Cognitive Computation (2020) 12:979–990
https://doi.org/10.1007/s12559-020-09752-2

Densely Connected Deep Extreme Learning Machine Algorithm

X. W. Jiang[1] · T. H. Yan[1] · J. J. Zhu[1] · B. He[2] · W. H. Li[3] · H. P. Du[3] · S. S. Sun[3]

Received: 23 May 2018 / Accepted: 3 July 2020 / Published online: 8 August 2020

Abstract
As a single hidden layer feed-forward neural network, the extreme learning machine (ELM) has been extensively studied for its short training time and good generalization ability. Recently, with the deep learning algorithm becoming a research hotspot, some deep extreme learning machine algorithms such as multi-layer extreme learning machine (ML-ELM) and hierarchical extreme learning machine (H-ELM) have also been proposed. However, the deep ELM algorithm also has many shortcomings: (1) when the number of model layers is shallow, the random feature mapping makes the sample features cannot be fully learned and utilized; (2) when the number of model layers is deep, the validity of the sample features will decrease after continuous abstraction and generalization. In order to solve the above problems, this paper proposes a densely connected deep ELM algorithm: dense-HELM (D-HELM). Benchmark data sets of different sizes have been employed for the property of the D-HELM algorithm. Compared with the H-ELM algorithm on the benchmark dataset, the average test accuracy is increased by 5.34% and the average training time is decreased by 21.15%. On the NORB dataset, the proposed D-HELM algorithm still maintains the best classification results and the fastest training speed. The D-HELM algorithm can make full use of the features of hidden layer learning by using the densely connected network structure and effectively reduce the number of parameters. Compared with the H-ELM algorithm, the D-HELM algorithm significantly improves the recognition accuracy and accelerates the training speed of the algorithm.

Keywords Extreme learning machine (ELM) · Densely connections · Deep learning · Representation learning

Introduction

Since the concept of extreme learning machine (ELM) algorithm had been proposed in 2006 [1], ELM algorithm has rapidly become an attractive area of research spot in the field of machine learning and artificial intelligence research, after continuous research and development by scholars. In recent years, extreme learning machine algorithm based on kernel function (K-ELM) [2], multi-layer extreme learning machine algorithm (ML-ELM) [3], hierarchical extreme learning machine algorithm (H-ELM) [4], local receptive field-based ELM (ELM-LRF) [5], and kernel multi-layer extreme learning machine algorithm (ML-KELM) [6] have been successively proposed. They are widely used in various fields such as image classification recognition [7, 8], big data analysis [9, 10], target detection and tracking [11, 12], and artificial intelligence [13, 14].

The ELM algorithm is a feed-forward neural network with a single hidden layer. The most basic structure includes three layers: input layer, hidden layer, and output layer, as shown in Fig. 1. The network structure is similar to the single hidden layer BP neural network [15], but the training methods are completely different. The BP neural network algorithm obtains the parameters of the learning model by (1) assigning initial values to the model parameters, (2) then adjusting the model parameters by non-iterative inverse iteration, (3) until the loss function has the smallest value. However, the ELM algorithm obtains the parameters of the learning model by (1) randomly generating the parameters from the input layer to the hidden layer and (2) then using the least square method to solve the parameters from the hidden layer to the output layer. This avoids repeated iterative calculations when training the model. Compared with BP neural network, it not only improves the recognition result, but also greatly saves training time.

✉ T. H. Yan
thyan@cjlu.edu.cn; thyan@163.com

[1] School of Mechanical & Electrical Engineering, China Jiliang University, Hangzhou, China

[2] School of Information Science & Engineering, Ocean University of China, Qingdao, Shandong, China

[3] Faculty of Engineering & Information System, University of Wollongong, Wollongong, NSW 2522, Australia

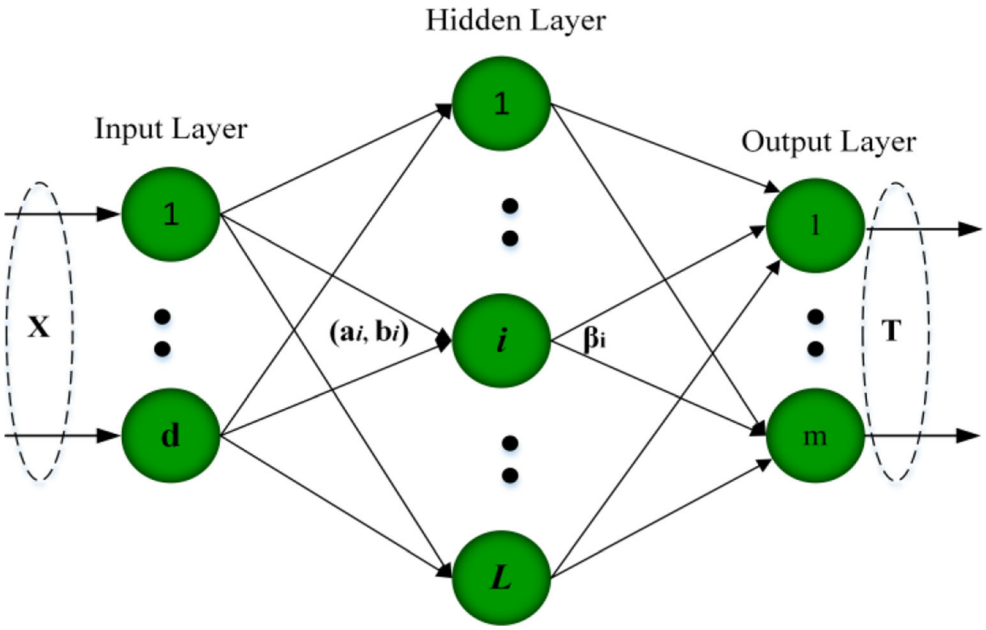

Fig. 1 Network structure diagram of the ELM algorithm

Complex classification or regression problems are often linearly inseparable in the low-dimensional feature space, but we map them to high-dimensional feature spaces through nonlinear mapping, and it is possible to achieve linear separability in high-dimensional feature spaces. However, if this mapping technique is used directly for classification or regression in high-dimensional space, there are problems such as the form, parameters of the nonlinear mapping function, and feature space dimension. And the biggest obstacle is the "curse of dimensionality" that will occur in the calculation of high-dimensional feature space. Using kernel function technology [16, 17] can effectively solve these problems. Support vector machine (SVM) [18, 19] is a typical representative in this field. With shorter training time and higher recognition accuracy, many complex classification and regression problems that were difficult to solve in the past have been solved. To compare with SVM, the proposed K-ELM cannot only effectively shorten training time but also has more excellent performance than SVM in the recognition accuracy; above aforementioned properties of K-ELM gradually replace SVM in various application fields.

But, both the conventional ELM and the K-ELM have a relatively shallow structure, and even if there are a large number of hidden nodes, it is difficult to obtain a good learning effect for images and videos that contain a large amount of learning characteristics. So, paper [4] proposed an H-ELM algorithm based on the principle of multi-layer perceptron. The H-ELM algorithm uses a sparsely self-encoded unsupervised learning method to train the model. After training the model, supervised feature classification is performed. This deep learning architecture, with the deepening of the model structure, continuously extracts effective features in images or videos, making the classification highly accurate, even exceeding many classical deep learning models, such as the Stacked Auto Encoders (SAE) [20], the Deep Belief Networks (DBN) [21], and the Deep Boltzmann Machines (DBM) [22]. However, as the network structure becomes increasingly deep, a new research problem emerges: as information about the input or gradient passes through many layers, it can vanish and "wash out" by the time it reaches the end (or beginning) of the network. In order to improve the training efficiency of parameters, some scholars have made intensive connection of the deep convolutional neural network (DCNN) algorithm [23], which makes the deep model easier to train and obtain good results. This paper applies this idea to ELM algorithm and proposes a learning model (D-HELM) for densely connected learning machines. D-HELM algorithm's training process is similar to that of H-ELM algorithm. It is also structurally divided into two separate phases: (1) unsupervised hierarchical feature representation and (2) supervised feature classification. The D-HELM algorithm consists of multiple ELM auto-encoders (ELM-AE) after dense connection. The weight parameters and bias parameters between each input layer and hidden layer are still randomly generated, except that the output parameters of the last layer are solved by least squares analysis. The output layer parameters of other layers are all determined by unsupervised sparse self-encoding method. The output of each feature presentation layer of the D-HELM algorithm not only serves as an input to the next layer, but also links with the output of all subsequent feature presentation layers; the structure formation is called densely connected network. The network structure of D-HELM algorithm makes full use of the feature results extracted by each feature extraction layer, so the recognition results are greatly improved compared to H-ELM algorithm. Especially for those datasets whose characteristic dimension itself is very small, it

can still obtain excellent recognition results when the network structure is not extremely deep.

The following organizational structure of this article is as follows: "Related Work" briefly introduces the related works of the article, including the basic structure and training method of H-ELM algorithm. "Proposed Learning Algorithm" introduces the proposed D-HELM algorithm network structure and training process. "Experiment and Evaluation" compares the proposed D-HELM algorithm and H-ELM algorithm recognition results on some common benchmark data sets and analyzes experimental parameters by plotting. And on the NORB dataset, the recognition results of the proposed algorithm are compared with other deep learning algorithms. Finally, the conclusion is given in "Conclusion".

Related Works

In order to better understand the D-HELM algorithm proposed in this paper, we will briefly introduce the related knowledge of ELM algorithm, the network structure, and training process of H-ELM algorithm in this section.

ELM Algorithm

According to the relevant theories of ELM in the paper [1], given a training dataset $\boldsymbol{X}=[x_1, x_2, \cdots, x_N]\in R^{d\times N}\in R^{d\times N}$ of $\boldsymbol{N}$ samples with label $\mathbf{T}=[t_1, t_2, \cdots, t_N]\in R^{d\times N}$, where d is the dimension of sample and m is the number of classes. The output function of an ELM with L hidden nodes can be expressed as [24]

$$f_{\mathrm{ELM}}(x_j)=\sum_{i=1}^{L}\beta_i h(a_i x_j+b_i)=y_j, j=1,2,\cdots,N \quad (1)$$

where $h(\cdot)$ is the nonlinear activation function, $\beta_i\in R^m$ denotes the weight vector connecting the hidden nodes and the output nodes, $a_i\in R^d$ is the weight vector connecting the hidden nodes and the input nodes, and b_i is the biases of the hidden nodes, in which a_i and b_i are randomly generated.

According to the minimization loss function, in order to reach the smallest training error and the smallest norm of output weights [25]

$$\text{Minimize}: \|\boldsymbol{\beta}\|_{\mu}^{\sigma_1}+C\|\mathbf{H}\boldsymbol{\beta}-\mathbf{T}\|_{\vartheta}^{\sigma_2} \quad (2)$$

where $\sigma_1>0$, $\sigma_2>0$, $\mu,\vartheta\vartheta=0, 0.5, 1, 2, \cdots, +\infty$. $\mathbf{H}$ is the hidden layer output matrix (randomized matrix). The resultant solution is equivalent to the ELM optimization solution with $\sigma_1=\sigma_2=\mu=\vartheta=2$, which is more stable and has better generalization performance.

when $\mathbf{H}\boldsymbol{\beta}$-$\mathbf{T}=0$, the equation (2) holds, and the value of $||\beta||_2^2$ is the smallest. Therefore, equation (1) can be written in the form of a matrix:

$$\mathbf{H}\boldsymbol{\beta}=\mathbf{T} \quad (3)$$

where

$$\mathbf{H}=\begin{bmatrix} g(\mathbf{a}_1\cdot\mathbf{x}_1+b_1) & \cdots & \mathbf{a}_L\cdot\mathbf{x}_1+b_L \\ \vdots & \ddots & \vdots \\ \mathbf{a}_1\cdot\mathbf{x}_N+b_1 & \cdots & \mathbf{a}_L\cdot\mathbf{x}_N+b_L \end{bmatrix}_{N\times L} \quad (4)$$

$$\boldsymbol{\beta}=\begin{bmatrix}\beta_1^T\\ \vdots\\ \beta_L^T\end{bmatrix},\quad \mathbf{T}=\begin{bmatrix}t_1^T\\ \vdots\\ t_N^T\end{bmatrix} \quad (5)$$

We can derive the least-square solution with minimum norm by

$$\boldsymbol{\beta}=\mathbf{H}^{\dagger}\mathbf{T} \quad (6)$$

$\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $\mathbf{H}$.

We can use the orthogonal projection method to compute MP inverse

$$\mathbf{H}^{\dagger}=\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T \quad (7)$$

To have better generalization performance and to make the solution more robust, we can add a regularization term as shown in [3].

$$\boldsymbol{\beta}=\left(\frac{\mathbf{I}}{\mathrm{C}}+\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \quad (8)$$

The corresponding output function of ELM is:

$$f_{ELM}(\mathbf{x})=h(\mathbf{x})\boldsymbol{\beta}=h(\mathbf{x})\left(\frac{\mathbf{I}}{\mathrm{C}}+\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}. \quad (9)$$

H-ELM Algorithm

The H-ELM algorithm framework is similar to the multi-layer perceptron. It replaces the perceptron structure in the multi-layer perceptron with the ELM. Each ELM input parameter and bias are randomly generated. The output parameters are determined by the ELM sparse autoencoder (ELM-AE). The network structure of ELM-AE is shown in Fig. 2.

In order to generate more sparse and efficient input features, H-ELM adds sparse constraints to ELM-AE to form an ELM sparse autoencoder. Its mathematical model can be expressed as the following equation:

$$\mathbf{o}_{\beta}=\underset{\beta}{\boldsymbol{argmin}}\left\{\|H\beta-X\|^2+\mathrm{C}\|\beta\|_{\ell_1}\right\} \quad (10)$$

The output weight of the hidden layer of the ELM-AE algorithm is solved using a fast iterative shrinkage-thresholding algorithm (FISTA) [26]. The FISTA algorithm converges fast and guarantees a global optimal solution. It

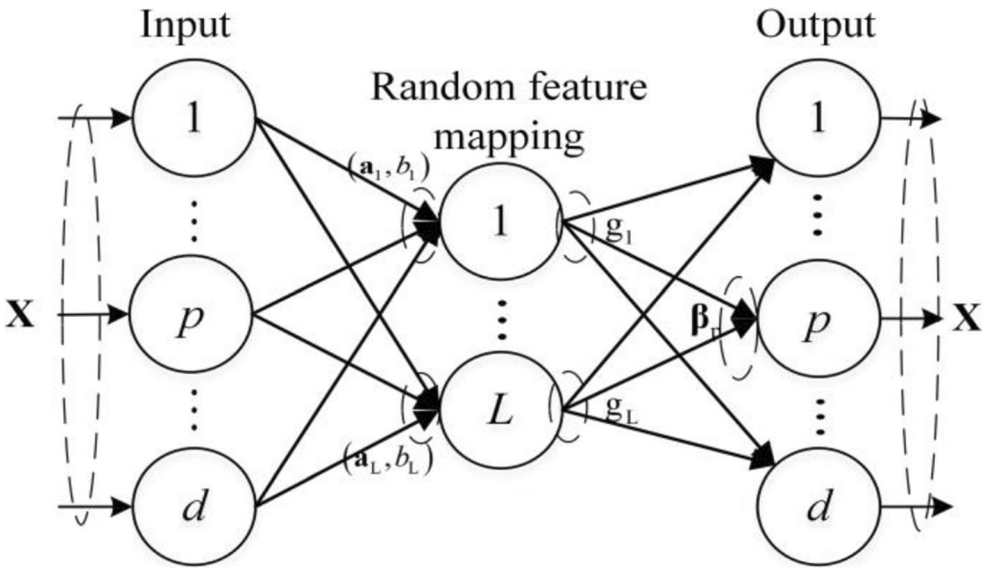

Fig. 2 The network structure of ELM-AE

minimizes a smooth convex function with complexity $O(1/j^2)$ and j is the number of iterations. The detailed iterative process of the FISTA algorithm is as follows:

(1) Calculate the Lipschitz constant γ of the gradient of smooth convex function ∇p.

(2) Begin the iteration by taking $y_1 = \beta_0 \in R^n$, $t_1 = 1$ as the initial points. Then, for j ($j \geq 1$), the following holds.

(a) $\beta_j = s_\gamma\left(y_j\right)$, where $s_\gamma$is given by

$$s_\gamma = \underset{\beta}{argmin}\left\{\frac{\gamma}{2}\left\|\beta-\left(\beta_{j-1}-\frac{1}{\gamma}\nabla p\left(\beta_{j-1}\right)\right)\right\|^2 + q(\beta)\right\} \quad (11)$$

$$t_{j+1} = \frac{1+\sqrt{1+4t_j^2}}{2} \quad (12)$$

$$y_{j+1} = \beta_j + \left(\frac{t_j-1}{t_{j+1}}\right)\left(\beta_j-\beta_{j-1}\right) \quad (13)$$

By computing the iterative steps above, we can manage to perfectly recover the data from the corrupted ones. Using the resultant bases β as the weights of the proposed autoencoder, the inner product of the inputs and learned features would reflect the compact representations of the original data.

The H-ELM algorithm is constructed in a multilayer manner, as shown in Fig. 3. Unlike the greedy layerwise training of the traditional deep learning frameworks, one can see that the H-ELM training architecture is structurally divided into two separate phases: (1) unsupervised hierarchical feature representation and (2) supervised feature classification. For the former phase, ELM-based autoencoder is used to extract multilayer sparse features of the input data, while for the latter one,

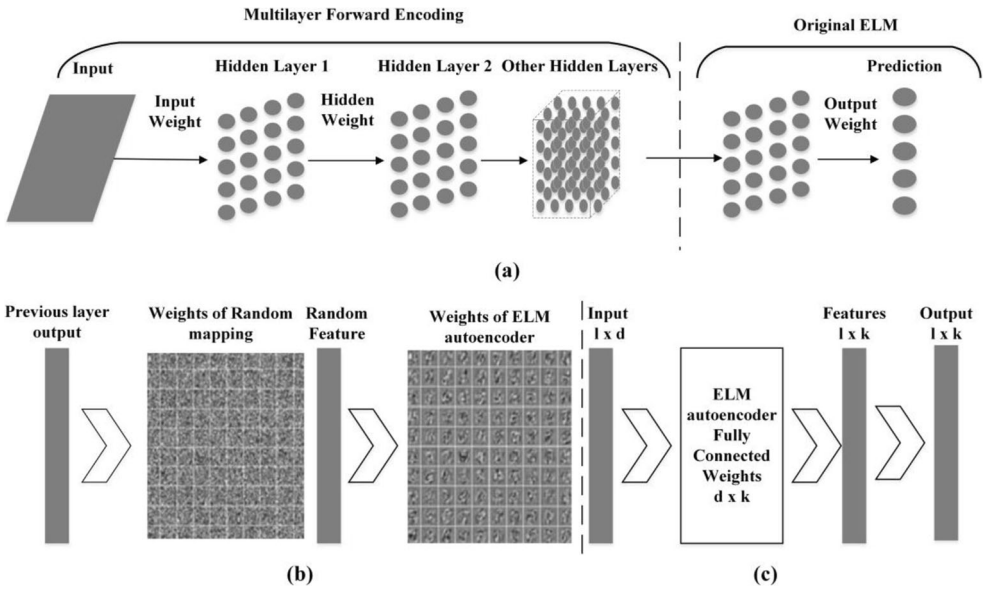

Fig. 3 H-ELM algorithm framework. **a** Overall framework of H-ELM, which is divided into two phases: multilayer forward encoding followed by the original ELM-based regression. **b** Implementation of ELM-autoencoder. **c** Layout of one single layer inside the H-ELM

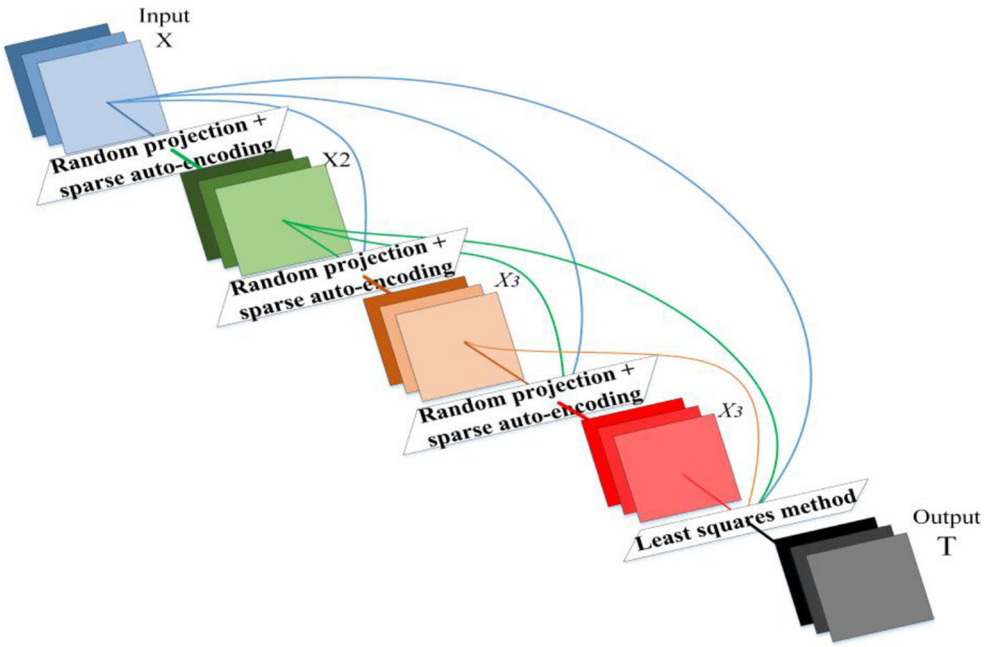

Fig. 4 A 5-layer D-HELM algorithm network framework

the original ELM-based regression is performed for final decision making.

In the unsupervised feature extraction process, first, the training sample $\mathbf{X}$ is standardized, and then, the input weight$\mathbf{a}^{(1)}$and the bias$b^{(1)}$are generated randomly, so the output of the first hidden layer can be represented as

$$\mathbf{H}^{(1)} = g\left(\mathbf{a}^{(1)}\mathbf{X} + b^{(1)}\right) \tag{14}$$

Then, a N-layer unsupervised learning is performed to eventually obtain the high-level sparse features. Mathematically, the output of each hidden layer can be represented as

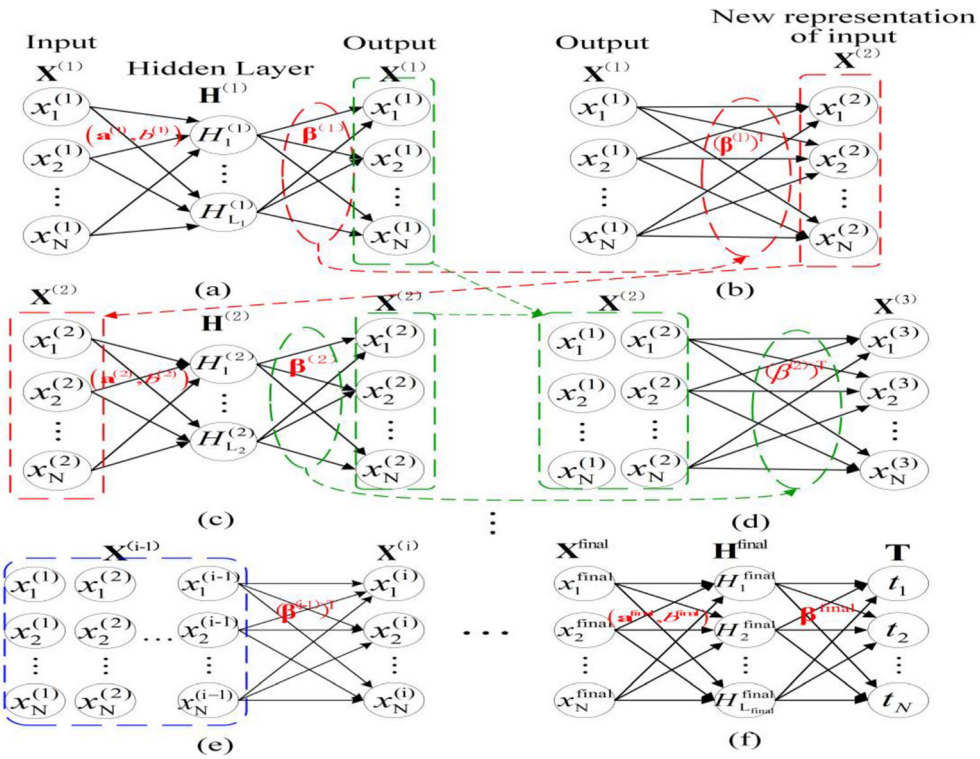

Fig. 5 Schematic diagram of the training process of the D-HELM algorithm

Table 1 Properties of benchmark data sets

Category	Dataset	Features	#Instances
Small	Balance scale	4	625
	Banknote	4	1372
	Liver disorders	6	345
	Cryotherapy	6	90
	E. coli	7	336
	Yeast	7	1484
	Fertility diagnosis	9	100
	Vowel	13	990
Medium	SPECT	22	267
	WDBC	30	569
	WPBC	32	198
	Dermatology	34	366
	Biodeg	42	1055
	SPECTF	44	267
	Sonar image	60	208
	Libras movement	90	360
Large	Urban land cover	148	675
	LSVT	309	126
	Isolet	618	1560
	CNAE	856	1080

$$\mathbf{H}^{(i)} = g\left(\left(\boldsymbol{\beta}^{(i)}\right)^{T}\mathbf{H}^{(i-1)}\right) \tag{15}$$

where $\mathbf{H}^{(i)}$is the output of the ith layer (i∈[1, K]), $\mathbf{H}^{(i-1)}$ is the output of the $(i-1)$th layer, $g(\cdot)$ denotes the activation function of the hidden layers, and $\boldsymbol{\beta}^{(i)}$ represents the output weights of the ith layer. It should be noted that the method of calculating the output weight$\boldsymbol{\beta}^{(i)}$between multiple hidden layers and the output layer of the last layer is different. The output layer weight $\boldsymbol{\beta}^{final}$ of the last layer is derived from equation (8). The output weight$\boldsymbol{\beta}^{(i)}$between hidden layers is obtained by the FISTA algorithm of ELM sparse autoencoder.

Proposed Learning Algorithm

The D-HELM algorithm proposed in this paper connects each layer to every other layer in a feed-forward fashion. In the framework of D-HELM algorithm, each layer takes all preceding feature maps as input, and its own feature maps are used as inputs into all subsequent layers. The combination of feature maps is consistent with the Inception module [27, 28]. The network structure of D-HELM algorithm is shown in Fig. 4.

The D-HELM algorithm is similar to the training process of H-ELM and its training architecture is also structurally divided into two separate phases: (1) unsupervised hierarchical feature representation and (2) supervised feature classification. In the unsupervised feature learning phase, the D-HELM algorithm uses the sparse ELM-AE to learn the characteristics of the input data and randomly maps the features learned by each layer of ELM-AE, ensuring the universal approximation ability of the D-HELM algorithm.

During training, the D-HELM algorithm calculates the output weight $\boldsymbol{\beta}^{(i)}$ of the sparse ELM-AE by using the aforementioned FISTA algorithm. The input $\mathbf{X}^{(i+1)}$ of the next layer is represented by the inner product of the expected output $\mathbf{X}^{(i)}$ of the previous layer and the output weight $\boldsymbol{\beta}^{(i)}$. The mathematical formula is as follows:

$$\mathbf{X}^{(i+1)} = \mathrm{g}\left(\mathbf{X}^{(\mathrm{i})}\boldsymbol{\beta}^{(i)}\right) \tag{16}$$

In the H-ELM algorithm, $\mathbf{X}^{(i)}$ represents the output matrix of the i-th ELM-AE. But in the D-HELM algorithm, $\mathbf{X}^{(i)}$is not only the output of the i-th ELM-AE, but also the output of the original input $\mathbf{X}$ and the ELM-AE before the i-th layer.

We denote the original input $\mathbf{X}$ as $\mathbf{X}^{(1)}$, and $\mathbf{X}^{(i+1)}$ can be expressed by the following form:

$$\mathbf{X}^{(i)} = \left[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(i)}\right] \tag{17}$$

$[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(i)}]$ is a matrix composed of $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(i)}$, their connected mode is similar to the Inception algorithm [28].

After unsupervised feature learning, supervised feature classification is performed. The learned sample feature is taken as the output $\mathbf{H}^{final}$ of the last layer of the hidden layer. The output weight $\boldsymbol{\beta}^{final}$ of the last layer of the network is calculated from the known tag $\mathbf{T}$. The final layer of output weight is calculated in the same way as the original ELM.

$$\mathbf{H}^{final}\boldsymbol{\beta}^{final} = \mathbf{T} \tag{18}$$

According to the least square method and the generalized inverse principle, we can calculate

$$\begin{aligned}\boldsymbol{\beta}^{final} &= \left(\mathbf{H}^{final}\right)^{\dagger}\mathbf{T} \\ &= \left(\mathbf{H}^{final}\right)^{T}\left(\frac{\mathbf{I}}{C} + \left(\mathbf{H}^{final}\right)^{T}\mathbf{H}^{final}\right)^{-1}\mathbf{T}\end{aligned} \tag{19}$$

The D-HELM algorithm is detailed in Algorithm 1.

Algorithm 1 Proposed D-HELM algorithm

Input: input matrix $\mathbf{X}$, output matrix $\mathbf{T}$, regularization $C^{(i)}$ for all layer, input weights $\mathbf{a}^{(i)}$, biases $\mathrm{b}^{(i)}$, and activation $g^{(i)}$, the number of layers N

Output: hidden layer feature representation matrix $\mathbf{H}^{\text{final}}$, output weights $\boldsymbol{\beta}^{\text{final}}$

Step 1: Let $\mathbf{X}^{(1)} = \mathbf{X}$, Calculate $\mathbf{H}^{(1)} \leftarrow g^{(1)}\left(\mathbf{a}^{(1)}\mathbf{X}^{(1)} + b^{(1)}\right)$

Step 2: Calculate $\boldsymbol{\beta}^{(1)} \leftarrow FISTA\left(\mathbf{X}^{(1)}, \mathbf{H}^{(1)}\right)$

Step 3: Calculate $\mathbf{X}^{(2)} \leftarrow g^{(1)}\left(\mathbf{X}^{(1)} \cdot \boldsymbol{\beta}^{(1)}\right)$

For i = 2: N-1 **do**

Step 4: Calculate $\mathbf{H}^{(i)} \leftarrow g^{(i)}\left(\mathbf{a}^{(i)}\mathbf{X}^{(i)} + b^{(i)}\right)$

Step 5: Calculate $\boldsymbol{\beta}^{(i)} \leftarrow FISTA\left(\mathbf{X}^{(i)}, \mathbf{H}^{(i)}\right)$

Step 6: Calculate $\mathbf{X}^{(i+1)} \leftarrow g^{(i)}\left(\mathbf{X}^{(i)}\boldsymbol{\beta}^{(i)}\right)$, where $\mathbf{X}^{(i)} = \left[\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(i)}\right]$.

Step 7: Let i = N, $\mathbf{H}^{\text{final}}\boldsymbol{\beta}^{\text{final}} = \mathbf{T}$, Calculate $\boldsymbol{\beta}^{\text{final}} \leftarrow \left(\mathbf{H}^{\text{final}}\right)^{\mathrm{T}}\left(\frac{\mathbf{I}}{C} + \left(\mathbf{H}^{\text{final}}\right)^{\mathrm{T}}\mathbf{H}^{\text{final}}\right)^{-1}\mathbf{T}$

Return $\mathbf{H}^{\text{final}}$ and $\boldsymbol{\beta}^{\text{final}}$

The training process of the D-HELM algorithm is shown in Fig. 5.

Experiment and Evaluation

In this section, D-HELM algorithm is evaluated over 20 publicly available benchmark data sets from UCI repository [29]. All data sets are, respectively, described in Table 1 and categorized into small, medium, and large in terms of features in order to have a thorough evaluation on the training time and testing accuracy. The model parameters for the best testing accuracy for all data sets are, respectively, described in Table 2.

Experiment Setup

In all the simulations below, the testing hardware and software conditions are listed as follows: Intel(R) Xeon(R) 2.4G CPU, 128G RAM, Windows 7, MATLAB R2017a. Following the practice in [3], the number of hidden layers is set to 3 for all experiments. For H-ELM algorithm and the proposed D-HELM algorithm, the numbers of hidden nodes N_i (i = 1 to 3) are, respectively, set as $10 \times m$ $\{m = 1, 2, \cdots, 40, 40\}$ (the model parameters for the best testing accuracy are described in Table 2). For D-HELM algorithm, it is more sensitive to the change of the regularization parameter C than H-ELM algorithm, respectively, set as 10^x, $\{x = -9, -8, \cdots, 8, 9\}$, but when x is near the optimal value, it changes by 0.1. S is the scaling factor of the activation function, set as $\{1, 2, \cdots, 60, 60\}$.

Model Parameter Analysis

From Table 2, when the accuracy of the algorithm model is highest on the test sample, we can clearly see that the D-HELM's number of hidden nodes N_i (i = 1 to 3) is significantly smaller than the H-ELM algorithm. Especially when the model is getting deeper and deeper, this contrast will be more obvious. This shows that deep ELM models can effectively reduce the number of hidden neurons by densely connected networks at the same network depth. And through Table 3, it can be seen that densely connected network structure can improve the recognition accuracy of the ELM algorithm model.

It can be clearly seen from Fig. 6 that in general, when the coefficient of the penalty term is constant, that is, the penalty power of the penalty function is constant; the testing accuracy of the H-ELM and D-HELM algorithms will increase as the number of hidden layer neurons increases. Similarly, when the number of hidden layer neurons is constant, the testing accuracy of the H-ELM and D-HELM algorithms will increase as the

Table 2 The model parameters for the best testing accuracy

Category	Dataset	H-ELM	D-HELM
Small	Balance scale	N1 = N2 = 100, N3 = 2000, C = 10^1.5, S = 30	N1 = N2 = 60, N3 = 1500, C = 10^2.4, S = 55
	Banknote	N1 = N2 = 200, N3 = 3500, C = 10^-0.1, S = 22	N1 = N2 = 120, N3 = 3000, C = 10^1.6, S = 50
	Liver disorders	N1 = N2 = 100, N3 = 2000, C = 10^1.4,S = 43	N1 = N2 = 60, N3 = 1400, C = 10^1.7,S = 54
	Cryotherapy	N1 = N2 = 120, N3 = 2000, C = 10^-0.1,S = 5	N1 = N2 = 60, N3 = 1000, C = 10^1.0,S = 10
	E. coli	N1 = N2 = 60, N3 = 2500, C = 10^0.2,S = 21	N1 = N2 = 10, N3 = 1200, C = 10^1.7,S = 49
	Yeast	N1 = N2 = 200, N3 = 4000, C = 10^0.3,S = 55	N1 = N2 = 120, N3 = 3500, C = 10^2.6,S = 65
	Fertility diagnosis	N1 = N2 = 80, N3 = 200, C = 10^0.3,S = 10	N1 = N2 = 15, N3 = 110, C = 10^1.0,S = 20
	Vowel	N1 = N2 = 120, N3 = 2800, C = 10^0.7,S = 8	N1 = N2 = 50, N3 = 1800, C = 10^1.9,S = 9
Medium	SPECT	N1 = N2 = 10, N3 = 1000, C = 10^-3.0,S = 2.4	N1 = N2 = 5, N3 = 520, C = 10^-1.7,S = 1.2
	WDBC	N1 = N2 = 200, N3 = 3500, C = 10^0.5,S = 8	N1 = N2 = 100, N3 = 3000, C = 10^3.6,S = 30
	WPBC	N1 = N2 = 60, N3 = 1000, C = 10^1.9,S = 35	N1 = N2 = 40, N3 = 700, C = 10^2.5,S = 40
	Dermatology	N1 = N2 = 60, N3 = 1000, C = 10^0.3,S = 3.3	N1 = N2 = 5, N3 = 500, C = 10^2.0,S = 1.4
	Biodeg	N1 = N2 = 50, N3 = 750, C = 10^-10,S = 5	N1 = N2 = 17, N3 = 700, C = 10^-0.5,S = 7
	SPECTF	N1 = N2 = 40, N3 = 1200, C = 10^-2.3,S = 1.7	N1 = N2 = 20, N3 = 700, C = 10^-2.2,S = 0.9
	Sonar image	N1 = N2 = 100, N3 = 2600, C = 10^1.7,S = 38	N1 = N2 = 30, N3 = 2000, C = 10^2.9,S = 43
	Libras movement	N1 = N2 = 70, N3 = 1300, C = 10^0.5,S = 16	N1 = N2 = 30, N3 = 800, C = 10^1.7,S = 16
Large	Urban land cover	N1 = N2 = 180, N3 = 1200, C = 10^0.4,S = 22	N1 = N2 = 10, N3 = 800, C = 10^2.1, s = 25
	LSVT	N1 = N2 = 120, N3 = 1200, C = 10^-1.5,S = 3.2	N1 = N2 = 5, N3 = 450, C = 10^1.0, s = 2.0
	Isolet	N1 = N2 = 80, N3 = 1600, C = 10^0.1,S = 13	N1 = N2 = 5, N3 = 1000, C = 10^1, s = 0.8
	CNAE	N1 = N2 = 120, N3 = 2600, C = 10^-1.8,S = 6.6	N1 = N2 = 10, N3 = 950, C = 10^1, s = 0.3

penalty factor increases. From Fig. 6, we will find that when the penalty coefficient is small and the number of hidden layer neurons increases, the testing accuracy of the D-HELM algorithm will firstly increase, and then decrease; in addition, the over-fitting phenomenon occurs, while H-ELM does not have above problems. The main reason is that the number of hidden layer neurons in the D-HELM algorithm and the H-ELM algorithm are equal; the D-HELM algorithm uses a densely connected network structure, which increases the number of unknown parameters of the algorithm. It is more prone to over-fitting tendencies at the same time. This can also be seen from Table 2. We can see from Table 2 that when the H-ELM and D-HELM algorithms are trained, the penalty coefficient of the D-HELM algorithm is significantly larger than that of H-ELM algorithm, which also shows that the D-HELM algorithm has a tendency to overfit compared with the H-ELM algorithm. However,

Table 3 The mean of testing accuracy (%) on benchmark data sets

Category	Dataset	H-ELM (%)	D-HELM (%)
Small	Balance scale	98.74 ±0.91	99.10 ±0.22
	Banknote	86.25 ±1.17	88.60 ±0.54
	Liver disorders	65.59 ±3.87	67.33 ±5.8
	Cryotherapy	83.76 ±3.68	88.00 ±4
	E. coli	81.12 ±4.6	85.57 ±5.8
	Yeast	48.94 ±3.19	49.39 ±4.04
	Fertility diagnosis	90.96 ±4.8	92.80 ±4
	Vowel	48.9 ±5.4	50.58 ±4.8
Medium	SPECT	57.47 ±7.76	62.58 ±9.52
	WDBC	89.29 ±1.77	93.43 ±2.36
	WPBC	71.63 ±7.63	84.17 ±2.07
	Dermatology	95.38 ±1.56	96.88 ±0.53
	Biodeg	58.34 ±4.72	64.01 ±9.37
	SPECTF	52.89 ±6.15	55.28 ±9.72
	Sonar image	55.96 ±6.61	63.58 ±8.42
	Libras movement	81.50 ±5	84.34 ±5.56
Large	Urban land cover	74.67 ±3.73	76.24 ±4.51
	LSVT	81.56 ±7.81	90.06 ±6.88
	Isolet	75.73 ±2.32	85.76 ±0.52
	CNAE	92.84 ±2.23	93.41 ±0.74
	Average	74.58	78.57

it is not necessary to afraid the problem that the proposed D-HELM algorithm reduces the accuracy of the algorithm recognition because of over-fitting. Because we can clearly see from Table 2 and Fig. 6b, f, when the test accuracy of the D-HELM algorithm is the highest, the number of hidden layer neurons is much smaller than the H-ELM algorithm. When $N_3 = 3000$, the testing accuracy curve Fig. 6b and the 3D contour map 3.6 (f) of D-HELM algorithm tend to be flat, and the recognition accuracy of D-HELM algorithm has more significant performance than H-ELM algorithm.

Evaluation of Testing Accuracy

The testing accuracy of H-ELM algorithm and D-HELM algorithm in Table 3 is the average result of 100 executions of the algorithm on each benchmark data set under the best model parameters. From the comparison results of the testing accuracy of H-ELM algorithm and D-HELM algorithm in each data set shown in Table 3, D-HELM algorithm is suitable for the datasets of various sample feature sizes. Compared with H-ELM algorithm, D-HELM's densely connected network structure can make full use of the features of the sample when the learning model is shallow (the depth of the model of H-ELM algorithm and D-HELM algorithm in the experiment was only 3 layers), and the recognition result is significantly improved.

Evaluation of Average Training Time

By comparing the training times of H-ELM algorithm and D-HELM algorithm in each data set in Table 4, it is found that when the number of features of the training sample is in the medium or small scale, because the densely connected network structure of D-HELM can reduce the number of hidden neurons to a certain extent, it will make the training time of D-HELM shorter than that of H-ELM. However, when the feature dimension of the training sample is at a very large scale, such as the training samples on the Isolet and the CNAE data sets, even if the number of neurons in each hidden layer of the D-HELM algorithm is smaller than that of the H-ELM model. But, with the deepening of the network structure of D-HELM algorithm, the number of columns of the feature representation matrix output by each hidden layer increases exponentially. Due to the limitations of computer computing capabilities, the training time of the D-HELM algorithm is slightly larger than the H-ELM algorithm.

Comparison with Other Deep Learning Algorithms

In this section, we used more complicated data sets with image patches to verify the learning performance of D-HELM algorithm over deep learning algorithms [including Stacked Auto Encoders (SAE), Stacked Denoising Autoencoder (SDA) [30], Deep Belief Networks (DBN), Deep Boltzmann Machines (DBM), Convolutional Neural Network (CNN), and ML-ELM]. Note that in the experiments, the effects of data preprocessing techniques (e.g., data augmentation) are avoided, and we mainly focused on the verification of learning capability of different deep learning algorithms. For BP-based multilayer perceptron training algorithms (SAE, SDA, DBN, and DBM), the initial learning rate is set as 0.1 with a decay rate 0.95 for each learning epoch. The pretraining and fine-tuning are set as 100 and 200 epochs, respectively. Besides, the input corruption rate of SDA is set at 0.5 with a dropout rate 0.2 for hidden layers. The network structure of the CNN algorithm is 96×96–$24 \times 24 \times 6$–$12 \times 12 \times 6$–$8 \times 8 \times 12$–$4 \times 4 \times 12$–192–5. 96×96 represents the number of neurons in the input layer. $24 \times 24 \times 6$ means six 24×24 convolution kernels in the first layer of convolutional layer, and $12 \times 12 \times 6$

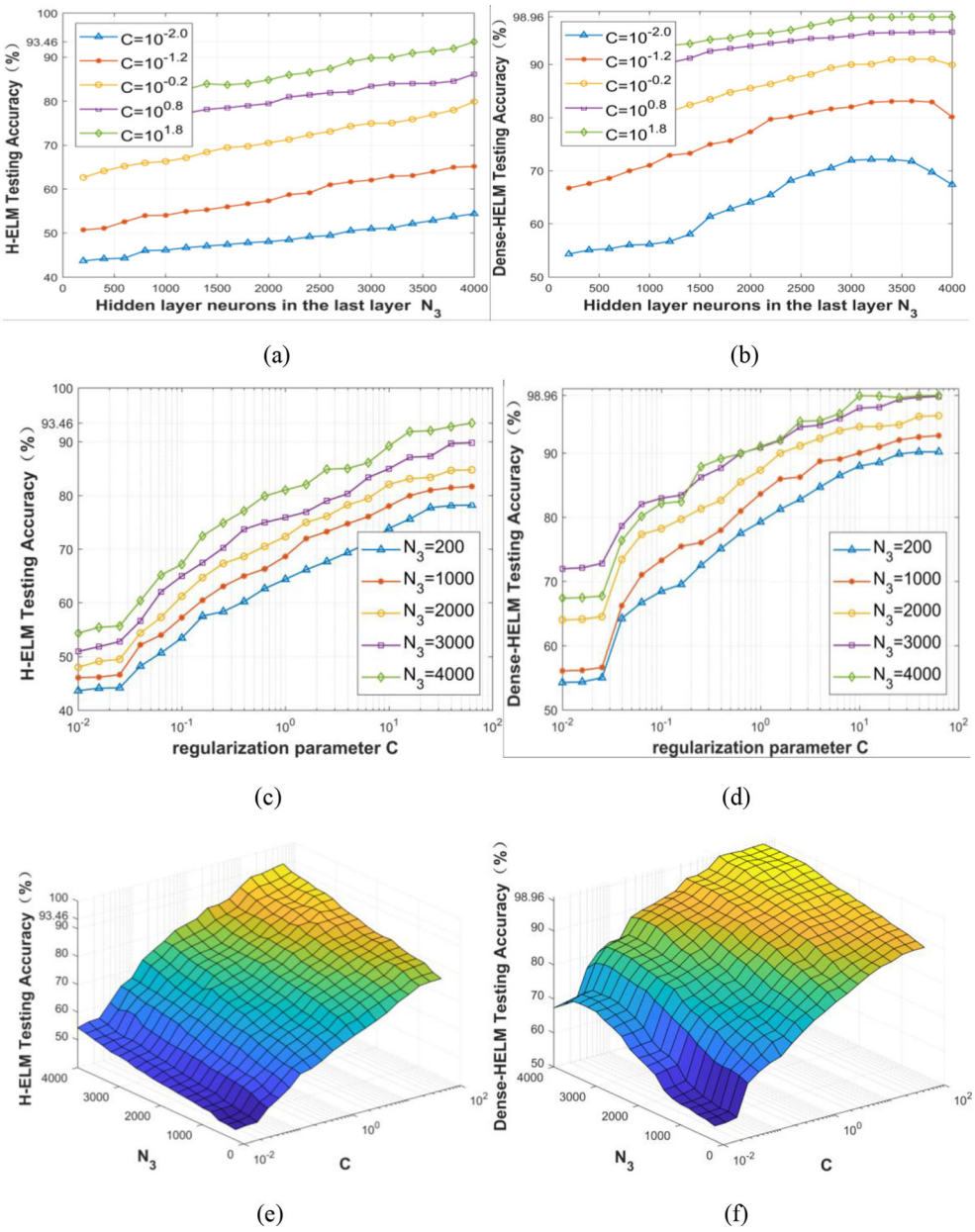

Fig. 6 Comparison of testing accuracy and N_3 and C relationship between H-ELM algorithm and D-HELM algorithm. **a**, **b** The relationship between the testing accuracy of H-ELM algorithm and D-HELM algorithm and the number N_3 of hidden layer neurons in the last layer. **c**, **d** The relationship between the testing accuracy of H-ELM algorithm and D-HELM algorithm and regularization parameter C. **e**, **f** The relationship between the testing accuracy of H-ELM algorithm and D-HELM algorithm and N_3 and C

means six 12×12 convolution kernels in the first pooling layer. The CNN algorithm has two convolution layers and two pooling layers. 192 indicates the number of neurons in the fully connected layer, and 5 indicates the number of neurons in the output layer. The learning rate of the CNN algorithm is 0.1, the batch size is 50, and the number of epochs is 50. The ℓ_2 penalty parameters of the three-layer ML-ELM are set as 10^{-1}, 10^3, and 10^8, respectively.

NORB Dataset: the NYU Object Recognition Benchmark (NORB) dataset [31] is used for 3D object shape recognition experiments, containing images of 50 different 3D toy objects, 5 general categories: (1) four-legged animals, (2) humans, (3) aircraft, (4) trucks, (5) cars, 10 objects per class. The image of each object is taken by two left and right cameras under different viewpoints and different lighting conditions. The training set contains 24,300 pairs of 25 object stereo image pairs (5 pairs per class), while the test set contains the remaining 25 object image pairs.

The testing accuracies of different deep learning algorithms on NORB dataset are shown in Table 5. It can be seen that compared with other time-consuming deep learning algorithms, the proposed D-HELM algorithm achieves 91.35% accuracy with hundreds times faster training time.

Table 4 The average training time (s) on benchmark data sets

Category	Dataset	H-ELM(s)	D-HELM(s)
Small	Balance scale	0.21	*0.11*
	Banknote	0.66	*0.53*
	Liver disorders	0.21	*0.16*
	Cryotherapy	0.14	*0.03*
	E. coli	0.27	*0.08*
	Yeast	1.00	*0.76*
	Fertility diagnosis	0.03	*0.03*
	Vowel	0.38	*0.18*
Medium	SPECT	0.06	*0.04*
	WDBC	0.62	*0.47*
	WPBC	0.06	*0.06*
	Dermatology	0.07	*0.04*
	Biodeg	0.08	*0.08*
	SPECTF	0.08	*0.07*
	Sonar image	0.30	*0.24*
	Libras movement	*0.09*	0.12
Large	Urban land cover	0.20	*0.16*
	LSVT	0.12	*0.08*
	Isolet	*0.27*	0.48
	CNAE	*0.35*	0.38
	Average	0.26	*0.21*

The results are obtained by our proposed alogrithm. Italic entries specified the perofrmance of D-HELM is better

Conclusion

Based on the existing deep extreme learning machine algorithm, this paper proposes a densely connected deep extreme learning machine algorithm: D-HELM. The D-HELM algorithm uses the conventional ELM as the basic structure and uses ELM-AE for feature learning. Its densely connected network structure maximizes the utilization of features for each layer of ELM-AE learning. Moreover, the D-HELM algorithm provides a direct path from the shallow layer to the deep layer in the network structure, which solves the problem that the feature information is reduced in effectiveness after multi-layer abstraction and generalization. Therefore, the D-HELM algorithm significantly improves the classification accuracy based on the H-ELM algorithm. Because the utilization of sample features is improved, the training time of the algorithm can be improved by deleting the number of useless hidden layer neurons. In general, the D-HELM algorithm improves the classification accuracy and shortens the training time. Compared with other deep learning algorithms (CNN, SAE, SDA, DBN, DBM, etc.), the proposed D-HELM algorithm not only maintains the advantage of fast training, but also achieves the best classification accuracy.

Table 5 Comparison of learning accuracy on NORB dataset

Method	Accuracy (%)	Training time (*s*)
SAE [20]	86.28	34,960.69
SDA [30]	87.62	37,909.72
DBN [21]	88.47	50,325.65
DBM [22]	89.65	105,046.72
CNN[31]	90.31	116,010
ML-ELM [3]	88.91	447.01
H-ELM [4]	91.28	249.73
D-HELM	91.35	213.89

Funding Information This work is partially supported by the Key Research and Development Program of China (2016YFC0301400) and Natural Science Foundation of China (51379198, 51075377, and 31202036).

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Informed Consent Informed consent was not required as no human or animals were involved.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Huang GB, Chen L, Siew CK. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw. Jul. 2006;17(4):879–92.
2. Ding S, Zhao H, Zhang Y, Xu X, Nie R. Extreme learning machine: algorithm, theory and applications. Artif Intell Rev. 2015;44(1):103–15.
3. Kasun LLC, Zhou H, Huang GB, Vong CM. Representational learning with extreme learning machine for big data. IEEE Intell Syst. 2013;28(6):31–4.
4. Tang J, Deng C, Huang GB. Extreme learning machine for multilayer perceptron. IEEE Trans Neural Netw Learn Syst. 2016;27(4):809–21.
5. Lv Q, Niu X, Dou Y, Xu J, Lei Y. Classification of hyperspectral remote sensing image using hierarchical local-receptive-field-based extreme learning machine. IEEE Geosci Remote Sens Lett. 2016;13(3):434–8.
6. Yin Y, Li H. Multi-view CSPMPR-ELM feature learning and classifying for RGB-D object recognition. Clust Comput. 2018;5:1–11.
7. Hu L, Chen Y, Wang J, Hu C, Jiang X. OKRELM: online kernelized and regularized extreme learning machine for wearable-based activity recognition. Int J Mach Learn Cybern. 2018;9(9):1577–90.
8. Yan S, Zhang S, Bo H, et al. Gaussian derivative models and ensemble extreme learning machine for texture image classification. Neurocomputing. 2018;277:53–64.

9. Zhai J, Zhang S, Zhang M, Liu X. Fuzzy integral-based elm ensemble for imbalanced big data classification. Soft Comput. 2018;22(11):3519–31.
10. Li Y, Qiu R, Jing S. Intrusion detection system using online sequence extreme learning machine (OS-ELM) in advanced metering infrastructure of smart grid. PLoS One. 2018;13(2):192–216.
11. Chen S, Zhang SF, Zhai JH, et al. Imbalanced data classification based on extreme learning machine autoencoder, 2018. Chendu: *International Conference on Machine Learning and Cybernetics(ICMLC)*; 2018. p. 399–404.
12. Zhang J, Feng L, Yu L. A novel target tracking method based on OSELM. Multidim Syst Sign Process. 2017;28(3):1091–108.
13. Cao JW, Zhang K, Yong HW, Lai XP, Chen BD, Lin ZP. Extreme learning machine with affine transformation inputs in an activation function. IEEE Transactions on Neural Networks and Learning Systems. 2019;30(7):2093–107.
14. Cosmo DL, Salles EOT. Multiple sequential regularized extreme learning machines for single image super resolution. IEEE Signal Processing Letters. 2019;26(3):440–4.
15. Wong CM, Vong CM, Wong PK, Cao J. Kernel-based multilayer extreme learning machines for representation learning. IEEE Trans Neural Netw Learn Syst. 2018;29(3):757–62.
16. Rafael J. On the kernel function. Int Math Forum. 2017;12(14):693–703.
17. Farooq M, Steinwart I. An svm-like approach for expectile regression. Comp Stat Data Anal. 2016;109:159–81.
18. Ramya S, Shama K. Comparison of SVM kernel effect on online handwriting recognition: a case study with Kannada script. Data Eng Intell Comp. 2018;542:75–82.
19. Wu YQ, Ma XD. Alarms-related wind turbine fault detection based on kernel support vector machines. *Journal of Engineering*. 2019;2019(18):4980–5.
20. Jia CC, Shao M, Li S, et al. Stacked denoising tensor auto-encoder for action recognition with spatiotemporal corruptions. IEEE Trans Image Process. 2017;27(4):1878–87.
21. Zhang Y, Li PS, Wang XH. Intrusion detection for IoT based on improved genetic algorithm and deep belief network. IEEE Access. 2019;7:31711–22.
22. Huang G, Liu Z, Weinberger KQ, and Maaten LVD, Densely connected convolutional networks, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, vol. 1, no. 2, pp. 2261–2269, 2017.
23. Zhu M, Song Y, Guo J, Feng C, Li G, Yan T, et al. PCA and kernel-based extreme learning machine for side-scan sonar image classification. In: *Underwater Technology (UT)*. Busan: 2017 IEEE; 2017. p. 1–4.
24. Huang GB. An insight into extreme learning machines: random neurons, random features and kernels. Cogn Comput. 2014;6(3):376–90.
25. Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. Siam J Imaging Sci. 2009;2(1):183–202.
26. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. Las Vegas: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 2818–26.
27. Szegedy C, Ioffe S, Vanhoucke V, and Alemi AA, Inception-v4, Inception-ResNet and the impact of residual connections on learning, *In* 2017 *AAAI*, vol. 4, 2017. http://arxiv.org/abs/1602.07261. Accessed 21 Jul 2020.
28. Dua D, Taniskidou EK. UCI machine learning repository. Irvine: University of California, School of Information and Computer Science; 2017. http://archive.ics.uci.edu/ml. Accessed 21 Jul 2020.
29. Vincent P, Larochelle H, Bengio Y, et al. Extracting and composing robust features with denoising autoencoders. In: *International Conference on Machine Learning*. Helsinki: ACM.; 2008. p. 1096–103.
30. Huang FJ, Yann L. THE small NORB DATASET, V1.0, 2005-05-30/2019-03-03, https://cs.nyu.edu/~ylclab/data/norb-v1.0-small/. Accessed 21 Jul 2020.