IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 31, 2018, accepted January 15, 2019, date of publication January 21, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2894014

Stacked Denoising Extreme Learning Machine Autoencoder Based on Graph Embedding for Feature Representation

HONGWEI GE[1], WEITING SUN[1], MINGDE ZHAO[2], AND YAO YAO[1]
[1]College of Computer Science and Technology, Dalian University of Technology, Dalian 116023, China
[2]School of Computer Science, McGill University, Montreal, QC H3A0E9, Canada

Corresponding author: Hongwei Ge (hwge@dlut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572104, Grant 61402076, and Grant 61103146, in part by the Project of the Key Laboratory of Symbolic Computation and Knowledge Engineering in Jilin University under Grant 93K172017K03, and in part by the Fundamental Research Funds for Central Universities under Grant DUT17JC04.

ABSTRACT Extreme learning machine is characterized by less training parameters, fast training speed, and strong generalization ability. It has been applied to obtain feature representations from the complex data in the tasks of data clustering or classification. In this paper, a graph embedding-based denoising extreme learning machine autoencoder (GDELM-AE) is proposed for capturing the structure of the inputs. Specifically, in GDELM-AE, a graph embedding framework that contains an intrinsic graph and a penalty graph constructed by local Fisher discrimination analysis is integrated into the autoencoder. So, it can exploit both local structure and global structure information in extreme learning machine (ELM) spaces. Further, we propose a stacked graph embedded denoising (SGD)-ELM by stacking several GDELM-AEs. The experimental results on several benchmarks validate that GDELM-AE can obtain efficient and robust feature representation of original data; moreover, the stacked GDELM-AE can obtain high-level and noise-robust representations. The comparative results with the state-of-the-art algorithms indicate that the proposed algorithm can obtain better accuracy as well as faster training speed.

INDEX TERMS Extreme learning machine, stacked autoencoder, denoising, graph embedding.

I. INTRODUCTION

Feedforward neural network is one of the most popular network structures in the tasks of representation and learning. Most of the existing training algorithms of feedforward neural network, such as back-propagation [1] and Levenberg-Marquardt [2], adjust the weights and the biases iteratively by using a certain optimization technique like gradient descent. However, such iterative computational technique suffers from low computational efficiency and easily falls into local optima. This problem greatly limits the generalization ability of the algorithm.

The concept of extreme learning for feedforward neural network was first proposed in 2004 [3]. And then extreme learning machine (ELM) algorithm is proposed to train single-hidden layer feedforward neural (SLFN) networks. Different from traditional learning algorithms of SLFN networks that adjust the network parameters iteratively, ELM randomly initiates the input weights and the hidden layer biases, and determines the output weights by Moore-Penrose generalized inverse. Another different aspect is that the ELM not only tends to seek the smallest training error but also the smallest norm of weights. Barlett's theory [4] on the generalization performance of feedforward neural networks has stated that the smaller training error and the norm of weight reach, the better generalization performance the networks tend to have. Afterwards, the relevant theoretical and applied researches have demonstrated that ELM has the advantages of less training parameters, fast learning speed and strong generalization ability [5]. So far, various extensions of original ELM have been proposed to make it more efficient for some specific applications [6], [7], such as computer vision [8]–[12], image processing [13], text understanding [14]. In [15], the original ELM is extended for semi-supervised and unsupervised tasks by using the manifold regularization to handle multi-classification and clustering. Reference [16] proposes an unsupervised discriminative extreme learning machine for clustering, which combines local manifold learning with global discriminative learning. GEELM proposed

in [17] exploits both local near-neighbor structure and global structure information by constructing intrinsic and penalty graphs under the graph embedding framework. GEELM shows good performance on some standard classification problems and human behavior analysis. However, compared with some deep learning models, GEELM as a SLFN network only has limited representation capability.

In view of the high efficiency of ELM, some researchers devote to studying how to integrate ELMs into the deep learning framework for alleviating the computational complexity and time-consuming problem of deep learning. The multilayer extreme learning machine (ML-ELM) is proposed by stacking several extreme learning machine autoencoders (ELM-AEs) [18]. In ML-ELM, the ELM-AE maps the original data to the ELM feature space with good representation ability. Furthermore, the multilayer structure of ELM-AE can extract high-level and abstract features for original data representation. Different from other deep learning models, such as deep belief network (DBN) [19], stacked autoencoder (SAE) [20], convolutional neural network (CNN) [21], ML-ELM does not require to fine-tune the parameters using back propagation. So it will reduce the computational cost in the training stage. In recent research, a hierarchical extreme learning machine (H-ELM) is proposed for multilayer perceptron [22]. In H-ELM, unsupervised hierarchical feature extraction is followed by supervised feature classification. During the stage of feature extraction, an ELM sparse autoencoder is developed to extract the features of input data; while in the feature classification stage, the original ELM-based regression is performed for final decision making. In [23], a method for synthesizing deep neural networks using ELMs as a stack of supervised autoencoders is presented to handle the multiclass image classification. In [24], a stacked sparse denoising autoencoder based on ridge regression (SSDAE-RR) is proposed for constructing feature representations. The proposed model effectively integrates the advantages of stacked AE, sparse AE, denoising AE, and ridge regression. SSDAE-RR not only increases the robustness to noise, but also reinforces the sparsity of weights, and this makes it easier to discover interesting and prominent features. In [25], a deep neural network called multilayer generalized extreme learning machine autoencoder (ML-GELM) is proposed. It is stacked by several graph regularization extreme learning machine autoencoders (GELM-AEs), which add the manifold regularization to the objective of ELM-AE. The generalization performance of the proposed method is not sensitive to the parameters of the networks.The above-mentioned ELM based deep neural networks for feature extraction mostly exploit underlying structure of the data using unlabeled data, or exploit the intrinsic manifold structure of the data via manifold learning. These methods only exploit the local neighborhood information of data structure. However, the global structure information is not fully considered.

To improve the performance of ELM based deep learning models, we firstly propose a new graph embedded denoising extreme learning machine autoencoder (GDELM-AE), which contains an intrinsic graph and a penalty graph constructed by local Fisher discrimination analysis (LFDA) [26]. The mechanism of graph embedding exploits both local near-neighbor structure and global structure information by maximizing the intraclass similarity and the global interclass separability. Then a stacked graph embedded denoising extreme learning machine (SGD-ELM) is proposed for feature representation by stacking several GDELM-AEs.

The rest of this paper is organized as follows. In Section II, we give the preliminaries about ELM-AE and graph embedding. In Section III, the graph embedded denoising extreme learning machine autoencoder (GDELM-AE) and the framework of stacked graph embedded denoising extreme learning machine (SGD-ELM) are proposed respectively. Extensive experimental comparisons on several benchmark datasets are conducted in Section IV. Finally conclusions are drawn in Section V.

II. RELATED PRELIMINARIES

A. EXTREME LEARNING MACHINE AUTOENCODER (ELM-AE)

Extreme learning machine autoencoder (ELM-AE) [18], [27] is an unsupervised learning algorithm. It is a three-layer neural network with one hidden layer as shown in Fig. 1. The autoencoder tries to learn an approximation that makes the target values equal to the inputs.

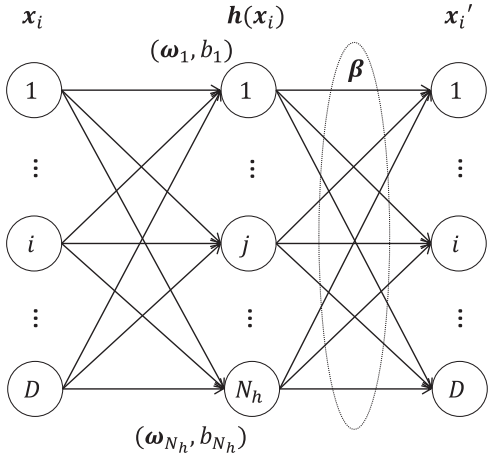

FIGURE 1. The network structure of ELM-AE.

The ELM-AE consists of D input neurons, N_h hidden neurons and D output neurons. Let $x_i \in R^D$ denote an input sample in a D-dimension vector space. In ELM-AE, the input weights $\omega \in R^{D*N_h}$ and the hidden bias values $b \in R^{N_h}$ are randomly generated and then orthogonalized, while the output weights $\beta \in R^{N_h*D}$ are analytically determined through minimizing least squares loss function. According to the size of D and N_h, an ELM-AE can represent the extracted features in three different representation forms: 1) $D > N_h$, compressed representation; 2) $D < N_h$, sparse representation; 3) $D = N_h$, equal dimension representation.

As for the sparse ELM-AE architecture, the feature of input x_i can be represented by mappings from the input space to the ELM space:

$$h(x_i) = g(x_i * \omega + b) \tag{1}$$

where ω and b are subjected to the constraints $\omega\omega^T = I$, $b^T b = 1$. $h(x_i) = [h_1(x_i), \ldots, h_{N_h}(x_i)] \in R^{N_h}$ is the output vector of the hidden layer with respect to input x_i. $g(\cdot)$ is an activation function usually taken as sigmoid function or Gaussian function and so on. I is an identity matrix. The output vector is denoted as $x_i' = x_i = [x_{i1}, \ldots, x_{iD}]$, and it can be obtained by

$$x_{ik}' = \sum_{j=1}^{N_h} \beta_{jk} h_j(x_i), \quad k = 1, \ldots, D \tag{2}$$

Mathematically, Eq.(2) can be expressed in a matrix form as

$$X' = H\beta \tag{3}$$

where $X' = [x_i', \ldots, x_N']$ and $H = [h(x_1), \ldots, h(x_N)]$.

ELM-AE learns to make its target values (outputs) to be equal to its inputs. The objective function of ELM-AE is as follows

$$\arg\min_{\beta} J_{\text{ELM-AE}} = \tfrac{1}{2}\|\beta\|^2 + \tfrac{c}{2}\|\xi\|^2 \tag{4}$$

where $\xi = X - H\beta$. The function is composed of regularization term and training error term. The regularization term controls the complexity of the model to prevent overfitting. The penalty coefficient c of the error term is utilized to balance the two terms in the model. By setting the gradient of $J_{\text{ELM-AE}}$ with respect to β as zero, we have

$$\nabla J_{\text{ELM-AE}} = \beta - cH^T(X - H\beta) = 0 \tag{5}$$

When the number of training samples N is larger than the number of the hidden neurons N_h, the output weights β can be obtained by

$$\beta = (H^T H + \tfrac{I}{c})^{-1} H^T X \tag{6}$$

If the number of training samples N is smaller than the number of the hidden neurons N_h, then H has more columns than rows. In this case, the output weights β can be expressed as the linear combinations of the rows of H: $\beta = H^T\alpha (\alpha \in R^{N*D})$. Then from Eq.(5) we can have

$$\beta = H^T(HH^T + \tfrac{I}{c})^{-1} X \tag{7}$$

Given input data X, we can obtain its feature representation in N_h dimensional space as $X_{\text{ELM-AE}} = X\beta^T$. Then the obtained features can be used as a new representation of the original data and can be applied to the tasks of data clustering or classification more effectively.

B. GRAPH EMBEDDING

The graph embedding model proposed in [28] defines two undirected weighted graphs. The two graphs have the same vertex set, but different edge weight matrices. One is an intrinsic graph $G = \{X, W\}$, where $X = [x_1, \ldots, x_N]$ $(x_i \in R^D)$ is the data matrix of sample set, $W \in R^{N*N}$ is the similarity matrix, and each element of W denotes the similarity between the vertices of G, where N is the sample number and D is the feature dimension. The other is a penalty graph $G^p = \{X, W^p\}$, where edge weight matrix $W^p \in R^{N*N}$ penalizes specific characteristics of the relationships between the training data. The intrinsic graph characterizes the intraclass compactness and it can establish connections between each data point and its neighbor within the same class, while the penalty graph connects the marginal points and characterizes the interclass separability.

The purpose of Graph embedding algorithm is to find a mapping function $f : x_i \to y_i$ that transforms $x_i \in R^D$ into a low-dimensional representation $y_i \in R^d (d < D)$ based on keeping the relationship of vertices in graph. The function f may be explicit or implicit, linear or nonlinear in different cases. Assuming that the mapping is linear as $y_i = {x_i}^T\omega$. Then the objective function of graph embedding can be described as

$$\begin{aligned} \omega^* &= \arg\min_{y^T By = q} \sum_{i \neq j} \|y_i - y_j\|^2 W_{ij} \\ &= \arg\min_{y^T By = q} \sum_{i \neq j} y^T L y \\ &= \arg\min_{\omega^T XBX^T\omega = q} \omega^T XLX^T\omega \end{aligned} \tag{8}$$

where $L \in R^{N*N}$ is the Laplacian matrix of intrinsic graph G, defined as $L = D - W$. D is the diagonal degree matrix and $D_{ii} = \sum_{j=1}^{N} W_{ij}$. In the constraint $\omega^T XBX^T\omega = q$, q is a constant value(usually $q = 1$), $B \in R^{N*N}$ is the constraint matrix defined to avoid a trivial solution of the objective function. Typically, B is a diagonal matrix for scale normalization and can also be the Laplacian matrix of penalty graph G^p. That is $B = L^p = D^p - W^p$.

According to Lagrange multiplier method, Eq.(8) can be transformed into solving the generalized eigenvalue decomposition problem,

$$XLX^T\nu = \lambda XL^pX^T\nu \tag{9}$$

Define $S_r = XLX^T$, $S_p = XL^pX^T$, then Eq.(9) can be represented as

$$S_r\nu = \lambda S_p\nu \tag{10}$$

Therefore, the optimal projection vector ν of the linear graph embedding algorithm can be obtained by solving the generalized eigenvalue of Eq.(10). The eigenvector corresponding to the smallest eigenvalue λ is the desired projection vector. That is, the columns of the transformation matrix ω are formed by the eigenvectors with respect to the first d minimal eigenvalues λ_i for the matrix $S = S_p^{-1}S_r$. When the matrix S_p

is singular, its generalized pseudo-inverse can be used instead of inverse, namely $S = S_p^{\dagger} S_r$. So the matrix $S = S_p^{\dagger} S_r$ can be used to describe the intrinsic relationship and the penalty relationship among the training data.

The second step of the proposed GDELM-AE is linear data mapping process. Graph embedding is introduced to maintain the local and global structure relational between samples. Specially, we construct the intrinsic graph and penalty graph by LFDA to express the structure relationship between samples.

III. STACKED GRAPH EMBEDDED DENOISING ELM (SGD-ELM)

In this section, we firstly propose a graph embedding based denoising extreme learning machine autoencoder (GDELM-AE) for representing the inputs. Furthermore, the framework of stacked graph embedded denoising extreme learning machine (SGD-ELM) is proposed for developing high-level and noise-robust representations.

A. GRAPH EMBEDDED DENOISING EXTREME LEARNING MACHINE AUTOENCODER (GDELM-AE)

ELM-AE can exploit the underlying structure information of unlabeled data for feature representation. Graph regularization extreme learning machine autoencoder (GELM-AE) [25] exploits the latent manifold structure of the local features for unlabeled data. The proposed denoising autoencoder GDELM-AE in this paper exploits both local structure and global structure information by integrating graph embedding, and in which denoising technique further acquires noise-robust features. The structure of GDELM-AE is as shown in Fig. 2. In GDELM-AE, feature representation is achieved by training the autoencoder to output denoised data features from the corresponding data corrupted by noise. First, Gaussian noises are superimposed to the original data, and then the corrupted data is taken as the input data of GDELM-AE to reconstruct clean features. This process consists of two phases: stochastic feature mapping and parameter solving. In the first phase, the input layer weights and hidden layer bias are initialized randomly and processed by orthogonalization, and then these parameters are fixed. The input data is mapped

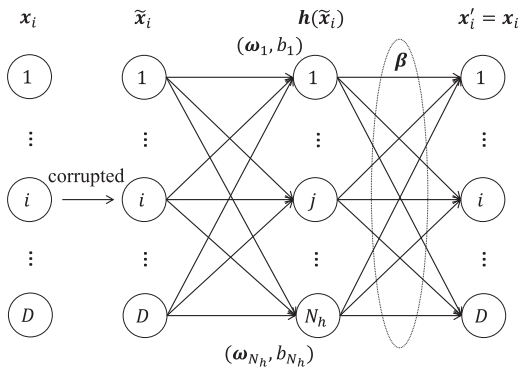

FIGURE 2. The network structure of GDELM-AE.

to the ELM feature space by nonlinear activation function. In the next phase, based on graph embedding framework, an intrinsic graph and a penalty graph constructed by LFDA are integrated into the autoencoder. So it can utilize local and global structure information to enhance feature representation by maximizing the interclass separability and the intraclass similarity.

Given a set of N vectors $\{x_i, c_i\}_{i=1,2,\ldots,N}$, where $x_i \in R^D$ is sample data, and $c_i \in \{1, \ldots, C\}$ is the corresponding class labels. For each training sample x_i, its corresponding label vector is denoted as $y_i = [y_{i1}, \ldots, y_{ij}, \ldots, y_{iC}]$, if $c_i = k$, then $y_{ik} = 1$, otherwise, $y_{ik} = -1$. First the corrupted data $\tilde{x_i}$ is constructed by adding Gaussian noise on the original data x_i. Then the corrupted input $\tilde{x_i}$ is mapped into the ELM space using Eq.(1) for acquiring the representation in hidden layer. It can be written as $H = [h(\tilde{x_1})^T \; h(\tilde{x_2})^T \; \ldots \; h(\tilde{x_N})^T]$ in the form of matrices. In this paper, the following objective function is established for GDELM-AE:

$$\arg\min_{\beta} J_{\text{GDELM-AE}} = \frac{1}{2}\|\beta\|^2 + \frac{c}{2}\sum_{i=1}^{N}\|\xi_i\|^2 + \frac{\lambda}{2} tr(\beta^T S \beta) \quad (11)$$

where $\xi_i = x_i - h(\tilde{x_i})\beta$, $i = 1, \ldots, N$. The matrix S is used to express the local near-neighbor structure and global structure in ELM space. λ is a tradeoff parameter between the two regularization terms of β and $\lambda > 0$. The first term in Eq.(11) is a regularization term which controls the complexity of the model, and prevents overfitting. The second term is a training error term, and c is the penalty coefficient. The third term is a regularization term based on graph embedding, which contributes to the intraclass similarity and the interclass separability in ELM space.

GDELM-AE employs graph embedding regularization to discover intrinsic structure and penalty structure in ELM space to analyze the nonlinear relationships between the input data $\tilde{x_i}$. We use hidden layer output vectors $h(\tilde{x_i})$, $i = 1, 2, \ldots, N$ to construct an intrinsic graph $G = \{H, W\}$, where $W \in R^{N*N}$ is a similarity matrix. The element W_{ij} of matrix W denotes the similarity between the graph vertices $h(\tilde{x_i})$ and $h(\tilde{x_j})$, and it is expected to be maximized. Besides, we construct a penalty graph $G^p = \{H, W^p\}$, where weight matrix $W^p \in R^{N*N}$ penalizes the relationships among the graph vertices, and it is also expected to be maximized. The matrices $S_r = HLH^T$ and $S_p = HL^pH^T$ can be defined, where $L = D - W$ and $L^p = D^p - W^p$ are the graph Laplacian matrices of G and G^p respectively. Then structure information matrix S can be defined as $S = S_p^{-1} S_r$. When S_p is singular, S can be defined as $S = S_p^{\dagger} S_r$, where $S_p^{\dagger}$ is the pesoinverse matrix of S_p. It should be noted that in order to better express the nonlinear relationship among the input data, the calculation of the matrix S is in the ELM space rather than in the input space.

The weight matrices W and W^p in intrinsic graph and penalty graph can be obtained with different criteria. In this

paper, we adopt LFDA to obtain the intrinsic structure and penalty structure for the input data. So the weight matrices W and W^p can defined by

$$W_{ij} = \begin{cases} \frac{\omega_{ij}}{N_{c_i}} & if\ c_i = c_j \\ 0 & otherwise \end{cases} \tag{12}$$

$$W^p_{ij} = \begin{cases} \omega_{ij}(\frac{1}{N} - \frac{1}{N_{c_i}}) & if\ c_i = c_j \\ \frac{1}{N} & otherwise \end{cases} \tag{13}$$

where ω_{ij} is a measure of the similarity between $h(\tilde{x}_i)$ and $h(\tilde{x}_j)$ based on Euclidean distance, and it can be defined by $\omega_{ij} = \exp(-(\|h_i - h_j\|_2^2)/(2\sigma^2))$, σ is a parameter scaling the Euclidean distance, usually it is taken as the mean value of Euclidean distance. N_{c_i} denotes the number of samples in class c_i. Note that $1/N - 1/N_{c_i}$ in (13) is negative while $1/N$ and $1/N_{c_i}$ in (13) (14) are positive. And it weights the value for the samples pairs in the same class according to the affinity. This implies it trends to keep the sample pairs in the same class close and the sample pairs in different class apart. What's more, far apart data pairs in the same class are not imposed to be close.

By solving the equation $\partial J_{\text{GDELM-AE}}/\partial\beta = 0$, then output weights β can be obtained as

$$\beta = (HH^T + \frac{1}{c}(I + \lambda S))^{-1} HX^T \tag{14}$$

where $HH^T + 1/c(I + \lambda S)$ is nonsingular. For the given input data X, its representation in N_h-dimensional ELM space can be defined as $X_{new} = X\beta^T$. Finally we use the new learned representation to replace the original data in the tasks of data clustering or classification. Algorithm 1 shows the pseudo-code of the GDELM-AE algorithm.

Algorithm 1 GDELM-AE

Input :
Data set $X = \{x_i, c_i\}, i = 1, 2, \ldots, N$;
The number of hidden neurons N_h;
Activation function $g(\cdot)$;

Output:
The feature representations of the inputs.

1 Corrupt original data X into $\tilde{X}$ using Gaussian noise;
2 Randomly generate input weights ω and hidden bias b, and then make them orthogonal;
3 Calculate the network hidden layer output $H = g(\tilde{X} * \omega + b)$;
4 Calculate graph weights W and W^p with Eq.(12) and Eq.(13);
5 Calculate graph Laplacian matrices $L = D - W$ and $L^p = D^p - W^p$;
6 Calculate structure information matrix $S = S_p^{\dagger} S_r$ where $S_r = XLX^T$ and $S_p = XL^pX^T$;
7 Calculate the output weights β with Eq.(14);
8 Obtain the new feature representation $X_{new} = X\beta^T$;

B. STACKED GRAPH EMBEDDED DENOISING ELM (SGD-ELM)

In this section, a stacked graph embedding denoising extreme learning machine (SGD-ELM) is proposed by stacking several GDELM-AEs to capture the high-level feature representation of the data. The structure of SGD-ELM and the details of the algorithm are given as follows.

SGD-ELM is a stacked multilayer neural network model which employs GDELM-AE as a base building block, and trained by a layer-by-layer greedy training method. Each GDELM-AE is trained by the feedforward way, and the output of the previous GDELM-AE is used as the input of the next GDELM-AE. The network structure of SGD-ELM is shown in Fig. 3. The hidden layer weights in SGD-ELM are inherited from the GDELM-AEs. The output matrix H_k of the kth hidden layer can be expressed as $H_k = g(H_{k-1}\beta_k^T)$, where β_k is the output weight matrix of the kth GDELM-AE. The input data $\tilde{X}$ can be considered as the 0th hidden layer where $H_0 = \tilde{X}$. Assume that there are m hidden layers in the SGD-ELM, the weights between the last hidden layer and the output layer in SGD-ELM are obtained by regularized least squares. The final objective function of SGD-ELM is taken as:

$$\arg\min_{\beta_{m+1}} J_{\text{SGD-ELM}} = \frac{1}{2}\|\beta_{m+1}\|^2 + \frac{c}{2}\|Y - H_m\beta_{m+1}\|^2 \tag{15}$$

where $Y = [y_1, \ldots, y_i, \ldots, y_N], y_i = [y_{i1}, \ldots, y_{iC}]$ is the label vector corresponding to x_i, H_m is the output matrix of the last hidden layer in SGD-ELM. If the number of training samples N is larger than the number of the hidden neurons N_{hm}, the output weights β_{m+1} can be obtained by

$$\beta_{m+1} = (H_m^T H_m + \frac{I}{c})^{-1} H_m^T Y \tag{16}$$

Otherwise, β_{m+1} can be obtained by

$$\beta_{m+1} = H_m^T (H_m H_m^T + \frac{I}{c})^{-1} Y \tag{17}$$

The motivation of GDELM-AE is to acquire efficient and noise-robust feature representation in ELM space by exploiting the local near-neighbor structure information and global structure information among the original input data. The stacked model SGD-ELM is to extract higher-level feature representation of the data. So the proposed model can be used to address some issues related to feature representation, such as classification, clustering and annotation. Taking classification as an example, the pseudo-code of the proposed SGD-ELM algorithm is shown as Algorithm 2. And Fig. 4 shows the flowchart of the proposed SGD-ELM.

This section analyzes the time complexity of the proposed SGD-ELM algorithm. In GDELM-AE, output weights β are calculated by Eq.(14), in which matrix S describes the graph structure. When only intrinsic graph structure is considered,

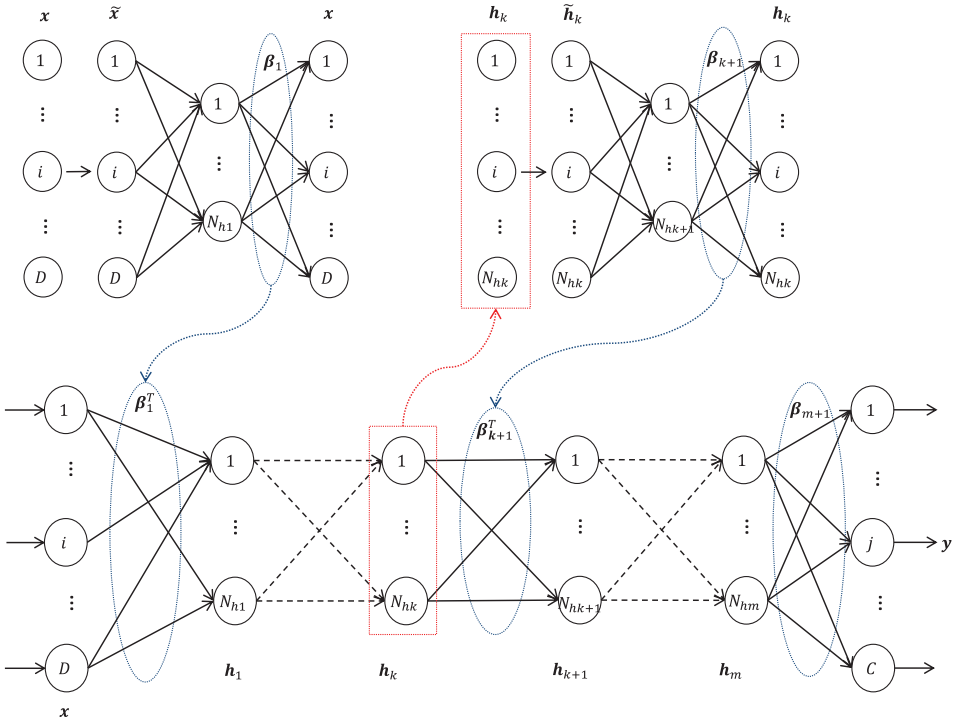

FIGURE 3. The network structure of SGD-ELM.

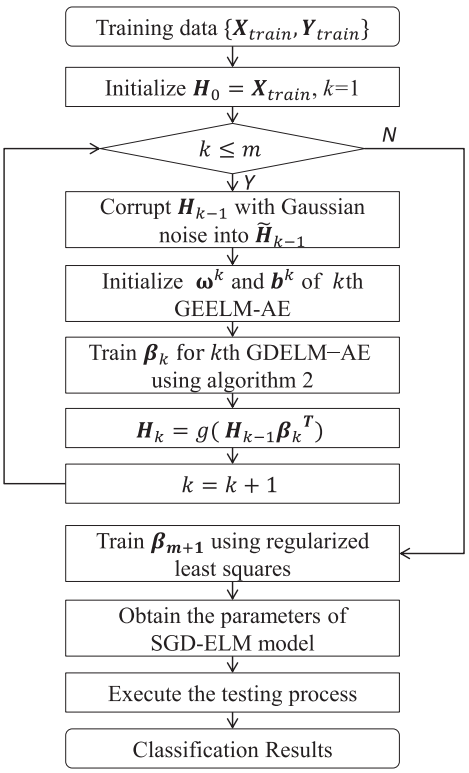

FIGURE 4. Flowchart of the proposed SGD-ELM algorithm.

the matrix S can be expressed as $S = S_i = HLH^T$, and the time complexity of the calculation of matrix S is equal to $O(N_h^2 N)$. When both the intrinsic graph structure and the penalty graph structure are considered, the time complexity of matrix S is equal to $O(N_h^3 + N_h^2 N)$. Subsequently, the time complexity of matrix multiplication and matrix inversion is $O(N_h^3 + N_h^2 N)$. Thus, the overall time complexity of GDELM-AE is equal to $O(N_h^3 + N_h^2 N)$. SGD-ELM is a stacked model based on GDELM-AE. If SGD-ELM consists of m stacked GDELM-AEs, the overall time complexity of SGD-ELM is $O(m(N_h^3 + N_h^2 N))$. Usually the value of m is not very large, and it is taken as 3 in this paper. Therefore, the time complexity of SGD-ELM is equivalent to $O(N_h^3 + N_h^2 N)$.

IV. EXPERIMENTAL STUDIES

A. DATASETS AND EXPERIMENTAL ENVIRONMENT

To test the performance of the proposed GDELM-AE and SGD-ELM, a series of experiments on several benchmark datasets are conducted and the results are compared with the state-of-the-art algorithms. The relevant attributes of the used benchmark datasets are summarized in Table 1. The IRIS, WINE, LIVER, SATIMAGE are public classification datasets from UCI machine learning repository [29]. The YALEB [30] is a face recognition dataset. The COIL20 [31] is a multiclass image classification benchmark from Columbia University image database. And the USPST is a subset (the testing set) of the handwritten digit recognition data set of USPS [32]. All the experiments are carried out on a PC with Xeon (R) CPU 5-2609 2.50GHz processor, 32 GB memory using Matlab 2014a.

B. RESULTS AND COMPARISONS FOR GDELM-AE

In order to validate the feature representation of GDELM-AE, we conduct some clustering experiments on the IRIS,

Algorithm 2 SGD-ELM for Classification

Input :
Training data
$(X_{train}, Y_{train}) = \{x_i, y_i\}, i = 1, 2, \ldots, N$;
Testing data
$(X_{test}, Y_{test}) = \{x_i, y_i\}, i = 1, 2, \ldots, M$;
The deep architecture depth m;
The number of hidden neurons of each GDELM-AE: $N_{h1}, N_{h2}, \ldots, N_{hm}$;
Activation function $g(\cdot)$;
Output:
The classification results of test data.

Train:
Initialize $H_0 = X_{train}$;
for *k=1:m* **do**
  Corrupt H_{k-1} with Gaussian noise into $\tilde{H}_{k-1}$;
  Initialize the input weights ω^k and hidden bias b_k of the kth GDELM-AE;
  Train the output weights β^k of the kth GDELM-AE using Algorithm 1 ;
  Compute the outputs $H_k = g(H_{k-1}\beta_k^T)$, obtain the new feature representation H_k;
end
Calculate β_{m+1} using Eq.(16) or Eq.(17);
Test:
Compute the classification results of the testing data X_{test} by using the above trained SGD-ELM model.

TABLE 1. The attributes of the benchmark datasets.

Datasets	Categories	Dimension	Sample size
IRIS	3	4	150
YALEB	15	1024	165
WINE	3	13	178
LIVER	2	6	345
COIL20	20	1024	1440
USPST	10	256	2007
SATIMAGE	6	36	6435

WINE, YALEB, LIVER and COIL20 datasets, in which the feature representations of the original data obtained by GDELM-AE are employed. For comparisons, the proposed algorithm GDELM-AE is compared with other existing unsupervised learning algorithms, which include k-means, ELM-AE [18] and GELM-AE [25]. ELM-AE uses a standard ELM autoencoder to exploit the underlying structure information of unlabeled data for feature representation. GELM-AE adds the manifold regularization into the objective of ELM-AE to learn the intrinsic structure relationship of the data.

In the experiments, the parameters are selected concerning the quality of solutions and the incurred computational efforts. The number of hidden neurons in ELM-AE, GELM-AE and GDELM-AE is selected from 500, 1000 and 2000. The hyper parameters c and λ in the regularization term of the object function are selected from the exponential sequence $\{10^{-4}, 10^{-1}, 10^{0}, \ldots, 10^{6}\}$ according to the experimental performance. The ELM based autoencoders adopt sigmoid function as the activation function. The basic k-means algorithm implements clustering in the original data space, while ELM-AE, GELM-AE and GDELM-AE perform clustering based on the acquired feature representations from their different ELM embedding spaces. The Gaussian noise added into the training data for the proposed GDELM-AE is a signal with zero mean and variance $\sigma = 0.01$. We test the ability of feature representation for the original data in each dataset using these algorithms independently.

To evaluate the performance of each algorithm, clustering accuracy (ACC) is employed to measure the experimental results. ACC is defined as

$$ACC = \frac{\sum_{i=1}^{N} \delta(c_i, map(t_i))}{N} \tag{18}$$

where N is the number of training samples, t_i and c_i are the predicted label and the true label of x_i, respectively. $\delta(c_i, map(t_i))$ is an indicator function, and if $c_i = map(t_i)$, $\delta(c_i, map(t_i)) = 1$, otherwise $\delta(c_i, map(t_i)) = 0$. The function $map(\cdot)$ is an optional permutation function that maps each cluster label to a category label by the Hungarian algorithm [33].

The algorithm is repeated 30 times independently and the best value, the mean value and the standard derivation of the 30 runs are presented in Table 2. The best results among the five tested approaches are highlighted with boldface. It can be seen that the proposed GDELM-AE achieves the highest value for the mean accuracy over all the datasets compared with other algorithms. Moreover, the proposed algorithm obtains the highest value for the best accuracy for four out of five benchmark datasets and and the lowest value for the standard derivation for three out of five. The experimental results indicate that the proposed GDELM-AE can enhance feature representation by exploiting the local and global structure information of the original data. Besides, the smaller standard deviation indicates that the proposed model has better robustness. To validate the noise robust feature representation of GDELM-AE further, we also provide a set of experiments that the Gaussian noise with the same noise level is added to test data, and the obtained results (GDELM-AE-noise) are also listed in Table 2. It can be seen that the obtained results are superior to the results obtained by the other three compared algorithms, and slightly worse than those obtained by GDELM-AE that has no noise added to the test data. The results indicate that the proposed model can acquire noise-robust feature representation.

Fig. 5 shows a box diagram of the results obtained by each algorithm, in which the middle mark labels present the mean value of 30 trails in each dataset using different algorithms and the lines with horizontal bars present the standard deviation. It can be seen that the middle mark labels of GDELM-AE in each dataset occupy the highest position, which means GDELM-AE get the best performance of mean value compared with other algorithms. And the length of horizontal bars

TABLE 2. Experimental results obtained by GDELM-AE and other compared algorithms.

Datasets	Metrics	GDELM-AE	GDELM-AE-noise	GELM-AE	ELM-AE	k-means
IRIS	Average±std	$\mathbf{93.16 \pm 0.52}$	92.96 ± 0.92	92.44 ± 3.94	84.44 ± 0.63	81.81 ± 13.45
	Best	93.33	93.33	**94**	85.33	89.33
YALEB	Average±std	$\mathbf{54.49 \pm 3.78}$	53.50 ± 2.81	44.93 ± 3.51	42.87 ± 4.44	41.78 ± 3.65
	Best	**60.61**	**60.61**	50.3	50.91	49.7
WINE	Average±std	$\mathbf{98.11 \pm 0.28}$	97.79 ± 0.51	96.37 ± 0.78	91.87 ± 0.90	95.89 ± 5.20
	Best	**98.32**	**98.32**	97.19	94.38	96.63
LIVER	Average±std	$\mathbf{60.91 \pm 1.86}$	60.77 ± 1.75	56.83 ± 3.52	54.55 ± 0.12	55.19 ± 0.57
	Best	**63.19**	**63.19**	60.90	54.78	56.23
CIOL20	Average±std	$\mathbf{68.47 \pm 3.69}$	67.04 ± 3.65	61.78 ± 4.99	59.39 ± 4.66	59.88 ± 4.71
	Best	**76.18**	73.68	71.25	68.96	69.17

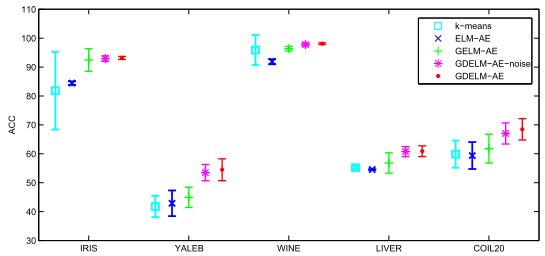

FIGURE 5. The box diagram of the results obtained by GDELM-AE and other algorithms.

of GDELM-AE is almost the shortest in most of the datasets, which demonstrates the stability of GDELM-AE.

To illustrate the abilities of feature representation for different algorithms intuitively, Fig. 6 gives a kind of visual express that describes the clustering results for IRIS dataset in different feature spaces. Fig. 6 (a) shows the visualization in the original feature space, Fig. 6 (b) (c) and (d) show the visualization in the ELM-AE embedding space, the GELM-AE embedding space and the GDELM-AE embedding space respectively. In the figures, the black marks indicate the clustering center, and the red circles represent the wrong data point of the clustering result. From Fig. 6, it can be observed that better clustering results can be obtained by using the feature representations learned from the proposed GDELM-AE. Moreover, it can be observed that the clustering results in the feature space obtained by GDELM-AE is relatively compact within the same cluster and separable between the different clusters.

C. RESULTS AND COMPARISONS FOR SGD-ELM

In order to validate the performance of SGD-ELM, the proposed algorithm is tested for classification on six benchmarks (LIVER, IRIS, WINE, SETIMAGE, USPST and COIL20). The proposed SGD-ELM algorithm is compared with several deep neural models (DBN [19], SAE [20], ML-ELM [18] and ML-GELM [25]). DBN is a generative graphical model using stacked restricted Boltzmann Machines (RBMs). Each RBM is trained greedily in a layer-by-layer manner. After pre-training, all layers are then jointly fine-tuned using the conventional backpropagation algorithm. SAE is similar to DBN, which replaces RBM with autoencoder (AE). AEs are stacked in a greedy layer-wise fashion for pre-training the weights and the features from the stacked autoencoders are fed to a softmax classifier for classification. Traditional deep learning models are computationally complex and time-consuming. The deep learning models ML-ELM and ML-GELM employ the ELM as the stacked base block. ML-ELM is stacked by several unsupervised ELM autoencoders. ML-GELM embeds the manifold learning into the ELM autoencoders and stacks them to form a multilayer network.

In the experiments, 50% of the samples in each dataset are used as training set, and another 50% as the testing set. In training process, Gaussian noise is added to the input data of each GDELM-AE with zero mean and variance $\sigma = 0.01$. For testing data, the original data are used in all the algorithms. The parameters of SGD-ELM include the depth of multilayer model m, the number of hidden layer neurons for each GDELM-AE module $N_{hk}(k = 1, 2, \ldots, m)$, and the regularization term hyper parameters c_k and $\lambda_k(k = 1, 2, \ldots, m)$ for each GDELM-AE. Parameter selection is challenging for deep structure due to the massive parameters analysis or experiments. In this paper, we select the widely used parameter values in some references [18], [34], and which have shown that such parameter values can usually obtain a better performance. Here, m is taken as 3. The hyper parameters c_k and λ_k are selected from the exponential sequence set $\{10^{-6}, 10^{-5}, \ldots, 10^{6}\}$, which are determined by the 5 fold cross-validation strategy applied on the training data set. To ensure the proportion of different classes, the samples of each category in training set are randomly divided into five parts, and four of which are used for training and the rest are used for verification. The average result after five cross validations is used to determine the parameters c_k and λ_k. In SGD-ELM, the number of hidden neurons ranges from 1000 to 5000 with an interval of 100 and it is also determined by cross validation. For fair comparisons, the selection strategy of the number of hidden neurons in the DBN, SAE, ML-ELM, ML-GELM is the same to that in SGD-ELM. Besides, the lost function used in DBN and SAE is cross-entropy cost function, which is defined as follows:

$$L = -\sum_i p_i \log(\hat{p}_i) - \sum_i (1 - p_i) \log(1 - \hat{p}_i) \qquad (19)$$

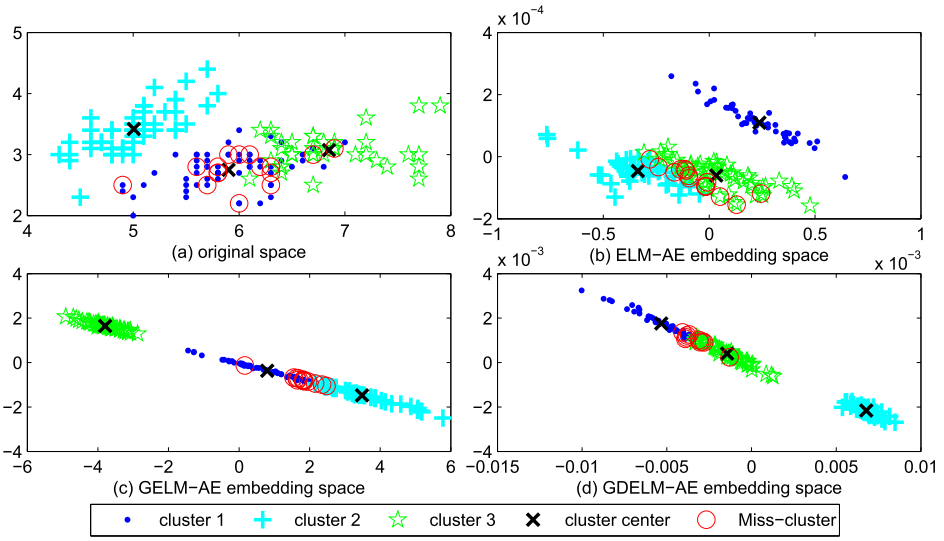

FIGURE 6. A visualization of clustering for IRIS data in different feature spaces. (a) the original space, (b) the ELM-AE embedding space and (c) the GELM-AE embedding space (d) the GDELM-AE embedding space.

TABLE 3. Comparison of experimental results for SGD-ELM and other compared algorithm.

Datasets	Metrics	SGD-ELM	ML-GELM	ML-ELM	SAE	DBN
IRIS	Average±std	$\mathbf{98.58 \pm 0.78}$	97.15 ± 0.54	93.73 ± 2.78	95.07 ± 2.95	96.67 ± 1.44
	Best	**100.00**	97.33	97.33	**100.00**	98.67
WINE	Average±std	$\mathbf{99.09 \pm 0.96}$	96.67 ± 0.48	94.56 ± 1.85	90.78 ± 0.54	96.00 ± 1.83
	Best	**100.00**	98.89	96.67	91.11	98.89
LIVER	Average±std	$\mathbf{73.80 \pm 1.26}$	70.08 ± 2.96	58.03 ± 3.61	61.97 ± 3.96	59.65 ± 0.53
	Best	**76.16**	75.72	64.74	65.90	60.69
COIL20	Average±std	99.85 ± 0.15	$\mathbf{99.90 \pm 0.99}$	96.58 ± 0.70	98.81 ± 0.20	98.85 ± 0.09
	Best	**100.00**	**100.00**	97.50	99.17	99.03
USPST	Average±std	$\mathbf{93.98 \pm 0.55}$	93.72 ± 0.48	89.78 ± 1.58	93.69 ± 0.33	88.96 ± 0.81
	Best	**95.02**	94.53	92.04	94.13	90.25
SATIMAGE	Average±std	$\mathbf{91.13 \pm 0.20}$	88.77 ± 0.44	85.52 ± 0.53	87.67 ± 1.33	86.20 ± 0.66
	Best	**91.43**	89.53	86.64	88.97	87.36

To evaluate the feature representation capacity of each algorithm, the obtained features by different models are input to the same fully connected classifier, and classification accuracy is employed to measure the performance. Each algorithm is repeated 30 times independently and the best accuracy, the average accuracy and the standard derivation of the 30 runs are presented in Table 3. It can be seen that the SGD-ELM obtains the highest average accuracy for five out of the six benchmark datasets compared with other deep neuron networks. And the average accuracy in COIL20 dataset is just slightly lower than other algorithms. It demonstrates that the proposed SGD-ELM stacked by several GDELM-AEs can enhance the feature representation. Besides, it achieves smallest standard deviation in most datasets, which indicates the stability of performance. Fig. 7 shows the box diagram of the results obtained by each algorithm.

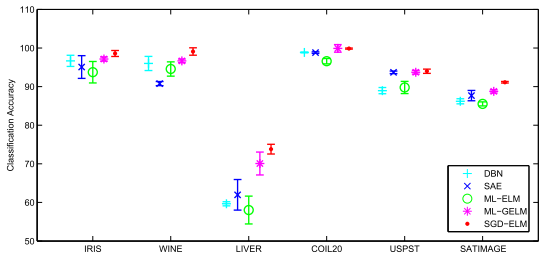

FIGURE 7. The box diagram of the results obtained by SGD-ELM and other algorithms.

To further illustrate the abilities of feature representation for the proposed stacked model intuitively, Fig. 8 shows an example of the visualization that describes the classification results for IRIS testing dataset in different layer feature spaces. Fig. 8 (a) shows the visualization of the classification result in the original data space, Fig. 8 (b), (c) and (d) show the visualization of the classification results using the features from the first hidden layer, the second hidden layer and the third hidden layer in SGD-ELM respectively. It can be seen that the algorithm can achieve better classification result when the features obtained from the higher layer are used. It indicates that stacked model enhances the feature representation of the original data, and more effective and

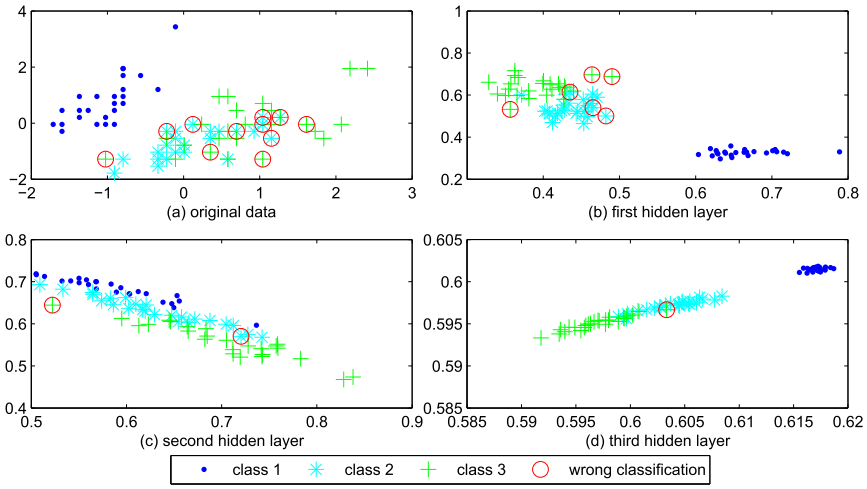

FIGURE 8. An example of the classification visualization in different layer feature spaces for IRIS data.

TABLE 4. Experimental results of SGD-ELM and other compared algorithm.

Datasets	Metrics	IRIS	WINE	LIVER	COIL20	USPST	SATIMAGE
$\sigma = 0$	Average±std	98.58 ± 0.78	99.09 ± 0.96	73.80 ± 1.26	99.85 ± 0.15	93.98 ± 0.55	91.13 ± 0.20
	Best	100.00	100.00	76.16	100.00	95.02	91.43
$\sigma = 0.005$	Average±std	97.92 ± 1.59	98.94 ± 0.99	73.41 ± 1.07	99.73 ± 0.15	93.86 ± 0.77	91.09 ± 0.23
	Best	100.00	100.00	76.74	100.00	94.92	91.55
$\sigma = 0.01$	Average±std	97.29 ± 1.24	98.63 ± 0.70	73.22 ± 1.90	99.70 ± 0.17	93.79 ± 0.59	91.09 ± 0.25
	Best	98.67	100.00	75.00	100.00	94.92	91.52
$\sigma = 0.02$	Average±std	96.98 ± 0.85	98.59 ± 0.58	73.18 ± 1.03	99.69 ± 0.23	93.74 ± 0.81	91.07 ± 0.21
	Best	98.67	100.00	74.42	100.00	94.92	91.52
$\sigma = 0.05$	Average±std	96.49 ± 0.82	98.52 ± 0.89	73.16 ± 0.84	99.65 ± 0.19	93.48 ± 0.60	91.07 ± 0.24
	Best	97.33	100.00	74.42	100.00	94.63	91.55
$\sigma = 0.1$	Average±std	96.44 ± 0.81	98.00 ± 0.79	72.98 ± 0.89	99.56 ± 0.11	93.11 ± 0.62	91.07 ± 0.22
	Best	97.33	100.00	75.00	99.86	94.13	91.64
$\sigma = 0.2$	Average±std	96.62 ± 1.15	97.00 ± 1.57	73.00 ± 0.91	99.45 ± 0.14	92.36 ± 0.77	91.06 ± 0.23
	Best	100.00	100.00	74.42	99.86	93.63	91.67
$\sigma = 0.5$	Average±std	96.13 ± 1.37	93.33 ± 2.33	72.91 ± 0.95	99.42 ± 0.11	91.64 ± 0.83	91.02 ± 0.25
	Best	98.67	97.78	74.42	99.72	92.83	91.52
$\sigma = 1.0$	Average±std	96.09 ± 1.31	85.96 ± 3.97	72.87 ± 0.87	99.32 ± 0.09	90.09 ± 1.13	90.97 ± 0.26
	Best	98.67	93.33	74.42	99.86	92.13	91.43
$\sigma = 1.5$	Average±std	94.22 ± 2.20	80.07 ± 3.74	72.67 ± 1.20	99.16 ± 0.10	87.26 ± 1.30	90.74 ± 0.29
	Best	98.67	85.56	74.42	99.44	90.61	91.18
$\sigma = 2.0$	Average±std	83.07 ± 4.09	75.44 ± 4.08	72.52 ± 1.24	99.11 ± 0.09	78.82 ± 1.92	90.41 ± 0.32
	Best	92.00	83.33	75.58	99.31	87.95	91.02
$\sigma = 3.0$	Average±std	66.18 ± 5.51	69.59 ± 4.47	72.03 ± 1.23	99.09 ± 0.14	67.37 ± 2.01	89.37 ± 0.45
	Best	74.67	75.56	75.00	99.31	79.88	90.28

higher-level feature representation can be acquired as the depth of networks increases.

To further reveal the feature representation capability of the proposed SGD-ELM for the noise data, different degrees of Gaussian noise are added to the test data. The adopted Gaussian noise is a signal with zero mean and variance σ, and σ is taken as 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 1.5, 2.0, 3.0 respectively. Each experiment is repeated 30 times independently, and the average results in each dataset are summarized in Table 4. Fig. 9 shows the curves of the average classification accuracy obtained by SGD-ELM for different degrees of noise data in each dataset. It can be seen that the clustering accuracies in LIVER, COIL20 and SATIMAGE datasets decrease slightly with the increase of noise level from variance $\sigma = 0$ to $\sigma = 3$. However, the clustering accuracies in WINE, IRIS and USPST datasets decreased rapidly when the variance σ of Gaussian noise is greater than 1, especially for WINE dataset, whose classification accuracy decreases almost linearly. So the proposed model has different noise-resistibility for different datasets. In general, the proposed SGD-ELM can achieve satisfactory classification accuracy when the variance σ of Gaussian noise is less than 1.0, and it indicates that the proposed SGD-ELM can obtain noise-robust feature representation.

In addition, to compare the time complexity of SGD-ELM with other algorithms, the training time of each algorithm

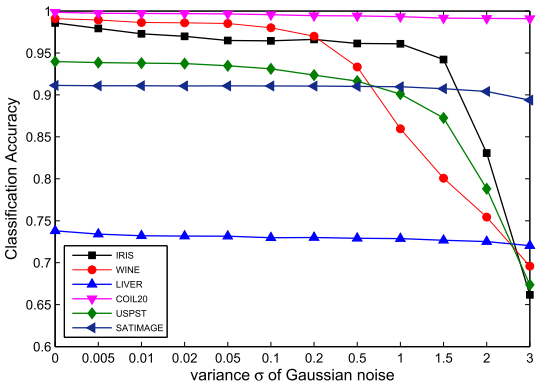

FIGURE 9. The curves of the average classification accuracy obtained by SGD-ELM for different degrees of noise data in each dataset.

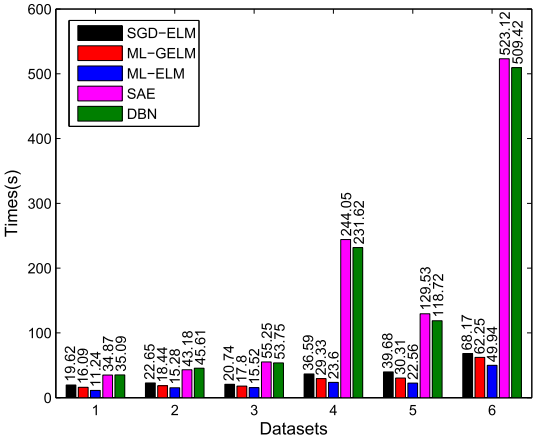

FIGURE 10. The line chart of average results of original data and corrupted data with different degree noise in each dataset.

in each dataset is recorded. Fig. 10 shows the histogram of the average training time of 30 runs for each deep model algorithm on five datasets sorted by the number of samples ($N_{IRIS} < N_{WINE} < N_{LIVER} < N_{COIL20} < N_{USPST} < N_{SATIMAGE}$). Among these algorithms, with the increase of the number of samples, the training speed of SGD-ELM is obviously faster than the conventional depth models SAE and DBN, and it is basically comparable with ML-GELM and ML-ELM. The reason lies in that the ELM based deep models do not need to learn the network parameters iteratively. It can also be demonstrated by the analysis of time complexity in Section III-B. The experiments here illustrate SGD-ELM not only can obtain more effective feature representation, but also has the advantages of faster training speed compared with traditional deep neural network models.

V. CONCLUSION

Multilayer neural networks can transform the original input feature to a high-level and abstract representation that can be better classified or discriminated, but they usually have high computational complexity because they rely on time-consuming procedure to perform iterative optimization of parameters. This paper takes the advantages of extreme learning machine and proposes a stacked denoising extreme learning machine model based on graph embedding (SGD-ELM) for feature representation. Firstly, a graph embedding based denoising extreme learning machine autoencoder (GDELM-AE) is proposed for capturing the structure of the inputs. GDELM-AE integrates an intrinsic graph and a penalty graph constructed by local Fisher discrimination analysis so that it can effectively exploit both local near-neighbor structure and global structure information of data in ELM space. To further enhance the feature representation, several GDELM-AEs are stacked to form the multilayer model (SGD-ELM) for obtaining high-level and noise-robust representations. A series of experiments on several benchmark datasets are conducted and the results are compared with the state-of-the-art algorithms. The comparative results demonstrate that the proposed algorithm can obtain better accuracy as well as faster training speed. Besides, the experiments with different degrees of Gaussian noise indicate that the proposed SGD-ELM can obtain noise-robust feature representation.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[2] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[3] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural network," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, Jul. 2004, pp. 985–990.

[4] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 525–536, Mar. 1998.

[5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.

[6] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 107–122, Jun. 2011.

[7] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Netw.*, vol. 61, pp. 32–48, Jan. 2015.

[8] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum class variance extreme learning machine for human action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 11, pp. 1968–1979, Nov. 2013.

[9] B. He *et al.*, "Fast face recognition via sparse coding and extreme learning machine," *Cognit. Comput.*, vol. 6, no. 2, pp. 264–277, Jun. 2014.

[10] A. A. Mohammed, R. Minhas, Q. J. Wu, and M. A. Sid-Ahmed, "Human face recognition based on multidimensional PCA and extreme learning machine," *Pattern Recognit.*, vol. 44, no. 10, pp. 2588–2597, Oct. 2011.

[11] B. Dash, S. Rup, A. Mohapatra, B. Majhi, and M. N. S. Swamy, "Multi-resolution extreme learning machine-based side information estimation in distributed video coding," *Multimedia Tools Appl.*, vol. 77, no. 20, pp. 27301–27335, Oct. 2018.

[12] Y. Yi, S. Qiao, W. Zhou, C. Zheng, Q. Liu, and J. Wang, "Adaptive multiple graph regularized semi-supervised extreme learning machine," *Soft Comput.*, vol. 22, no. 11, pp. 3545–3562, Jun. 2018.

[13] L. An and B. Bhanu, "Image super-resolution by extreme learning machine," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2012, pp. 2209–2212.

[14] S. Poria, E. Cambria, G. Winterstein, and G.-B. Huang, "Sentic patterns: Dependency-based rules for concept-level sentiment analysis," *Knowl.-Based Syst.*, vol. 69, pp. 45–63, Oct. 2014.

[15] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2405–2417, Dec. 2014.
[16] Y. Peng, W. L. Zheng, and B. L. Lu, "An unsupervised discriminative extreme learning machine and its applications to data clustering," *Neurocomputing*, vol. 174, pp. 250–264, Jan. 2016.
[17] A. Iosifidis, A. Tefas, and I. Pitas, "Graph embedded extreme learning machine," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 311–324, Jan. 2016.
[18] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 31–34, Nov. 2013.
[19] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[20] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[22] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.
[23] M. D. Tissera and M. D. Mcdonnell, "Deep extreme learning machines: Supervised autoencoding architecture for classification," *Neurocomputing*, vol. 174, pp. 42–49, Jan. 2016.
[24] L.-L. Cao, W.-B. Huang, and F.-C. Sun, "Building feature space of extreme learning machine with sparse denoising stacked-autoencoder," *Neurocomputing*, vol. 174, pp. 60–71, Jan. 2016.
[25] K. Sun, J. Zhang, C. Zhang, and J. Hu, "Generalized extreme learning machine autoencoder and a new deep neural network," *Neurocomputing*, vol. 230, pp. 374–381, Mar. 2017.
[26] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1027–1061, May 2007.
[27] C. L. Lekamalage *et al.*, "Dimension reduction with extreme learning machine," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3906–3918, Aug. 2016.
[28] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
[29] C. Blake and C. Merz, "UCI repository of machine learning databases," Dept. Inf. Comput. Sci., Univ. California, Irvin, Irvine, CA, USA, Tech. Rep., 1998. [Online]. Available: http://archive.ics.uci.edu/ml/index.php
[30] K.-C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, May 2005.
[31] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *J. Mach. Learn. Res.*, vol. 12, pp. 1149–1184, Jul. 2011.
[32] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.
[33] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logistics*, vol. 52, pp. 7–21, Feb. 2005.
[34] N. Zhang, S. Ding, and J. Zhang, "Multi layer ELM-RBF for multi-label learning," *Appl. Soft Comput.*, vol. 43, pp. 535–545, Jun. 2016.

HONGWEI GE received the B.S. and M.S. degrees in mathematics from Jilin University, China, and the Ph.D. degree in computer application technology from Jilin University, in 2006. He is currently a Professor with the College of Computer Science and Technology, Dalian University of Technology, Dalian, China. His research interests include machine learning, computational intelligence, optimization and modeling, deep learning, and representation learning. He has published more than 80 papers in these areas. His research was featured in the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART A: SYSTEMS AND HUMANS, *Pattern Recognition*, and *Information Science*.

WEITING SUN received the B.S. and M.S. degrees from the Dalian University of Technology, China, in 2015 and 2018, respectively, where she is currently pursuing the Ph.D. degree with the College of Computer Science and Technology. Her main research interests include machine learning, representation learning, neural networks, and extreme learning machine.

MINGDE ZHAO received the B.S. degree in computer science and technology from the Dalian University of Technology, Dalian, China, in 2018. He is currently pursuing the master's degree in computer science with the School of Computer Science, McGill University, Montreal, Canada. His research interests include artificial intelligence and machine learning.

YAO YAO received the B.S. and M.S. degrees from the Dalian University of Technology, Dalian, China, in 2015 and 2018, respectively, where she is currently pursuing the Ph.D. degree with the College of Computer Science and Technology. Her main research interests include machine learning, deep learning, and computer vision.