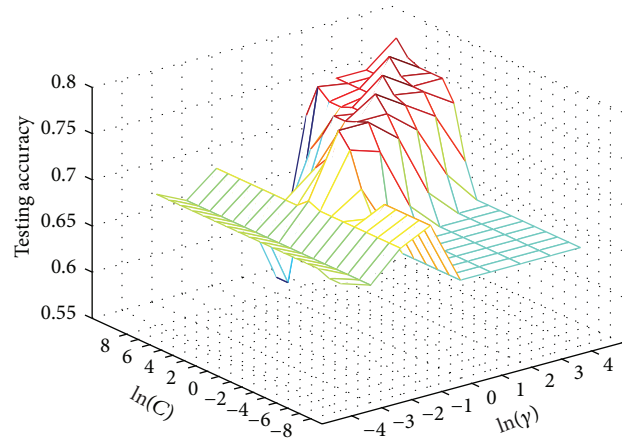
(a) Basic ELM for ionosphere dataset



(b) KELM for ionosphere dataset



(c) Basic ELM for diabetes dataset



(d) KELM for diabetes dataset

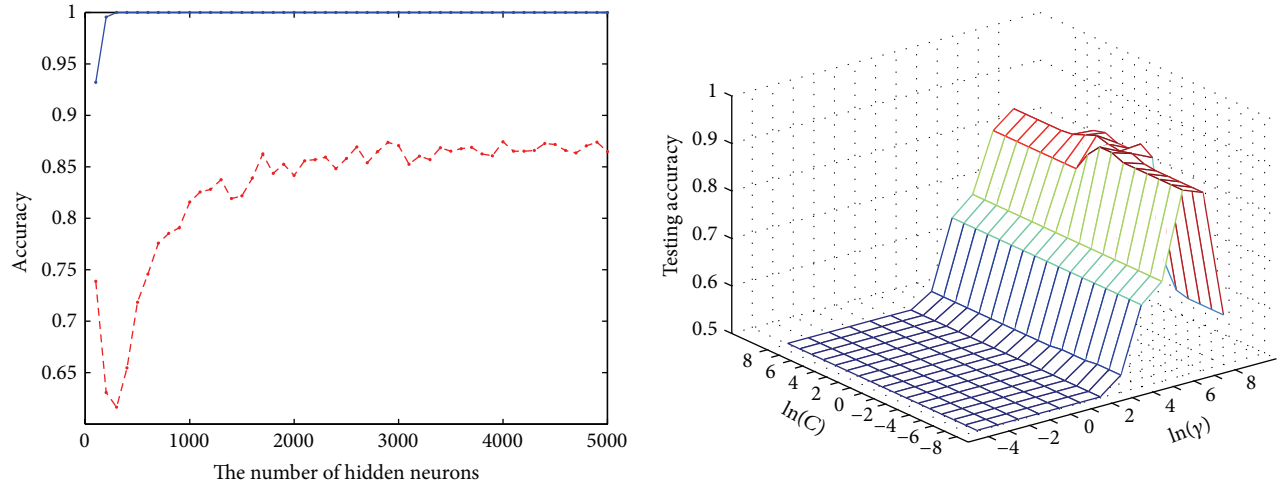
Figure 5: Basic ELM and KELM for UCI dataset.

Table 3: The details of the second BCI competition dataset IA.

Dataset	Number of samples		Number of attributes	Number of labels
	Training samples	Testing samples		
BCI competition II dataset IA	268	293	5376	2

Table 4: Performance comparison of DELM with ELM, KELM, and MLELM on the BCI competition II dataset IA.

Dataset	Algorithm	#	Training accuracy	Testing accuracy	Training time (s)	Testing time (s)
BCI competition II dataset IA	Basic ELM	Average	1.0000 ± 0	0.8609 ± 0.0187	3.3670 ± 0.0866	2.2361 ± 0.0498
		Best	1.0000	0.9078	—	—
	KELM	—	0.8582	0.9010	0.0754	0.1430
	ML-ELM	Average	0.7849 ± 0.0213	0.8642 ± 0.0216	9.2012 ± 0.1444	0.4820 ± 0.0272
		Best	0.8358	0.9113	—	—
	DELM	Average	0.7515 ± 0.0161	0.8650 ± 0.0224	6.7438 ± 0.2099	0.2932 ± 0.0290
		Best	0.7873	0.9181	—	—



(a) Basic ELM for the BCI competition II dataset IA

(b) KELM for the BCI competition II dataset IA

Figure 6: Basic ELM and KELM for the BCI competition II dataset IA.

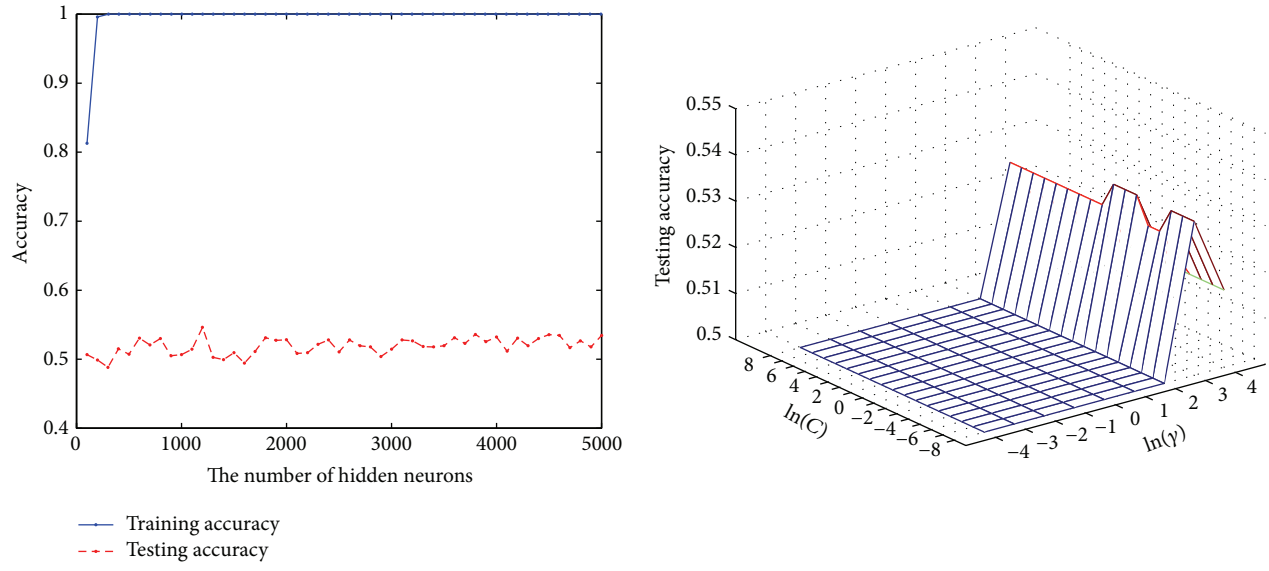
KELM. And DELM training time is the longest, but there is little difference between testing times. Sigillito et al. investigated ionosphere dataset using backpropagation and the perceptron training algorithm; they found that "linear" perceptron achieved 90.7%, a "nonlinear" perceptron achieved 92%, and backprop an average of over 96% accuracy [23]. Although the average value of DELM on ionosphere dataset only achieves 94.74%, the best value has reached to 99.34%.

5.2. EEG Classification. The effectiveness of DELM has been confirmed, so the effectiveness of the application of DELM in EEG classification is tested in this part.

5.2.1. Visual Feedback Experiment (Healthy Subject). The performances of DELM on the second BCI competition dataset IA are tested in this section, and this dataset comes from the visual feedback experiment (healthy subject) provided by University of Tuebingen [24].

The datasets were taken from a healthy subject. The subject was asked to move a cursor up and down on a computer screen, while his cortical potentials were taken. Cortical positivity leads to a downward movement of the cursor on the screen. Cortical negativity leads to an upward movement of the cursor. Each trial lasted 6 s. The visual feedback was presented from second 2 to second 5.5. Only this 3.5-second interval of every trial is provided for training and testing. The sampling rate of 256 Hz and the recording length of 3.5 s result in 896 samples per channel for every trial, and the details are presented in Table 3.

As shown in Figure 6, we can make choices that the number of ELM hidden layer nodes on BCI competition II dataset IA is 3000; the regularized parameter C and the kernel parameter γ of KELM are 10^3 and 10^4. The structure of



(a) Basic ELM for the BCI competition II dataset IA

(b) KELM for the BCI competition II dataset IA

Figure 7: Basic ELM and KELM for the BCI competition II dataset IA.

MLELM is 5376-500-500-3000-2, where the parameter C for layer 5376-500 is 2^1, the parameter C for layer 500-3000 is 2^8, and the parameter C for layer 3000-2 is 2^{-7}. The structure of DELM is 5376-500-500-L-2, where the parameter C for layer 5376-500 is 10^{-1}, the parameter C for layer L-2 is 10^{-1}, and the kernel parameter γ is 10^2.

The performance comparison of DELM with ELM, KELM, and MLELM on the BCI competition II dataset IA is shown in Table 4. It is clearly observed that DELM testing accuracy is higher than MLELM, either the average or the maximum, and the best values of DELM testing accuracy are higher than ELM and KELM. MLELM training time is the longest, and the testing time of MLELM and DELM is less than ELM. The performance comparison of DELM with the results of BCI competition II dataset IA is shown in Table 5. It is clear that the average error value of DELM on BCI competition II dataset IA achieves 13.50%, but the min error value has reduced to 8.19%, which is much lower than the results of BCI competition II.

Table 5: Performance comparison of DELM with the results of BCI competition II on dataset IA.

#	Contributor	Error (%)
1	Brett Mensh	11.3
2	Guido Dornhege	11.6
3	Kai-Min Chung	11.9
4	Tzu-Kuo Huang	15.0
5	David Pinto	15.7
6	Juma Mbwana	17.1
7	Vladimir Bostanov	17.4
8	Ulrich Hoffmann	17.8
9	Deniz Erdogmus	19.1
10	Justin Sanchez	19.8
*	Ours (the average value of DELM)	13.50
*	Ours (the best value of DELM)	8.19

5.2.2. Visual Feedback Experiment (ALS Patient). The performances of DELM on the second BCI competition dataset IB are tested in this section, and this dataset comes from the visual feedback experiment (ALS patient) provided by University of Tuebingen.

The datasets were taken from an artificially respirated ALS patient. The subject was asked to move a cursor up and down on a computer screen, while his cortical potentials were taken. Cortical positivity leads to a downward movement of the cursor on the screen. Cortical negativity leads to an upward movement of the cursor. Each trial lasted 8 s. The visual feedback was presented from second 2 to second 6.5. Only this 4.5-second interval of every trial is provided for training and testing. The sampling rate of 256 Hz and the recording length of 4.5 s result in 1152 samples per channel for every trial, and the details are presented in Table 6.

As shown in Figure 7, we can make choices that the number of ELM hidden layer nodes on BCI competition II dataset IA is 2000; the regularized parameter C and the kernel parameter γ of KELM are 10^{-1} and 10^3. The structure of MLELM is 8064-500-500-2000-2, where the parameter C for layer 8064-500 is 10^1, the parameter C for layer 500-2000 is 10^8, and the parameter C for layer 2000-2 is 10^4. The structure of DELM is 8064-500-500-L-2, where the parameter C for layer 8064-500 is 10^{-2}, the parameter C for layer L-2 is 10^{-8}, and the kernel parameter γ is 10^1.

The performance comparison of DELM with ELM, KELM, and MLELM on the BCI competition II dataset IB is shown in Table 7. It is clearly observed that the best of DELM testing accuracy is not lower than MLELM, ELM, and KELM.

Table 6: The details of the second BCI competition dataset IB.

Dataset	Number of samples		Number of attributes	Number of labels
	Training samples	Testing samples		
BCI competition II dataset IB	200	180	8064	2

Table 7: Performance comparison of DELM with ELM, KELM, and MLELM on the BCI competition II dataset IB.

Dataset	Algorithm	#	Training accuracy	Testing accuracy	Training time (s)	Testing time (s)
BCI competition II dataset IB	Basic ELM	Average	1.0000 ± 0	0.5172 ± 0.0395	3.4636 ± 0.1255	2.0602 ± 0.0670
		Best	1.0000	0.6056	—	—
	KELM	—	1.0000	0.5333	0.0738	0.1285
	ML-ELM	Average	0.6145 ± 0.0306	0.5219 ± 0.0284	10.4225 ± 0.1134	0.3970 ± 0.0182
		Best	0.6750	0.5833	—	—
	DELM	Average	0.7151 ± 0.0485	0.5211 ± 0.0266	8.9814 ± 0.2085	0.2603 ± 0.0231
		Best	0.8450	0.6056	—	—

Table 8: Performance comparison of DELM with the results of BCI competition II on dataset IB.

#	Contributor	error
1	Vladimir Bostanov	45.6%
2	Tzu-Kuo Huang	46.7%
2	Juma Mbwana	46.7%
4	Kai-Min Chung	47.8%
5	Xichen Sun	48.3%
6	Amir Saffari	53.3%
7	Fabien Torre	54.4%
8	Brett Mensh	56.1%
*	Ours (the average value of DELM)	47.89%
*	Ours (the best value of DELM)	39.44%

MLELM training time is the longest, and the testing time of MLELM and DELM is less than ELM. The performance comparison of DELM with the results of BCI competition II dataset IA is shown in Table 8. It is clear that the average error value of DELM on BCI competition II dataset IA achieves 47.89%, but the min error value has reduced to 39.44%, which is much lower than the results of BCI competition II.

6. Conclusions

This paper explores the application of DELM in EEG classification and makes use of two BCI competition datasets to test the performances of DELM. Experimental results show that DELM has the advantage of the least training time and the good efficiency and DELM is an effective BCI classifier. Although DELM has these advantages, there are some places which should be improved, such as the number of all hidden layer nodes, each hidden layer activation function, and each layer parameter C that are difficult to determine. In this paper, DELM is used to classify preprocessed EEG data and the feature attributes of preprocessed EEG are not extracted, which has certain effects on the experimental results. Future research is to combine the EEG feature extraction methods and DELM, which will be applied to the EEG classification.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61379101), the National Key Basic Research Program of China (No. 2013CB329502), the Natural Science Foundation of Jiangsu Province (No. BK20130209), and the Fundamental Research Funds for the Central Universities (No. 2013XK10).

References

[1] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, 2003.

[2] J.-B. Zhao and Z.-J. Zhang, "Progress in brain-computer interface based on cortical evoked potential," *Space Medicine & Medical Engineering*, vol. 23, no. 1, pp. 74–78, 2010.

[3] M. Middendorf, G. McMillan, G. Calhoun, and K. S. Jones, "Brain-computer interfaces based on the steady-state visual-evoked response," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 2, pp. 211–214, 2000.

[4] Z.-Y. Feng, *EEG Applied Research in Personal Identification and Fatigue Detection*, Beijing University of Posts and Telecommunications, 2013.

[5] N. Ye, Y.-G. Sun, and X. Wang, "Classification of brain-computer interface signals based on common spatial patterns and K-nearest neighbors," *Journal of Northeastern University*, vol. 30, no. 8, pp. 1107–1110, 2009.

[6] M. Meng and Z.-Z. Luo, "Hand motion classification based on eye-moving assisted EEG," *Pattern Recognition and Artificial Intelligence*, vol. 25, no. 6, pp. 1007–1012, 2012.

[7] B.-H. Yang, M.-Y. He, L. Liu, and W.-Y. Lu, "EEG classification based on batch incremental SVM in brain computer interfaces," *Journal of Zhejiang University (Engineering Science)*, vol. 47, no. 8, pp. 1431–1436, 2013.

[8] Q. Yuan, W. Zhou, S. Li, and D. Cai, "Approach of EEG detection based on ELM and approximate entropy," *Chinese Journal of Scientific Instrument*, vol. 33, no. 3, pp. 514–519, 2012.

[9] Y. Bengio and O. Delalleau, "On the expressive power of deep architectures," in *Algorithmic Learning Theory*, vol. 6925 of *Lecture Notes in Computer Science*, pp. 18–36, Springer, Berlin, Germany, 2011.

[10] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009.

[11] Y. Bengio and Y. Lecun, "Scaling learning algorithms towards AI," in *Large-Scale Kernel Machines*, vol. 34, pp. 1–41, 2007.

[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *American Association for the Advancement of Science: Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, article 5947, 2009.

[14] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[15] M. Norouzi, M. Ranjbar, and G. Mori, "Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pp. 2735–2742, IEEE Press, Miami, Fla, USA, 2009.

[16] L. L. C. Kasun, H.-M. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intelligent System*, vol. 28, no. 6, pp. 31–34, 2013.

[17] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66–77, 2012.

[18] E. Cambria, G.-B. Huang, L. L. C. Kasun et al., "Extreme learning machines," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 30–59, 2013.

[19] W.-Y. Deng, Q.-H. Zheng, L. Chen, and X.-B. Xu, "Research on extreme learning of neural networks," *Chinese Journal of Computers*, vol. 33, no. 2, pp. 279–287, 2010.

[20] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[21] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: optimally pruned extreme learning ," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158–162, 2010.

[22] F. M. Pouzols and A. Lendasse, "Evolving fuzzy optimally pruned extreme learning machine for regression problems," *Evolving Systems*, vol. 1, no. 1, pp. 43–58, 2010.

[23] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, vol. 10, no. 3, pp. 262–266, 1989.

[24] N. Birbaumer, N. Ghanayim, T. Hinterberger et al., "A spelling device for the paralysed," *Nature*, vol. 398, no. 6725, pp. 297–298, 1999.