Multilayer Extreme Learning Machine With Subnetwork Nodes for Representation Learning

Yimin Yang, *Member, IEEE*, and Q. M. Jonathan Wu, *Senior Member, IEEE*

***Abstract*—The extreme learning machine (ELM), which was originally proposed for "generalized" single-hidden layer feedforward neural networks, provides efficient unified learning solutions for the applications of clustering, regression, and classification. It presents competitive accuracy with superb efficiency in many applications. However, ELM with subnetwork nodes architecture has not attracted much research attentions. Recently, many methods have been proposed for supervised/unsupervised dimension reduction or representation learning, but these methods normally only work for one type of problem. This paper studies the general architecture of multilayer ELM (ML-ELM) with subnetwork nodes, showing that: 1) the proposed method provides a representation learning platform with unsupervised/supervised and compressed/sparse representation learning and 2) experimental results on ten image datasets and 16 classification datasets show that, compared to other conventional feature learning methods, the proposed ML-ELM with subnetwork nodes performs competitively or much better than other feature learning methods.**

***Index Terms*—Dimension reduction, extreme learning machine (ELM), feedforward neural network (NN), representation learning.**

I. Introduction

THE PERFORMANCE of machine learning methods is heavily dependent on the choice of data representation (or features) to which they are applied. For that reason, much of the actual effort in deploying machine learning algorithms goes into the design of processing pipelines and data transformations that result in a representation of the data that can support effective machine learning. Feature selection and extraction techniques are designed to reduce dimensionality by finding a meaningful feature subset or feature combinations, because high dimensionality significantly increases the time and space requirements for processing the data [1]. In fact, feature selection and extraction techniques can be conducted in a supervised, unsupervised or semi-supervised manner, based on whether the label information is used to guide the selection of relevant features. Supervised feature extraction methods normally evaluate feature importance by the correlation between feature and class labels and require a large amount of labeled training data. Typical supervised feature extraction methods include fisher score, linear discrimination analysis (LDA), neighborhood components analysis (NCA), locality sensitive discriminant analysis (LSDA) [2], isometric projection (IsoP) [3], maximum margin projection (MPP) [4], neighborhood preserving embedding (NPE) [5], and so on. Unsupervised feature extraction usually finds the best subspace or projection mapping function to preserve the geometrical structure of the data space. And the mapping function essentially projects the data points along the dimensions of maximum variances. Typical unsupervised feature extraction methods include principal component analysis (PCA), linear graph embedding (LGE) [6], unsupervised locality preserving projection (LPP) [7], and wrapper approach based on expectation maximization. To overcome the disadvantage of supervised or unsupervised learning algorithms that can only make use of unlabeled data or labeled data, semi-supervised feature selection methods have been proposed to leverage both labeled and unlabeled data. Semi-supervised approaches can be roughly categorized into three groups [8]. The first group is based on the clustering assumption that most data examples, including both labeled and unlabeled examples, should be far away from the decision boundary of the target classes. The representative approaches in this category include semi-supervised support vector machine (SVM) [9] or semi-supervised extreme learning machine (ELM) [10]. The second group is based on the manifold assumptions that the input data follows some cluster structure or low-dimensional manifold in the input space. The well-known algorithms in this category include manifold regularization [11], kernel learning [12], and so on. The third group is based on the smoothness assumption. It states that if two points in a high-density region are close, then their corresponding labels should be the same or consistent. The representative approaches in this category include semi-supervised local spline regression [13], smooth graphs [14], and so on.

In 2006, Hinton and Salakhutdinov [15] initiated a breakthrough in representation (feature) learning, which was quickly followed up in the following years [16]–[18]. Hinton and Salakhutdinov [15] and Vincent *et al.* [17] showed that multilayer networks with the backpropagation (BP) learning method can be used to reduce the dimensionality of data. These data features can be used to train multiple-layer neural networks (NNs), or deep learning. Similar to deep networks, recent research developments [18]–[20] propose ELMs with multilayer network architecture for representation learning.

Manuscript received March 30, 2015; revised July 8, 2015 and September 16, 2015; accepted September 20, 2015. This work was supported by the Natural Sciences and Engineering Research Council of Canada. This paper was recommended by Associate Editor X. Wang. *(Corresponding author: Q. M. Jonathan Wu.)*

The authors are with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B3P4, Canada (e-mail: yangyi_min@126.com; jwu@uwindsor.ca).

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

Digital Object Identifier 10.1109/TCYB.2015.2481713

Different from pure feature extraction methods, the main purpose of which is dimension reduction, the purpose of learning representations of data is to extract useful information when building classifiers or other predictors. In the case of probabilistic models, a good representation often captures the posterior distribution of the underlying explanatory factors for the observed input. A good representation is also useful input to a supervised predictor. As mentioned in [17] and [18], general performance could be better when expanded dimension data are used. Benefitting from auto-encoder multilayer structure, the representation learning model can be not only used for compressed data (feature extraction or dimension reduction), but can also be used to generate sparse data.

ELMs provide a unified learning framework for "generalized" single-hidden layer feedforward NNs (SLFNs), including but not limited to sigmoid networks, radial basis function networks, threshold networks, trigonometric networks, fuzzy inference systems, high-order networks wavelet networks, etc. According to conventional NN theories, SLFNs are universal approximators when all parameters of the networks $(\mathbf{a}, b, \beta)$ (e.g., the hidden node parameters and the output weights) are allowed to be adjustable. But unlike these conventional learning methods, the ELM [21]–[28] is a full-random learning method that differs from the usual understanding of learning. In ELM methods, the hidden layer parameters of NN need not be tuned during training, but generated randomly. Huang *et al.* [31] demonstrate that ELM can approximate any continuous target function with any required level of accuracy. Recent ELM developments [29]–[31] show that ELM unifies wide type of SLFNs and SVM/LS-SVM. Also, applications of ELM have recently been presented in sparse learning and classification [32], [33], computer vision [34], clustering learning [10], image processing [35], high-dimensional and large-data applications, etc. [36], [37].

Unlike current single-hidden-layer ELM methods, in this paper, we indicate that with general hidden nodes (or called subnetwork nodes) added to existing networks, ML-ELM can be used for classification, dimensions reduction, and feature learning. In particular, this paper makes the following contributions.

1) This paper provides a learning platform with unsupervised/supervised and compressed/expanded representation learning. The learning speed of the proposed method can be several to tens times faster compared to deep networks such as deep belief networks (DBNs) and stacked auto-encoders (SAE). Furthermore, our platform can provide a much better generalization performance than other feature extraction methods such as LGE, LPP, IsoP, and LDA, etc.
2) Different from current NN-based learning methods, which are sensitive to the combination of parameters, experimental results show that generalization performance of the proposed method are not sensitive to the parameters of the networks. Thus, the user can choose parameters randomly at the outset without affecting the generalization performance in the learning process.
3) Unlike other ELM-based learning methods, in which universal approximation capability only exists in single-hidden-layer NN architecture, this paper shows that ML-ELM with subnetwork nodes not only has universal approximation capability, but also the capability for representation learning. Thus, this paper extends ELM theories from single-hidden-layer architecture to multilayer network with subnetwork nodes architecture.

TABLE I
NOTATIONS TO BE USED IN THE PROPOSED METHOD

Notation	Meaning
$(\boldsymbol{\alpha}, \flat)$	a hidden node.
$(\mathbf{a}, b)$	a general hidden node (or subnetwork node).
$\hat{\mathbf{a}}_f^j$	input weight of the jth general hidden node in feature mapping layer. $\hat{\mathbf{a}}_f^j \in \mathbf{R}^{d\times n}$
$\hat{b_f^j}$	bias of the jth general hidden node in feature mapping layer $\hat{b_f^j} \in \mathbf{R}$.
$(\boldsymbol{\alpha}_{fi}^j, \flat_f^j)$	the ith hidden node in the jth general hidden node.
$(\hat{\mathbf{a}_h}, \hat{b}_h)$	hidden nodes in ELM learning layer and $\hat{\mathbf{a}_h} \in \mathbf{R}^{m\times d}$.
u_j	normalized function in the jth general node, $u_j(\cdot): \mathbf{R} \to (0, 1]$, u_j^{-1} represent its reverse function.
$\mathbf{H}_f^j$	feature data generated by j general nodes in a feature mapping layer, i.e., $\mathbf{H}_f^j = \sum_{i=1}^{j} u_i^{-1} \cdot g(\mathbf{x}, \hat{\mathbf{a}}_f^i, \hat{b}_f^i), \mathbf{H}_f^j \in \mathbf{R}^{d\times M}$
$\mathbf{H}_f^{ji}$	feature data generated by the ith feature mapping layer
M	number of training samples.
n	input data dimension.
m	output data dimension.
d	feature data dimension
$\mathbf{e}_L$	the residual error of current two-layer network (L general nodes in the first layer and $(\mathbf{a}_h, b_h)$ in the second layer).
$\mathbf{e}_L^j$	the residual error of current two-layer network (L general nodes in the first layer and j general nodes in the second layer).
L	the numbers of general hidden nodes

II. Preliminaries and Basic-ELM

A. Notations

The sets of real numbers is denoted by $\mathbf{R}$. For M arbitrary distinct samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^M$ ($\mathbf{x}_i \in \mathbf{R}^n, \mathbf{y}_i \in \mathbf{R}^m$), $\mathbf{x}$ denotes the input data and $\mathbf{y}$ denotes the desired output data. $\boldsymbol{\alpha}_i$ is the weight vector connecting the ith hidden nodes and the input nodes, $\flat_i$ is the bias of the ith hidden nodes. β_i is the output weight between the ith hidden node and the output nodes. $\mathbf{e}$ denotes the residual error of current network output, i.e., $\mathbf{e} = \mathbf{y} - \mathbf{f}$. $\mathbf{I}$ is unit matrix. sum($\mathbf{e}$) denotes the sum of all elements of the matrix $\mathbf{e}$. g is a sigmoid or sine activation function. Other notations are defined in Table I.

B. Basic-ELM

For M arbitrary distinct samples $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} \in \mathbf{R}^{n\times M}$ and $\mathbf{y} \in \mathbf{R}^{m\times M}$. ELM is proposed for SLFNs and output function of ELM for SLFNs is

$$f_n(\mathbf{x}) = \sum_{i=1}^{n} \beta_i g(\mathbf{x}, \boldsymbol{\alpha}_i, \flat_i). \tag{1}$$

According to [38], from the learning point of view, ELM theory aims to reach the smallest training error but also the smallest norm of output weights

$$\text{Minimize: } \|\beta_i\|_p^{\mu_1} + C\left\|\sum_{i=1}^{n} \beta_i g(\mathbf{x}, \boldsymbol{\alpha}_i, \flat_i) - \mathbf{y}\right\|_q^{\mu_2}, i = 1, \ldots, n \tag{2}$$

where $\mu_1 > 0, \mu_2 > 0, p, q = 0, 1/2, 1, 2, \ldots, +\infty$, C is a positive value, $g(\mathbf{x}, \boldsymbol{\alpha}, \flat)$ is referred to as ELM feature mapping

or Huang's transform. Huang *et al.* have proved the following lemma.

Lemma 1 [39], [66]: Given M arbitrary distinct samples $\{(\mathbf{x}, \mathbf{y})\}$, $\mathbf{x} \in \mathbf{R}^{n\times M}$, and $\mathbf{y} \in \mathbf{R}^{m\times M}$ sampled from a continuous system, an activation function g, then for any continuous target function $\mathbf{y}$ and any function sequence $g(\mathbf{x}, \boldsymbol{\alpha}_n^r, \flat_n^r)$ randomly generated based on any continuous sampling distribution, $\lim_{n\to\infty} \|\mathbf{y} - (f_{n-1} + g(\mathbf{x}, \boldsymbol{\alpha}_n^r, \flat_n^r)\| = 0$ holds with probability one if

$$\beta_n = \frac{\langle \mathbf{e}_{n-1}, g(\mathbf{x}, \boldsymbol{\alpha}_n^r, \flat_n^r)\rangle}{\|g(\mathbf{x}, \boldsymbol{\alpha}_n^r, \flat_n^r)\|^2} \tag{3}$$

where $(\boldsymbol{\alpha}_n^r, \flat_n^r)$ represents the nth random hidden node, and $\mathbf{e}_{n-1} = \mathbf{y} - f_{n-1}$.

III. Proposed Method

A. ELM With Subnetwork Nodes

Inspired by [19], [31], [66], here we find that a hidden node itself can be a subnetwork formed by several hidden nodes which naturally forms biological learning, and thus results in learning features (see Fig. 1). In this sense, a single mapping layer can contain multiple networks. There are three differences between ELM feature mapping layer and our feature mapping layer.

1) Standard ELM randomly generates hidden nodes one by one or block by block [see Fig. 1(a)]. Unlike standard ELM, subnetwork nodes (general hidden neurons) in our method are used to instead of single hidden nodes [see Fig. 1(b)].
2) The number of hidden nodes (L) and dimension of outputs (m) in ELM are independent. In our mapping layers, the number of general nodes and output dimension are also independent, but the number of hidden nodes in each general neuron should equal the dimension of outputs. In other words, the number of hidden neurons in a general hidden neuron (d) in Fig. 1(b) should equal output nodes (m).
3) Node in the ELM feature mapping in Fig. 1(a) is a special case of the subnetwork node shown in Fig. 1(b).

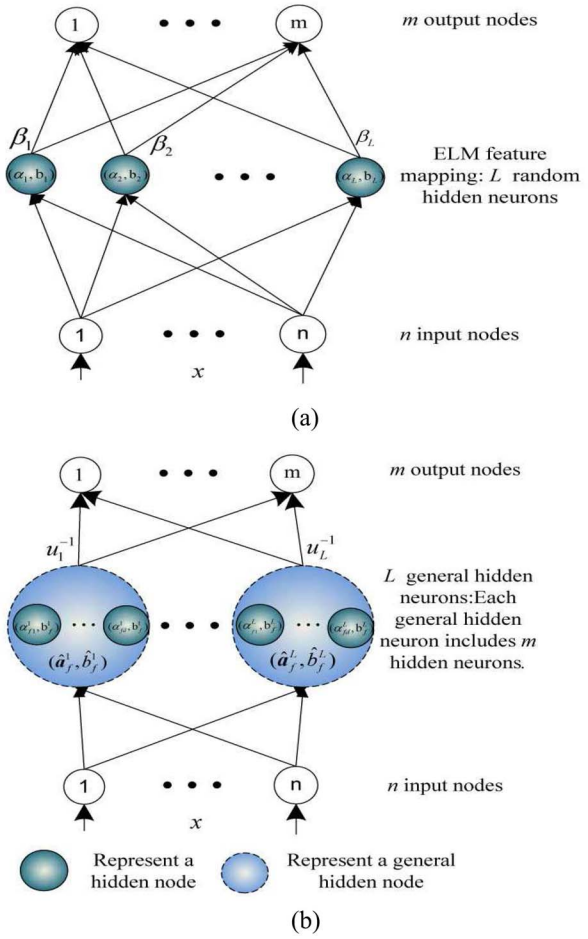


Fig. 1. Difference and relationship among ELM and our method. (a) ELM feature mapping. (b) Our feature mapping.

B. Proposed Method for Representation Learning

Based on the structure of our feature mapping layer, in this section we give the structure of our method in Fig. 2 and then indicate the main function of the proposed method in Definition 1 and Remark 1 in Section III-B1. Then, we give the learning steps of our method in Section III-B2.

1) Optimal Projecting Parameters and Optimal Feature Data: We give the structure of our method in Fig. 2. Our main objective is to represent the input features meaningfully in several different representations as follows.

1) *Dimension Reduction:* Represent features from a higher dimensional input data space to a lower dimensional feature space. If we set $\hat{\mathbf{a}}_f^j, j = 1, \ldots, L$ belongs to $\mathbf{R}^{d\times n}, d < n$, feature data $\mathbf{H}_f(\mathbf{x}_k, \hat{\mathbf{a}}_f, \hat{b}_f)$ can be called compressed features. Thus, the proposed method can be used for dimensionality reduction of data.

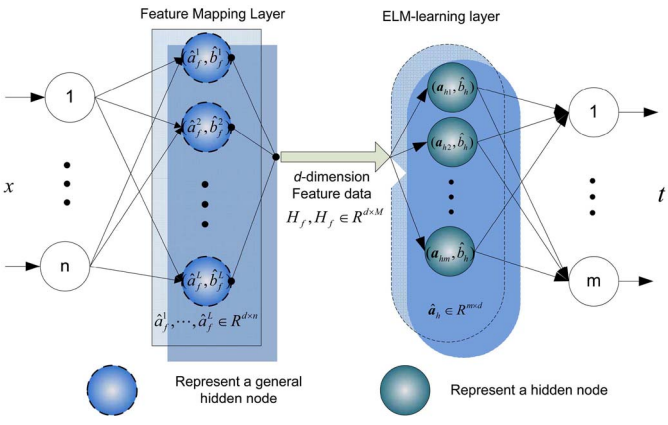


Fig. 2. Structure of our proposed method. $d < n$: dimension reduction, $d > n$: pexpanded dimension representation, outputs $\mathbf{t} = \mathbf{x}(m = n)$: unsupervised learning, outputs $\mathbf{t}$ equals desired label: supervised learning.

2) *Expanded Dimension Representation:* Represent features from a lower dimensional input data space to a higher dimensional feature space. If we set $\hat{\mathbf{a}}_f^j$, $j = 1, \ldots, L$, $d > n$, feature data $\mathbf{H}_f(\mathbf{x}_k, \hat{\mathbf{a}}_f + \hat{b}_f)$ can be called expanded-dimension data.
3) *Unsupervised/Supervised Learning Mode:* If we set outputs $\mathbf{y} = \mathbf{x}$, the features data are generated by unlabeled input data. If we set outputs $\mathbf{y}$ as equal to desired labels,

the feature data are obtained by a supervised learning model.

Based on the above analysis, unlike the standard ELM network, our multilayer network can be used for supervised/unsupervised expanded-dimension representation, dimensionality reduction, and classification. Thus, if we can find an optimal feature mapping layer, we see that the feature data ($\mathbf{H}_f$) should be optimal feature data. We give the following definition.

Definition 1: Given a nonlinear piecewise continuous activation function g, we call $\{(\hat{\mathbf{a}}_f^j, \hat{b}_f^j)_{j=1}^L\}$ ($\hat{\mathbf{a}}_f^j \in \mathbf{R}^{d\times n}$) L optimal general hidden nodes and $\mathbf{H}_f^* = \sum_{j=1}^L g(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j)$ call optimal feature data if it satisfies

$$\|\mathbf{e}_L\| \le \min_{\mathbf{H}_f^L \in \mathbf{R}^{d\times M}} \left(\min_{\mathbf{a}_h \in \mathbf{R}^{m\times d}} \left\| \mathbf{y} - u_h^{-1} g(\mathbf{H}_f^L, \mathbf{a}_h, b_h) \right\| \right) \quad (4)$$

where $\mathbf{e}_L = \|\mathbf{y} - u_h^{-1} g(\mathbf{H}_f^{L*}, \hat{\mathbf{a}_h}, \hat{b_h})\|$ and sequence $\|\mathbf{e}_L\|$ is decreasing and bounded below by zero.

Remark 1: According to Definition 1 and Figs. 1 and 2, if we obtain the optimal projecting parameters in feature mapping layer $\{(\hat{\mathbf{a}}_f^j, \hat{b}_f^j)_{j=1}^L\}, \hat{\mathbf{a}}_f \in \mathbf{R}^{d\times n}$, the original n-dimension data points $\mathbf{x}$ are convert to d-dimension data points $\mathbf{H}_f = \sum_{j=1}^L g(\mathbf{x}_k, \hat{\mathbf{a}}_f^j, \hat{b}_f^j)$. And the d-dimension feature data $\mathbf{H}_f = \sum_{j=1}^L g(\mathbf{x}_k, \hat{\mathbf{a}}_f^j, \hat{b}_f^j)$ should satisfy (4). Thus the purpose of our method is to find optimal projecting parameters that make (4) true for all data points.

2) Learning Steps of Our Method: The learning steps are based on the use of the inverse of the activation function in order to obtain the optimal general parameters in feature mapping layer, and using L general nodes constitute a multiple ELMs learning systems. There are several learning steps in this proposed method.

1) *Step 1:* Given M arbitrary distinct training samples $\{(\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^M, \mathbf{x}_k \in \mathbf{R}^n, \mathbf{y} \in \mathbf{R}^m$, which is sampled from a continuous system, set $j = 1$, and then the initial general node of the feature mapping layer are generated randomly as

$$\mathbf{H}_f^j = g\left(\hat{\mathbf{a}}_f^j \cdot \mathbf{x} + \hat{b}_f^j\right), \quad \left(\hat{\mathbf{a}}_f^j\right)^T \cdot \hat{\mathbf{a}}_f^j = \mathbf{I}, \quad \left(\hat{b}_f^j\right)^T \cdot \hat{b}_f^j = 1 \quad (5)$$

where $\hat{\mathbf{a}}_f^j \in \mathbf{R}^{d\times n}$, $\hat{b}_f^j \in \mathbf{R}$ is the orthogonal random weight and bias of feature mapping layer. $\mathbf{H}_f^j$ is current feature data.

2) *Step 2:* Given a sigmoid or sine activation function g, for any continuous desired outputs $\mathbf{y}$, the parameters in ELM learning layer are obtained as

$$\begin{aligned} \hat{\mathbf{a}}_h &= g^{-1}(u_n(\mathbf{y})) \cdot \left(\mathbf{H}_f^j\right)^{-1}, \hat{\mathbf{a}}_h^j \in \mathbf{R}^{d\times m} \\ \hat{b}_h &= \sqrt{\mathrm{mse}\left(\hat{\mathbf{a}}_h^j \cdot \mathbf{H}_f^j - g^{-1}(u_n(\mathbf{y}))\right)}, \hat{b}_n^j \in \mathbf{R} \\ g^{-1}(\cdot) &\begin{cases} \arcsin(\cdot) & \text{if } g(\cdot) = \sin(\cdot) \\ -\log\left(\dfrac{1}{(\cdot)} - 1\right) & \text{if } g(\cdot) = 1/\left(1 + e^{-(\cdot)}\right) \end{cases} \end{aligned} \quad (6)$$

where $\mathbf{H}^{-1} = \mathbf{H}^T((C/\mathbf{I}) + \mathbf{H}\mathbf{H}^T)^{-1}$; C is a positive value; u_n is a normalized function $u_n(\mathbf{y}) : \mathbf{R} \to (0, 1]$; g^{-1} and u_n^{-1} represent their reverse function.

3) *Step 3:* Update the output error $\mathbf{e}_j$ as

$$\mathbf{e}_j = \mathbf{y} - u_n^{-1} g\left(\mathbf{H}_f^j, \hat{\mathbf{a}}_h, \hat{b}_h\right). \quad (7)$$

We can get error feedback data $\mathbf{P}_j = g^{-1}(u_n(\mathbf{e}_j)) \cdot (\hat{\mathbf{a}}_h)^{-1}$.

4) *Step 4:* Set $j = j + 1$, add a new general node $\hat{\mathbf{a}}_f^j, \hat{b}_f^j$ in the feature mapping layer by

$$\begin{aligned} \hat{\mathbf{a}}_f^j &= g^{-1}\left(u_j\left(\mathbf{P}_{j-1}\right)\right) \cdot \mathbf{x}^{-1}, \hat{\mathbf{a}}_f^j \in \mathbf{R}^{n\times d} \\ \hat{b}_f^j &= \sqrt{\mathrm{mse}\left(\hat{\mathbf{a}}_f^j \cdot \mathbf{x} - \mathbf{P}_{j-1}\right)}, \hat{b}_f^j \in \mathbf{R} \end{aligned} \quad (8)$$

and update the feature data $\mathbf{H}_f^j = \sum_{l=1}^j u_l^{-1} g(\mathbf{x}, \hat{\mathbf{a}}_f^l, \hat{b}_f^l)$.

5) *Step 5:* Repeat steps 2–4 $L - 1$ times.[1] The parameters $\{\hat{\mathbf{a}}_f^j, \hat{b}_f^j\}_{j=1}^L$ are optimal projecting parameters and the feature data $\mathbf{H}_f^L = \sum_{j=1}^L u_j^{-1} g(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j) = \mathbf{H}_f^*$ are the optimal feature data.

The above five steps constitute the proposed method, as shown in Algorithm 1. This algorithm ensures the feature data $\mathbf{H}_f(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j)$ are the optimal feature data. We give the proof of this method in Section III-C.

Algorithm 1 Proposed Method

Initialization: Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^M \subset \mathbf{R}^n \times \mathbf{R}^m$, a sine or sigmoid function $h(\cdot)$, a continuous target function f, maximum loop number L, $\mathbf{e}_1 = \mathbf{y}$, and $j = 1$.

Learning step:

Step 1) Generate the initial general node $\hat{\mathbf{a}}_f^j$ and $\hat{b}_f^j$ according to equation (5).

while $j < L$ **do**

Step 2) Calculate $\hat{\mathbf{a}}_h$ and $\hat{b}_h$ according to equation (6).

Step 3) Update outputs error $\mathbf{e}_j$ and error feedback data $\mathbf{P}_j$ according to equation (7).

step 4) Set $j = j + 1$, add a new general node $\hat{\mathbf{a}}_f^j, \hat{b}_f^j$ according to equation (8), and update $\mathbf{H}_f^j$.

end while

Obtain optimal feature data $\mathbf{H}_f^L$.

C. Proof of the Proposed Method

In this section, we prove these results. Given M arbitrary distinct samples $\{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^M, \mathbf{x}_k \in \mathbf{R}^n$, and $\mathbf{y} \in \mathbf{R}^m$, in Lemma 2 and Theorem 1, we prove that if parameters in ELM learning layer $\hat{\mathbf{a}_h}, \hat{b_h}$ are generated by (6), these parameters satisfy

$$\begin{aligned} \hat{\mathbf{a}_h} &= \operatorname*{argmin}_{\mathbf{a}_h \in \mathbf{R}^{m\times d}} \left\| u^{-1} \cdot g(\mathbf{x}, \mathbf{a}_h, b_h) - \mathbf{y} \right\| \\ \left\| g(\mathbf{x}, \hat{\mathbf{a}_h}, \hat{b_h}) - \mathbf{y} \right\| &\le \min_{\mathbf{a}_h \in \mathbf{R}^{m\times n}} \left\| u^{-1}(g(\mathbf{a}_h \cdot \mathbf{x})) - \mathbf{y} \right\|. \end{aligned} \quad (9)$$

Based on Lemmas 2 and Theorem 1, in Theorem 2 we prove that $\hat{\mathbf{a}}_f^j, \hat{b}_f^j$ are optimal projecting parameters and

[1] When repeating steps 2–4 once, a new general node is added to the existing network.

$\mathbf{H}_f^L = \sum_{j=1}^{L} g(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j)$ are the optimal feature data. In Theorem 3, we further prove that if given optimal feature data $\mathbf{H}_f^L$, we have $\lim_{j\to+\infty} \|f_{j-1} - \beta_j u_j^{-1} g(\mathbf{H}_f^L, \hat{\mathbf{a}}_j, \hat{b}_j)\| = 0$.

Lemma 2 [39]: Given a bounded nonconstant piecewise continuous activation function g, we have

$$\lim_{(\boldsymbol{\alpha},\mathfrak{b})\to\left(\hat{\boldsymbol{\alpha}}_h,\hat{\mathfrak{b}}\right)} \left\| g(\mathbf{x},\boldsymbol{\alpha},\mathfrak{b}) - g\left(\mathbf{x},\hat{\boldsymbol{\alpha}},\hat{\mathfrak{b}}\right)\right\| = 0. \tag{10}$$

Remark 2: Lemma 2 shows that SLFN training problem can be considered as finding optimal hidden parameters which satisfy $g(\hat{\boldsymbol{\alpha}}_1, \hat{\mathfrak{b}}_1) + \cdots + g(\hat{\boldsymbol{\alpha}}_L, \hat{\mathfrak{b}}_L) \longrightarrow \mathbf{y}$. Thus training an SLFN is equivalent to finding a least-square general input weight $\hat{\mathbf{a}_h}$ of the system $g(\mathbf{a}_h \cdot \mathbf{x}) = \mathbf{y}$. If activation function g is invertible, input weights matrix can be obtained by pulling back the residual error to the hidden layer. For example, for M arbitrary distinct samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^M$, $\mathbf{x}_i \in \mathbf{R}^n$, and $\mathbf{t}_i \in \mathbf{R}^m$, if the activation function is a sine function, the output of the hidden layer matrix is $y = \sin(\mathbf{a}_h \cdot \mathbf{x})$. Thus, similar to [21], hidden nodes can be obtained by finding a least-square general input weight $\hat{\mathbf{a}_h}$ of the linear system $\mathbf{a}_h \cdot \mathbf{x} = \arcsin(\mathbf{y})$, $\mathbf{y} \in (0, 1]$, such that

$$\left\|\sin\left(\hat{\mathbf{a}}_h, \mathbf{x}\right) - \mathbf{y}\right\| = \min_{\mathbf{a}_h\in\mathbf{R}^{m\times n}} \|\sin(\mathbf{a}_h \cdot \mathbf{x}) - \mathbf{y}\|. \tag{11}$$

According to [21] and [40], the smallest norm least-squares solution of the linear system is $\hat{\mathbf{a}_n} = \arcsin(\mathbf{y}) \cdot \mathbf{x}^{-1}$, where $\mathbf{x}^{-1}$ is the Moore–Penrose generalized inverse of matrix $\mathbf{x}$. We give the following theorem.

Theorem 1: Given M arbitrary distinct samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^M$, $\mathbf{x}_i \in \mathbf{R}^n$, and $\mathbf{y}_i \in \mathbf{R}^m$ and a sigmoid or sine activation function g, for any continuous desired outputs $\mathbf{y}$, we have

$$\hat{\mathbf{a}_h} = \underset{\mathbf{a}_h\in\mathbf{R}^{m\times n}}{\operatorname{argmin}} \left\| u^{-1} g(\mathbf{x}, \mathbf{a}_h) - \mathbf{y}\right\|$$
$$\left\| g(\mathbf{x}, \hat{\mathbf{a}_h}, \hat{b}_h) - \mathbf{y}\right\| \le \min_{\mathbf{a}_h\in\mathbf{R}^{m\times n}} \left\| u^{-1}(g(\mathbf{a}_h \cdot \mathbf{x})) - \mathbf{y}\right\| \tag{12}$$

if the parameters are obtained by

$$\begin{aligned}
&\hat{\mathbf{a}_h} = g^{-1}(u(\mathbf{y})) \cdot \mathbf{x}^{-1}, \quad \hat{\mathbf{a}_h} \in \mathbf{R}^{m\times n}\\
&\hat{b_h} = \sqrt{\operatorname{mse}\left(\hat{\mathbf{a}_h} \cdot \mathbf{x} - g^{-1}(u(\mathbf{y}))\right)}, \hat{b_h} \in \mathbf{R}\\
&g^{-1}(\cdot) \begin{cases} \arcsin(\cdot) & \text{if } g(\cdot) = \sin(\cdot)\\ -\log\left(\dfrac{1}{(\cdot)} - 1\right) & \text{if } g(\cdot) = 1/\left(1 + e^{-(\cdot)}\right)\end{cases}
\end{aligned} \tag{13}$$

where $\mathbf{x}^{-1} = \mathbf{x}^T((C/\mathbf{I}) + \mathbf{x}\mathbf{x}^T)^{-1}$; u is a normalized function $u : \mathbf{R} \to (0, 1]$; and g^{-1} and u^{-1} represent their reverse functions.

Proof: Let $\boldsymbol{\lambda} = \mathbf{a}_h \cdot \mathbf{x}$, and $\boldsymbol{\lambda}$ satisfy $g(\boldsymbol{\lambda}) = \mathbf{y}$. In order to let $\boldsymbol{\lambda} \in \mathbf{R}$, we give normalized function u: $u(\cdot) \in (0, 1]$. Thus, for a sine hidden node we have

$$\boldsymbol{\lambda} = g^{-1}(u(\mathbf{y})) = \arcsin(u(\mathbf{y})). \tag{14}$$

For a sigmoid hidden node, we have

$$\boldsymbol{\lambda} = g^{-1}(u(\mathbf{y})) = -\log\left(\frac{1}{u(\mathbf{y})} - 1\right). \tag{15}$$

Because we have $\boldsymbol{\lambda} = \mathbf{a}_h \cdot \mathbf{x}$, for a sine activation function, we can find the solution $\hat{\mathbf{a}_h}$ of the linear system

$$\hat{\mathbf{a}_h} = g^{-1}(u(\mathbf{y})) \cdot \mathbf{x}^{-1} = \arcsin(u(\mathbf{y})) \cdot \mathbf{x}^{-1}. \tag{16}$$

For a sigmoid activation function, we have

$$\hat{\mathbf{a}_h} = g^{-1}(u(\mathbf{y})) \cdot \mathbf{x}^{-1} = -\log\left(\frac{1}{u(\mathbf{y})} - 1\right) \cdot \mathbf{x}^{-1} \tag{17}$$

where $\mathbf{x}^{-1}$ is the Moore–Penrose generalized inverse of the given set of training examples [41]. Similar to [21], we have the following.

1) $\hat{\mathbf{a}_h} = g^{-1}(u(\mathbf{y}))\cdot\mathbf{x}^{-1}$ is one of the least-squares solutions of a general linear system $\mathbf{a}_h \cdot \mathbf{x} = \boldsymbol{\lambda}$, meaning that the smallest error can be reached by the solution

$$\left\|\hat{\mathbf{a}_h} \cdot \mathbf{x} - \boldsymbol{\lambda}_n\right\| = \min_{\mathbf{a}_h\in\mathbf{R}^{m\times n}} \left\|\mathbf{a}_h \cdot \mathbf{x} - g^{-1}(u(\mathbf{y}))\right\|. \tag{18}$$

2) The special solution $\hat{\mathbf{a}_h} = g^{-1}(u(\mathbf{y})) \cdot \mathbf{x}^{-1}$ has the smallest norm among all the least-squares solutions of $\mathbf{a}_h \cdot \mathbf{x} = \boldsymbol{\lambda}$. Although the smallest error can be reached by (16) and (17), we can further reduce its error by adding bias b_n

$$\hat{b}_h = \sqrt{\operatorname{mse}\left(\hat{\mathbf{a}_h} \cdot \mathbf{x} - h^{-1}(u(\mathbf{y}))\right)}. \tag{19}$$

According to (18) and Lemma 2, we have

$$\begin{aligned}
&\min_{\mathbf{a}_h\in\mathbf{R}^{n\times m}} \left\|u^{-1}(g(\mathbf{a}_h \cdot \mathbf{x})) - u^{-1}(g(\boldsymbol{\lambda}))\right\|\\
&= \left\|u^{-1}\left(g\left(\hat{\mathbf{a}_h} \cdot \mathbf{x}\right)\right) - u^{-1}(g(\boldsymbol{\lambda}))\right\| \ge \left\|u^{-1}\left(g\left(\hat{\mathbf{a}_h} \cdot \mathbf{x} + \hat{b}_h\right)\right) - \mathbf{y}\right\|.
\end{aligned} \tag{20}$$

Based on (18) and (20), we have $\hat{\mathbf{a}_h} = \operatorname{argmin}_{\mathbf{a}_h\in\mathbf{R}^{m\times n}} \|g(\mathbf{x}, \mathbf{a}_h) - \mathbf{y}\|$ and $\|g(\mathbf{x}, \hat{\mathbf{a}_h}, \hat{b}_h) - \mathbf{y}\| \le \min_{\mathbf{a}_h\in\mathbf{R}^{m\times n}} \|u^{-1}(g(\mathbf{a}_h \cdot \mathbf{x})) - \mathbf{y}\|$. ■

Now, based on Lemma 2 and Theorem 1, we give Theorem 2.

Theorem 2: Given M arbitrary distinct samples $(\mathbf{x}, \mathbf{y})$, $\mathbf{x} \in \mathbf{R}^{n\times M}$, and $\mathbf{y} \in \mathbf{R}^{m\times M}$, a sigmoid or sine activation function g, and the initial orthogonal random weights and bias $\hat{\mathbf{a}}_f^1, \hat{b}_f^1$, for any continuous desired output $\mathbf{y}$ we can get optimal feature data $\mathbf{H}_f^{L*}(\mathbf{x}, (\hat{\mathbf{a}}_f^1 \cdots \hat{\mathbf{a}}_f^L), (\hat{b}_f^1 \cdots \hat{b}_f^L)) = \sum_{j=1}^L u_j^{-1} g(\mathbf{x} \cdot \hat{\mathbf{a}}_f^j + \hat{b}_f^j)$, which satisfy

$$\|\mathbf{e}_L\| \le \min_{\mathbf{a}_f^j\in\mathbf{R}^{n\times d}} \left(\min_{\mathbf{a}_h\in\mathbf{R}^{m\times d}} \left\|\mathbf{y} - u_n^{-1} g\left(\mathbf{H}_f^L, \mathbf{a}_h, b_h\right)\right\|\right) \tag{21}$$

and $\|\mathbf{e}_L\|$ is decreasing and bounded below by zero if these parameters are obtained by

$$\begin{aligned}
&\mathbf{H}_f^j = \sum_{i=1}^{j} u_i^{-1} g\left(\mathbf{x}, \hat{\mathbf{a}}_f^i, \hat{b}_f^i\right),\\
&\hat{\mathbf{a}}_h = g^{-1}(u_n(\mathbf{y})) \cdot \left(\mathbf{H}_f^j\right)^{-1}, \hat{\mathbf{a}}_h \in \mathbf{R}^{m\times d}\\
&\hat{b}_h = \sqrt{\operatorname{mse}\left(\hat{\mathbf{a}}_h \cdot \mathbf{H}_f^j - g^{-1}(u_n(\mathbf{y}))\right)}, \hat{b}_n \in \mathbf{R}\\
&g^{-1}(\cdot) \begin{cases} \arcsin(\cdot) & \text{if } g(\cdot) = \sin(\cdot)\\ -\log\left(\dfrac{1}{(\cdot)} - 1\right) & \text{if } g(\cdot) = 1/\left(1 + e^{-(\cdot)}\right)\end{cases}
\end{aligned} \tag{22}$$

$$\begin{aligned}\mathbf{e}_j &= \mathbf{y} - u_n^{-1} g\left(\mathbf{H}_f^j, \hat{\mathbf{a}}_h, \hat{b}_h\right), \quad \mathbf{P}_j = g^{-1}\left(u_n\left(\mathbf{e}_j\right)\right) \cdot \left(\hat{\mathbf{a}}_h\right)^{-1} \\ \hat{\mathbf{a}}_f^j &= g^{-1}\left(u_j\left(\mathbf{P}_{j-1}\right)\right) \cdot \mathbf{x}^{-1}, \hat{\mathbf{a}}_f^j \in \mathbf{R}^{n\times d} \\ \hat{b}_f^j &= \sqrt{\text{mse}\left(\hat{\mathbf{a}}_f^j \cdot \mathbf{x} - \mathbf{P}_{j-1}\right)}, \hat{b}_f^j \in \mathbf{R}. \end{aligned} \tag{23}$$

Proof: According to Theorem 1, the validity of (21) is obvious. Here, we just prove that error $\|\mathbf{e}\|$ is decreasing and bounded below by zero. First, let $\Delta = \|\mathbf{e}_{j-1}\|^2 - \|\mathbf{y} - u_n^{-1} g(\mathbf{H}_f^j, \hat{\mathbf{a}}_h, \hat{b}_h)\|^2$. We have

$$\begin{aligned}\Delta &= \left\|\mathbf{e}_{j-1}\right\|^2 - \left\|\mathbf{y} - u_n^{-1} g\left(\mathbf{H}_f^j, \hat{\mathbf{a}}_h, \hat{b}_h\right)\right\|^2 \\ &= \left\|\mathbf{e}_{j-1}\right\|^2 - \left\|\mathbf{y} - u_n^{-1} g\left(\left(\left(\sum_{i=1}^{j-1} u_i^{-1} g\left(\mathbf{x}, \hat{\mathbf{a}}_f^i, \hat{b}_f^i\right)\right), \right.\right.\right. \\ &\qquad \left.\left.\left. + u_i^{-1} g\left(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j\right)\right), \hat{\mathbf{a}}_h, \hat{b}_h\right)\right\|^2. \end{aligned} \tag{24}$$

Let $\hat{\mathbf{T}}^j = u_n^{-1} g(u_j^{-1} g(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j), \hat{\mathbf{a}}_h, \hat{b}_h)$. Because activation function is sigmoid or sine function, (24) can be rewritten as

$$\begin{aligned}\Delta &\geq \left\|\mathbf{e}_{j-1}\right\|^2 - \left\|\mathbf{y} - u_n^{-1} g\left(\left(\left(\sum_{i=1}^{j-1} u_j^{-1} g\left(\mathbf{x}, \hat{\mathbf{a}}_f^i, \hat{b}_f^i\right)\right), \hat{\mathbf{a}}_h, \hat{b}_h\right) - \hat{\mathbf{T}}^j\right\|^2 \\ &= \left\|\mathbf{e}_{j-1}\right\|^2 - \left\|\mathbf{e}_{j-1} - \hat{\mathbf{T}}^j\right\|^2 = \left\|\hat{\mathbf{T}}^j\right\|^2 \left(\frac{2 < \mathbf{e}_{j-1}, \left\|\hat{\mathbf{T}}^j\right\| >}{\left\|\hat{\mathbf{T}}^j\right\|^2} - 1\right). \end{aligned} \tag{25}$$

We set

$$\hat{\mathbf{T}}^j = u_n^{-1} g\left(u_j^{-1} g\left(\mathbf{x}, \hat{\mathbf{a}}_f^j, \hat{b}_f^j\right), \hat{\mathbf{a}}_h, \hat{b}_h\right) = \mathbf{e}_{j-1} \pm \sigma. \tag{26}$$

According Theorem 1 and (7), we can get

$$\begin{aligned}\left\|\hat{\mathbf{T}}^j - \mathbf{e}_{j-1}\right\| &\leq \min_{\mathbf{a}_f^j \in \mathbf{R}^{d\times n}} \left\|u_n^{-1} g\left(u_j^{-1} g\left(\mathbf{x}, \mathbf{a}_f^j, b_f^j\right), \hat{\mathbf{a}}_h, \hat{b}_h\right) - \mathbf{e}_{j-1}\right\| \\ \|\sigma\| &\leq \left\|\hat{\mathbf{T}}^j\right\|. \end{aligned} \tag{27}$$

Thus $\Delta \geq 0$ is still valid for

$$\begin{aligned}\Delta &\geq \left\|\hat{\mathbf{T}}^j\right\|^2 \left(\frac{2 < \hat{\mathbf{T}}^j \pm \sigma, \hat{\mathbf{T}}^j >}{\left\|\hat{\mathbf{T}}^j\right\|^2} - 1\right) \\ &= \left\|\hat{\mathbf{T}}^j\right\|^2 \left(\frac{2\left(\left\|\hat{\mathbf{T}}^j\right\|^2 \pm < \hat{\mathbf{T}}^j, \sigma\| >\right)}{\left\|\hat{\mathbf{T}}^j\right\|^2} - 1\right) \\ &= \left\|\hat{\mathbf{T}}^j\right\|^2 \left(1 \pm \frac{\left\|\sigma \cdot \left(\hat{\mathbf{T}}^j\right)^T\right\|}{\left\|\hat{\mathbf{T}}^j\right\|^2}\right) \geq \left\|\hat{\mathbf{T}}^j\right\|^2 \left(1 \pm \frac{\|\sigma\|}{\left\|\hat{\mathbf{T}}^j\right\|}\right) \geq 0. \end{aligned} \tag{28}$$

Based on (28), we have $\|\mathbf{e}_{j-1}\| \geq \|\mathbf{e}_j\|$ and $\|\mathbf{e}\|$ is decreasing and bounded below by zero. ■

Based on Theorem 2, we give Theorem 3.

Theorem 3: Given M arbitrary distinct samples $(\mathbf{x}, \mathbf{y}), \mathbf{x} \in \mathbf{R}^{n\times M}$, and $\mathbf{y} \in \mathbf{R}^{m\times M}$, a sigmoid or sine activation function g, optimal feature data $\mathbf{H}_f^L$ obtained by Algorithm 1, then we have $\lim_{j\to+\infty} \|\mathbf{y} - \beta_1 \cdot u_1^{-1} g(\mathbf{H}_f^L, \mathbf{a}_1, b_1) - \cdots - \beta_j \cdot u_j^{-1} g(\mathbf{H}_f^L, \mathbf{a}_j, b_j)\| = 0$ holds with probability one if

$$\begin{aligned}\mathbf{a}_j &= g^{-1}(u(\mathbf{y})) \cdot \left(\mathbf{H}_f^L\right)^{-1}, \quad \hat{\mathbf{a}}_j \in \mathbf{R}^{m\times n} \\ b_j &= \sqrt{\text{mse}\left(\hat{\mathbf{a}}_j \cdot \left(\mathbf{H}_f^L\right) - g^{-1}(u(\mathbf{y}))\right)}, \hat{b}_j \in \mathbf{R} \\ \beta_j &= \frac{\left\langle \mathbf{e}_{j-1}, g\left(\mathbf{H}_f^L, \mathbf{a}_j, b_j\right)\right\rangle}{\left\|g\left(\mathbf{H}_f^L, \mathbf{a}_j, b_j\right)\right\|^2}, \beta_j \in \mathbf{R}. \end{aligned} \tag{29}$$

Proof: We first prove that the sequence $\|\mathbf{e}_j^L\|$ is decreasing and bounded below by zero. Then we further prove $\lim_{j\to+\infty} \|\mathbf{e}_j^L\| = 0$.

1) According to Theorem 1 and Lemma 1, the network output error satisfies

$$\begin{aligned}\left\|\mathbf{e}_j^L\right\| &= \left\|\mathbf{y} - \beta_1 \cdot u_1^{-1} g\left(\mathbf{H}_f^L, \mathbf{a}_1, b_1\right) \right. \\ &\qquad \left. - \cdots - \beta_j \cdot u_j^{-1} g\left(\mathbf{H}_f^L, \mathbf{a}_j, b_j\right)\right\| \\ &\leq \left\|\mathbf{y} - u_1^{-1} g\left(\mathbf{H}_f^L, \mathbf{a}_1, b_1\right)\right\| = \left\|\mathbf{e}_1^L\right\|. \end{aligned} \tag{30}$$

According to Theorem 2, we have

$$\left\|\mathbf{e}_j^L\right\| \leq \left\|\mathbf{e}_j^{L-1}\right\| \leq \cdots \leq \left\|\mathbf{e}_j^1\right\|. \tag{31}$$

Thus we have $\|\mathbf{e}_j^L\| \leq \|\mathbf{e}_j^{L-1}\| \leq \cdots \leq \|\mathbf{e}_j^1\| \leq \cdots \| \leq \|\mathbf{e}_1^1\|$ and $\|\mathbf{e}_j^L\|$ is decreasing and bounded below by zero.

2) According to Lemma 1, when all hidden nodes randomly generated based on any continuous sampling distribution, $\lim_{n\to\infty} \|f - (f_{n-1} + \beta_n \cdot g(\mathbf{x}, \boldsymbol{\alpha}_n^r, \flat_n^r))\| = 0$ holds with probability one if $\beta_n = (\langle \mathbf{e}_{n-1}, g(\mathbf{H}_j^L, \boldsymbol{\alpha}_n^r, \flat_n^r)\rangle / \|g(\mathbf{H}_j^L, \boldsymbol{\alpha}_n^r, \flat_n^r)\|^2)$. In addition, ELM theories have shown that almost any nonlinear piecewise continuous random hidden node can be used in ELM, and the resultant networks have universal approximation capabilities.[2] According to the definition of general hidden neurons, we set $(\mathbf{a}, b) = (\boldsymbol{\alpha}_1^r, \ldots, \boldsymbol{\alpha}_m^r, \flat_1^r, \ldots, \flat_m^r)$. Thus we have

$$g\left(\mathbf{H}_f^L, \mathbf{a}_j^r, b_j^r\right) \equiv \sum_{i=1}^{m} g\left(\mathbf{H}_f^L, \boldsymbol{\alpha}_i^r, \flat_i^r\right). \tag{32}$$

We can get

$$\begin{aligned}\lim_{j\to+\infty} &\left\|\mathbf{y} - \beta_1 \cdot u_1^{-1} g\left(\mathbf{H}_f^L, \mathbf{a}_1, b_1\right) \right. \\ &\left. - \cdots - \beta_j \cdot u_j^{-1} g\left(\mathbf{H}_f^L, \mathbf{a}_j, b_j\right)\right\| \\ &= \lim_{n\to\infty} \left\|f - \left(f_{n-1} + \beta_j u_j^{-1} g\left(\mathbf{H}_f^L, \mathbf{a}_j^r, b_j^r\right)\right)\right\| = 0. \end{aligned} \tag{33}$$

■

[2]Feng *et al.* [30] mentioned: the generalization performance of ELM is not sensitive to the dimensionality L of the feature space (the number of the hidden nodes) as long as L is set large enough (e.g., $L > 1000$ for all the real-world cases tested in our simulation).

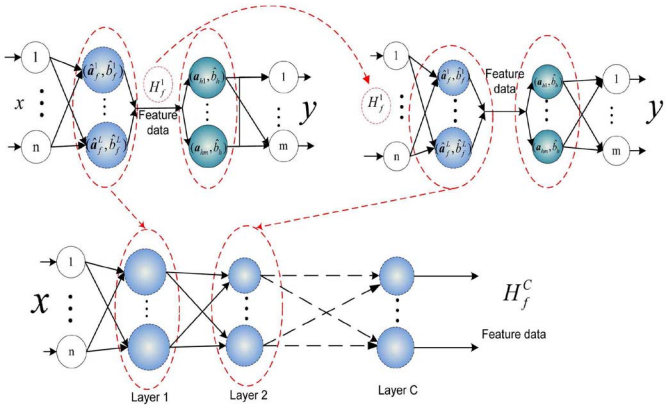


Fig. 3. Proposed method with c-layer structure. In the multilayer network structure, each layer is generated by Algorithm 1 and functions as a separated feature extractor. The input weights $[(\hat{\mathbf{a}}_f^1, \hat{b}_f^1), \ldots, (\hat{\mathbf{a}}_f^L, \hat{b}_f^L)]$, with respect to first layer in two-layer ELM, is a hidden layer in multilayer network.

D. Proposed Method With Multinetwork Structures

The proposed method can be used for multinetwork structures (see Fig. 3). The proposed method with multilayer structure generally provides a better general performance than single-layer structure.

In this multinetwork structure, the input data first should be transformed into several ELM-based layers, and the input raw data will be converted into d-dimension features. Mathematically, given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^M \subset \mathbf{R}^n \times \mathbf{R}^m$, the output of each hidden layer can be represented as $\mathbf{H}_f^c = \sum_{i=1}^L g(\mathbf{H}_f^{c-1} \cdot \hat{\mathbf{a}}_f^i + \hat{b}_f^i)$, where $\mathbf{H}_f^c$ is the output of the cth layer, $\mathbf{H}_f^{c-1}$ is the output of $c-1$th layer, and g denotes the activation function of the hidden layers.

IV. Experimental Verification

In this section, aimed at examining the performance of our proposed learning method, we test the proposed method on 16 classification problems. The experiments are conducted in MATLAB 2013a with 32 GB of memory and an I7-4770 (3.4G) processor. In Section IV-B, we compare the generalization performance of the proposed method with nine transform-based supervised learning methods for dimensionality reduction (Section IV-B1). In Section IV-B2, we compare the performance and efficiency of the proposed method on several image datasets Caltech-101, Caltech-256, Yale Face, Scene15, Mnist, Flower102, Olivetti face, Indoor67, and PFID61 (see Table II) with many "complex" methods [42], [43] for expanded feature learning. Different from above mentioned transform-based dimension reduction methods, these "complex" methods always combine preprocessing methods, multifeature fusion, image segmentation technologies, or multiclassifiers. The nine transform-based supervised feature extraction methods are as follows.

1) The proposed two-layer representation learning (TLRL).
2) Our proposed multilayer representation learning (MLRL).
3) NCA.
4) NPE [5].
5) LSDA [2].
6) IsoP [3].
7) Supervised LPP [7].
8) LDA.
9) MMP [4].

TABLE II
Specification of 16 Benchmark Datasets

Datasets	#features	#Train	# Test
Acoustic	51	40000	58000
Codrna	8	40000	19000
Colon	62	1500	500
Connect-4	126	50000	17557
Covtype.binary	54	300000	280000
DNA	180	1046	1186
Duke	7129	29	15
Gisette	5000	6000	1000
Hill Valley	101	606	606
IJCNN	23	40000	101000
Leukemia	7129	38	34
Movement	91	300	60
Mushrooms	112	4000	4122
Protein	357	17766	6621
Sonar	60	150	58
USPS	256	7291	2007

TABLE III
Specification of Nine Image Datasets

Datasets	Feature Type	#Feature	#Image	# Category
Caltech-101	SIFT	21504	9144	102
Caltech-256	SIFT	21504	30607	257
Car	Raw Image	530432	126	_
Flower102	SIFT	21504	8189	102
Indoor67	SIFT	21504	15619	67
Mnist	Raw Image	780	70000	10
Olivetti faces	Raw Image	4096	400	40
PFID61	SIFT	21504	1098	61
Scene15	SIFT	43008	4485	15
Yale Face	Raw Image	4096	165	15

In Section IV-C, we carry out a comparative analysis of our proposed method with six other unsupervised methods. Section IV-C1 provides the results for image reconstruction and dimensionality reduction, while Section IV-C2 provides the results for unsupervised sparse representation. The six unsupervised feature learning methods are as follows.

1) DBN [15].
2) SAE [17].
3) LGE [6].
4) Orthogonal LGE (OLGE) [44].
5) Unsupervised LPP [7].
6) ML-ELM [18].

The codes used for DBN,[3] SAE,[3] LPP,[4] OLGE,[4] MMP,[4] and IsoP[4] are downloaded from the Internet.

A. Datasets and Experiment Environment Settings

To verify the performance of different algorithms in our experiments, tested datasets are selected from a wide variety of available datasets including small size, medium dimensions, large size, and/or high dimensions. Here, we use 15 classification problems, and nine image datasets (see Tables II and III).

[3]http://www.mathworks.com/matlabcentral/fileexchange/38310-deep-learning-toolbox

[4]http://www.cad.zju.edu.cn/home/dengcai/Data/DimensionReduction.html

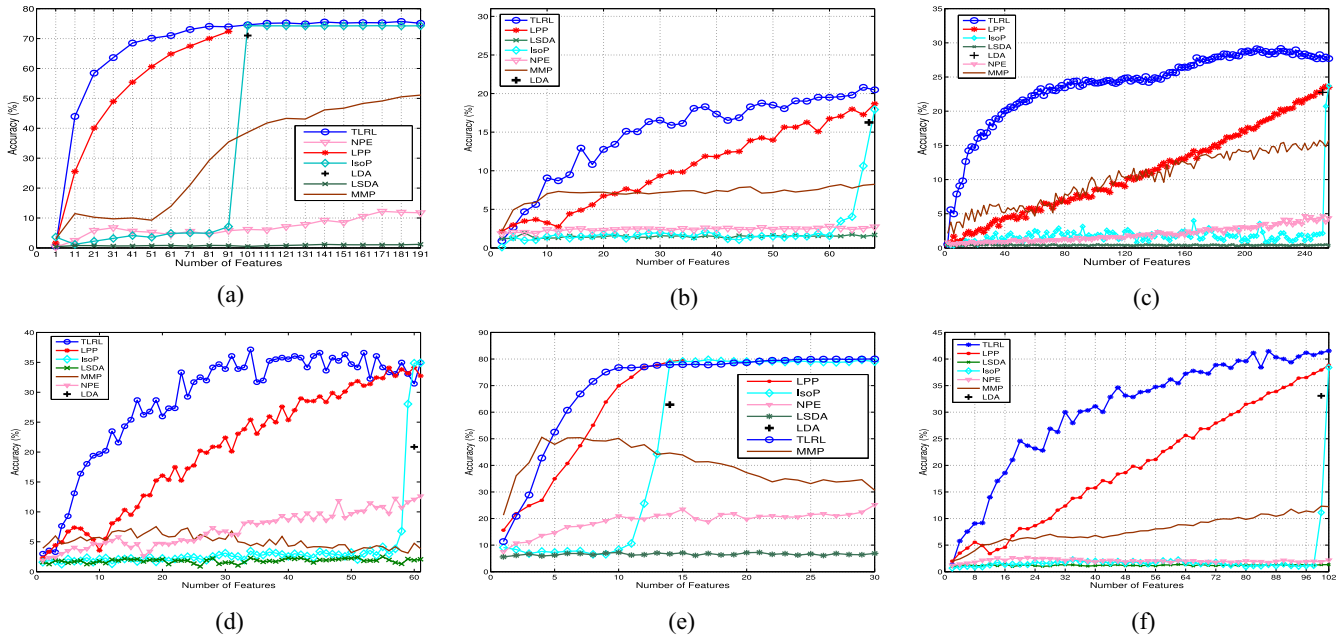


Fig. 4. Average testing accuracy by using LPP, IsoP, LSDA, LDA, MMP, NPE and the proposed method on Caltech101, Caltech256, Indoor67, PFID61, Scene15, Flower102, Mushroom, Movement, and Olive face, where the x- and y-axis show the number of features and average testing accuracy, respectively. Results on (a) Caltech101, (b) Indoor67, (c) Caltech256, (d) PFID61, (e) Scene15, and (f) Flower102.

Most of the datasets are taken from the UCI Machine Learning Repository[5] and LIBSVM DATASETS.[6] The other publicly and commonly used image datasets are described as follows. All databases are preprocessed in the same way (held-out method). Table II shows the training and testing data of the corresponding datasets. Table III shows detailed information about image datasets. For these image datasets, we perform all processing in grayscale even when color images are available. The average results in the tables are obtained over 30 trials, but the results in the figures are achieved over ten trails. Our experimental results for both Caltech-101 and Caltech-256 are generated by 15 or 30 images per category for training; for the Indoor67, Flower102, PFID61, Scene15, Yale faces, and Olivetti faces datasets, we use 80, 30, 12, 100, 8, and eight images per category for training. All the remaining images are used for testing. For our proposed method, parameter C is selected from $C \in \{2^{-4}, \ldots, 2^{8}\}$. For DBN, we set epoch at 20 and select the parameter batch size from $[10, \#(\text{Train})/10, \#(\text{Train})/9, \ldots, \#(\text{Train})]$, and the parameter momentum γ from $[0.01, 0.02, \ldots, 0.1]$. The parameter α equals $1 - \gamma$. For SAE, the learning rate equals 1; the parameter batch size is selected from $[10, \#(\text{Train})/20, \#(\text{Train})/18, \ldots, \#(\text{Train})/2]$; and the parameter masked fraction rate δ is selected from $[0.45, 0.46, \ldots, 0.55]$.

All the testing accuracy is obtained following the same steps: first, we use these methods to obtain data features, and then an SLFN classifier is used to generate testing accuracy. For DBN and SAE, we first reduce or increase the dimensions of the testing datasets, and then BP and Fuzzy NN networks are used to generate testing accuracy, respectively. But in Table VI, we use DBN/SAE to reduce the dimensions of the testing datasets, and then a 1000-hidden-nodes ELM classifier is used to generate testing accuracy. For the other methods, we first reduce the dimensions of the dataset, and then a 1000-hidden-nodes ELM network is used to obtain testing accuracy. The experimental results of the proposed method and these competing methods are given in Tables IV–VII. In these tables, the apparent best results are shown in boldface.

[5]http://archive.ics.uci.edu/ml/datasets.html

[6]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

B. Supervised Feature Learning

1) Supervised Dimension Reduction: To indicate the advantage of the proposed method for supervised dimension reduction performance, tests have been conducted of the accuracy of the proposed method compared to other supervised dimension reduction methods. Table IV and Fig. 4 display the performance comparison of NCA, NPE, IsoP, LSDA, LPP, LDA, and the proposed method. For these compared methods, we directly reduce the data dimensions from the original dimensions to the target dimensions. The target dimensions should meet the following two rules: 1) the target dimension should be equal to or less than the category numbers. For example, the category numbers of the USPS dataset is ten, so the target dimension for the USPS dataset equals ten and 2) if the category numbers are very high, the target dimension should be much less than the dataset category numbers. As we can see, our proposed algorithm consistently outperforms most of the compared feature extraction algorithms on the 16 classification problems and ten image datasets. Consider Duke (small number of samples with large input dimensions), Caltech256 (large number of samples with large input dimensions), and Flower102 (medium number of samples with large input dimensions).

TABLE IV
PERFORMANCE COMPARISON OF SUPERVISED DIMENSION REDUCTION (MEAN: AVERAGE TESTING ACCURACY, STD: STANDARD DEVIATION)

DATASETS	#FEATURES	NCA		NPE		IsoP		LSDA		LPP		LDA		PROPOSED TLRL	
		MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
Acoustic	51 ⟶ 1	_a	-	_b	-	_b	-	_b	-	_b	-	69.92%	0.0076	**70.95**%	0.0017
Colon	62 ⟶ 1	_a	-	60.74%	0.0289	84.08%	0.0207	59.47%	0.0077	_b	-	83.08%	0.0476	**84.72**%	0.0309
Connect-4	126 ⟶ 2	_a	-	_a	-	_b	-	_b	-	_b	-	75.54%	0.0016	**75.87**%	0.0027
Covtype.binary	54 ⟶ 2	34.13%	0.0217	_a	-	_b	-	_b	-	_b	-	-	-	**75.91**%	0.0005
DNA	180 ⟶ 2	78.70%	0.1041	_a	-	93.68%	0.0005	57.91%	0.0216	92.95%	0.0170	93.04%	0.0170	**94.10**%	0.0113
Duke	7129 ⟶ 2	84.83%	0.0875	75.82%	0.1838	84.91%	0.0708	_b	-	90.03%	0.0818	-	-	**90.18**%	0.0273
Gisettle	5000 ⟶ 2	_a	-	52.60%	0.0017	83.88%	0.0009	50.95%	0.0006	87.90%	0.0780	-	-	**93.10**%	0.0057
Hill Valley	101⟶ 2	_a	-	57.73%	0.0737	67.58%	0.0261	64.03%	0.0581	67.81%	0.0266	-	-	**69.15**%	0.0594
IJCNN	23 ⟶ 1	_a	-	_b	-	_b	-	_b	-	_b	-	91.30%	0.0009	91.31%	0.0007
Leukemia	7129 ⟶ 2	72.38%	0.0072	58.82%	<0.0001	82.35%	<0.0001	75.41%	0.0204	**96.01**%	0.0206	-	-	93.21%	0.0745
Movement	91 ⟶ 15	64.13%	0.0174	68.78%	0.0010	63.70%	0.0067	62.18%	0.0275	62.27%	0.0030	-	-	**88.17**%	0.0074
Mushroom	112 ⟶ 2	_a	-	86.44%	0.0031	90.28%	0.0003	79.60%	0.0212	94.66%	0.0231	-	-	**99.93**%	0.0004
Protein	357⟶ 3	_a	-	46.88%	<0.0001	68.60%	0.0011	_b	-	52.88%	0.0007	-	-	**68.73**%	0.0057
Sonar	60⟶ 2	68.75%	0.0671	62.07%	0.0857	69.08%	0.0741	68.79%	0.0626	69.25%	0.0566	-	-	**72.53**%	0.0470
USPS	256 ⟶ 10	51.66%	0.0108	90.60%	0.0016	90.03%	0.0007	91.23%	0.0018	91.27%	0.0012	-	-	**91.70**%	0.0155
Caltech101	21504⟶ 31	_a	-	6.77%	0.0086	3.20%	0.0076	0.62%	0.0001	48.98%	0.0070	-	-	**63.56**%	0.0059
Caltech256	21504⟶ 120	_a	-	1.52%	0.0017	2.61%	<0.0001	0.38%	<0.0001	8.92%	0.0027	-	-	**28.79**%	0.0023
Flower102	21504⟶ 48	_a	-	1.80%	<0.0001	1.52%	<0.0001	1.15%	<0.0001	15.77%	0.0027	-	-	**32.33**%	0.0033
Indoor67	21504⟶ 35	_a	-	2.50%	<0.0001	1.64%	<0.0001	1.47%	<0.0001	9.83%	0.0010	-	-	**19.21**%	0.0020
Olivetti face	4096⟶ 5	_a	-	9.97%	0.0025	2.33%	0.0001	2.70%	0.0001	28.23%	0.0301	-	-	**51.12**%	0.0109
PFID61	21504⟶ 20	_a	-	4.70%	0.0001	2.24%	<0.0001	2.07%	<0.0001	16.07%	0.0050	-	-	**35.30**%	0.0334
Scene15	43008⟶ 5	_a	-	14.58%	0.0287	7.26%	0.0089	6.20%	0.0287	34.96%	0.0101	-	-	**55.41**%	0.0149

_a: The method can not give results in 2 hours; _b: Out of memory in Matlab environment;
_: LDA can not give results under current dimension.

TABLE V
IMAGE RECOGNITION ACCURACY BASE ON SINGLE FEATURE

METHOD	Caltech-101 (15 tr.)	Caltech-101 (30 tr.)	Caltech-256 (15 tr.)	Caltech-256 (30 tr.)	Scene15 (100 tr.)	Flower102 (15 tr.)	Indoor67 (30 tr.)	Pfid 61 (15 tr.)
BOIMAN (CVPR 08[44])	65.0	70.4	26.8	-	-	-	-	-
JAIN (CVPR 08[47])	61.0	69.5	-	-	-	-	-	-
LAZEBNIK (CVPR 06[48])	56.4	64.4	-	-	81.4	-	-	-
VAN GEMERT (TPAMI 10[49])	-	64.1	-	27.4	76.7	-	-	-
YANG (CVPR 09[50])	67.0	73.2	27.7	34.0	80.3	-	-	-
ZHANG (CVPR 06[51])	59.1	66.2	-	-	-	-	-	-
WANG (CVPR 10[52])	65.4	73.4	**34.5**	**41.1**	-	-	-	-
YU (PR 13[53])	-	-	-	-	71.6	-	-	-
GEMERT (ECCV 08[54])	-	64.2	-	27.2	76.6	-	-	-
WRIGHT (TPAMI 09[55])	64.9	70.7	-	-	-	-	-	-
AHARON (TSP 06[56])	65.2	73.2	-	-	-	-	-	-
JIANG (TPAMI 13[57])	67.7	73.6	28.9	34.3	-	-	-	-
LIU (ICCV 11[58])	-	76.5	-	-	82.2	-	-	-
BOUREAU (ICML 10[59])	-	75.1	-	-	80.6	-	-	-
BOUREAU (ICCV 11[60])	-	77.1	-	38.1	-	-	-	-
KHAN F (IJCV 12[61])	-	76.2	-	-	-	-	-	-
KHAN R (CVPR 13[62])	-	-	-	-	-	47.0	-	-
NI (CVPR 12[63])	-	-	-	-	-	-	-	42.9
YANG (CVPR 12[64])	-	-	-	-	-	-	-	28.2
BAO (TIP 13[65])	-	-	-	-	-	-	-	42.6
p NILSBACK(CVGI 08[66])	-	-	-	-	-	32.0	-	-
GEHLER(ICCV 09[45])	61.2	69.7	-	-	-	32.0	-	-
YUAN(CVPR 10[67])	65.0	-	-	-	-	33.0	-	-
THE PROPOSED MLRL (5000-5000-10000)	**71.8**	**77.6**	31.3	38.2	**82.4**	**53.9**	30.1	**45.6**

_: The literature do not give results.

1) For the Caltech256 dataset, the testing accuracy of the proposed method is about **19**, **12**, **75**, and **3.2** times higher than that of NPE, IsoP, LSDA, and LPP, respectively.
2) For the Flower102 dataset, the testing accuracy of the proposed method is about **17**, **21**, **29**, and **2** times higher than that of NPE, IsoP, LSDA, and LPP, respectively.

2) Supervised Expanded Representation for Image Recognition: In this section, we test six image datasets and compare our method with many other state-of-the-art approaches. The classification performance difference per category among the different state-of-the-art approaches is given in Table V. Our proposed method provides a better or comparable performance than the compared methods. It should be highlighted that these comparative experiments are unfair, since the proposed method is a single classifier with single type features, while other state-of-the-art approaches combine preprocessing methods, multifeature fusion, image segmentation technologies, and multiclassifiers. But the generalization performance obtained by the proposed method is mostly better than that of all the competing approaches.

C. Unsupervised Feature Representation

1) Image Reconstruction and Dimension Reduction: As the most important property of a deep network, the image reconstruction capability (or image compression) has been widely discussed in recent years. Hinton and Salakhutdinov [15] showed that multilayer BP networks with auto-encoders can be used for image reconstruction. In this section, our proposed

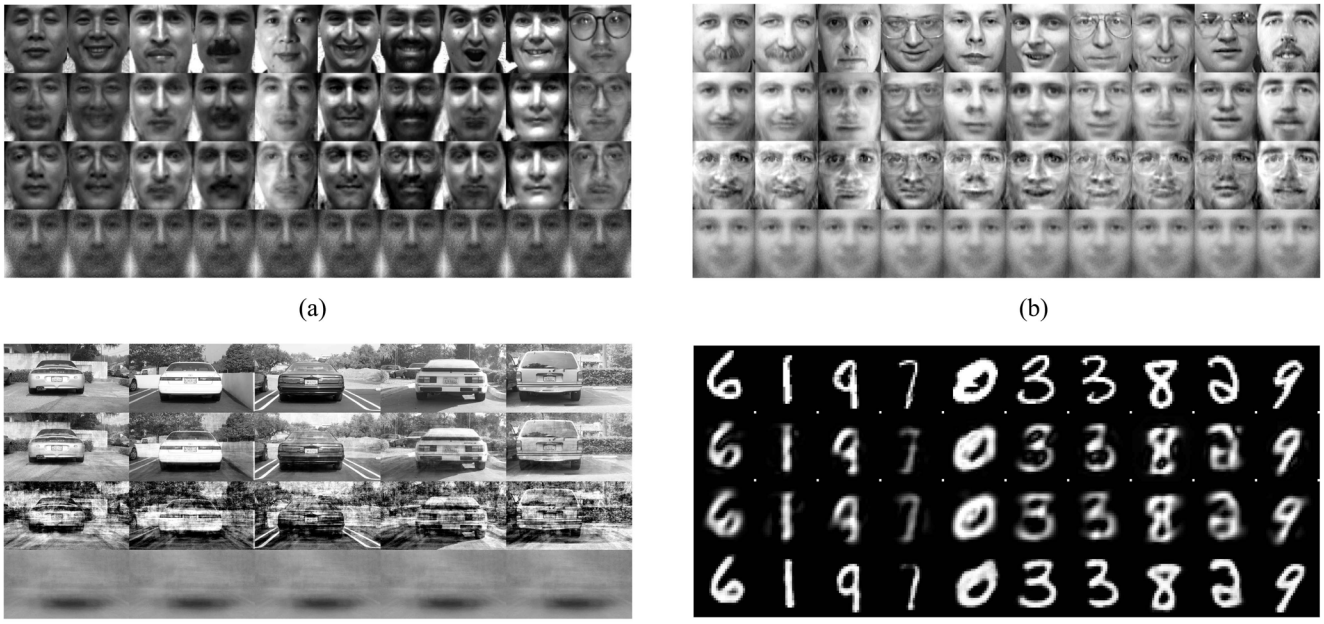


Fig. 5. For each subfigure top to bottom. (a) Random samples from Yale test dataset, reconstructions produced by the 20-dimensional proposed method, reconstructions by the 20-dimensional standard PCA, and reconstructions by 20-dimensional DBN. The average testing RMSE per image for the last three rows is 0.0961, 0.1127, and 0.1761. (b) Top to bottom: random samples from Olivetti face test dataset, reconstructions produced by the 40-dimensional proposed method, reconstructions by 40-dimensional standard PCA, and reconstructions by 40-dimensional DBN. The average testing RMSE per image for the last three rows is 0.1012, 0.1270, and 0.1968. (c) Top to bottom: random samples from Car dataset, reconstructions produced by the 100-dimensional proposed method, reconstructions by 100-dimensional standard PCA, and reconstructions by 100-dimensional DBN. The average training RMSE per image for the last three rows is 0.0630, 0.2170, and 0.2370. (d) Top to bottom: random samples from Mnist dataset, reconstructions produced by the 20-dimensional proposed method, reconstructions by 20-dimensional standard PCA, and reconstructions by 20-dimensional DBN. The average training RMSE per image for the last three rows is 0.1563, 0.1586, and 0.1236.

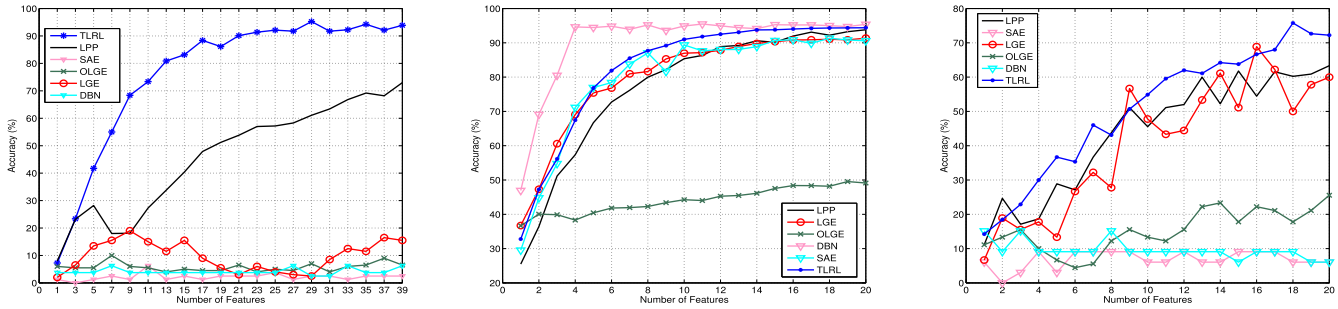
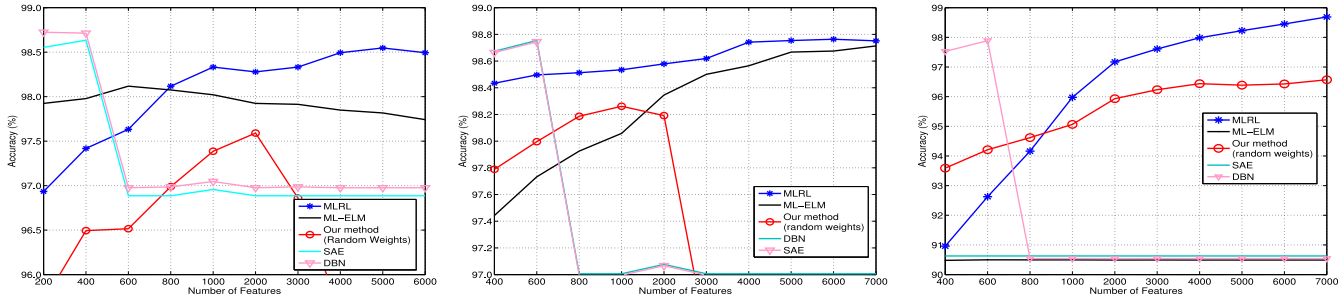


Fig. 6. Average testing accuracy by using LPP, SAE, OLGE, LGE, DBN and the proposed method on Olivetti face, USPS, and Yale Face, where the x- and y-axis show the number of features and average testing accuracy, respectively. Results on (a) Olivetti face, (b) USPS, and (c) Yale Face.

TABLE VI
PERFORMANCE COMPARISON OF UNSUPERVISED DIMENSION REDUCTION (MEAN: AVERAGE TESTING ACCURACY, STD: STANDARD DEVIATION)

DATASETS	#DIMENSIONALITY	OLGE		LPP		LGE		SAE		DBN		PROPOSED TLRL	
		MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
Caltech101	21504⟶ 51	21.55%	0.0071	16.09%	0.0079	28.98%	0.0048	34.12%	0.0287	9.06%	0.0074	**64.37**%	0.0064
Olivetti face	4096⟶ 39	6.57%	0.0268	73.23%	0.0676	52.63%	0.0644	2.57%	0.0375	6.74%	0.0271	**85.60**%	0.0297
Scene15	43008⟶ 15	38.84%	0.0090	45.63%	0.0149	40.08%	0.0198	25.10%	0.0704	25.23%	0.2578	**54.82**%	0.0189
USPS	256⟶ 10	44.25%	0.0061	90.37%	0.0014	87.00%	0.0015	89.37%	0.0028	**94.49**%	0.0010	90.65%	0.0010
Yale face	4096⟶ 10	10.53%	0.0372	45.56%	0.1553	52.63%	0.0610	6.62%	0.0328	7.87%	0.0487	**62.67**%	0.0209

method has been compared with the state-of-the-art DBN method and PCA method on the Yale face, the Olivetti face, the MNIST, and the Car image sets. In Fig. 5(a) and (b), we use a 4096(input)-500-100-20/40 DBN network to extract codes for all testing samples in the Yale and Olivetti face sets. Also, our proposed method [network architecture: 4096(input)-100-20/40] is used to extract features for all the testing samples in the Yale and Olivetti face sets. As seen



(a) (b) (c)

Fig. 7. Expanded representation learning: compared experimental results on USPS, W3a, and Covtype, where the x- and y-axis show the number of features and average testing accuracy, respectively. Results on (a) USPS, (b) w3a, and (c) IJCNN.

TABLE VII
PERFORMANCE COMPARISON OF EXPANDED DIMENSION REPRESENTATION (MEAN: AVERAGE TESTING RMSE, TIME: AVERAGE TRAINING TIME)

DATASETS	ML-ELM (OUTPUT DIMENSIONS)		MLRL (RANDOM WEIGHTS)) (OUTPUT DIMENSIONS)		SAE (OUTPUT DIMENSIONS)		DBN (OUTPUT DIMENSIONS)		MLRL (OUTPUT DIMENSIONS)	
	MEAN	TIME (s)	MEAN	TIME (s)	MEAN	TIME (s)	MEAN	TIME (s)	MEAN	TIME (s)
Codrna	81.76% (1000)	4.71	95.80% (1000)	3.21	90.63% (400)	176.60	66.79% (400)	154.87	**96.19%** (1000)	39.77
Connect-4	75.76% (7000)	72.47	82.98% (7000)	67.03	83.88% (600)	871.89	84.09% (600)	515.79	**84.33%** (7000)	323.69
Covtype	76.67% (1000)	12.41	81.37% (1000)	25.84	**84.07%** (200)	1978.09	83.85% (200)	1521.67	81.84% (1000)	233.52
IJCNN	90.05% (7000)	77.89	96.57% (7000)	216.50	90.63% (600)	241.06	97.87% (600)	226.79	**98.69%** (6000)	996.78
USPS	98.12% (600)	4.78	97.59% (2000)	1.23	98.71% (400)	97.64	**98.78%** (400)	73.14	98.55% (6000)	28.61
W3a	98.71% (7000)	57.42	98.26% (7000)	74.01	98.73% (600)	497.70	98.74% (600)	383.03	**98.78%** (7000)	270.87

TABLE VIII
PERFORMANCE COMPARISON (ACCURACY: MEAN TESTING CLASSIFICATION ACCURACY, TIME: TRAINING TIME, STD: STANDARD DEVIATION)

DATASETS(#NODES)	ELM			I-ELM			MLRL			TLRL		
	ACCURACY	STD	TIMES(s)	ACCURACY	STD	TIMES(s)	ACCURACY	STD	TIMES(s)	ACCURACY	STD	TIMES(s)
a9a (2000)	84.28%	0.0083	109.76	86.42%	0.0064	38.27	**86.89%**	0.0037	31.43	86.45%	0.0029	21.01
acoustic (2000)	69.08%	0.0027	107.17	77.09%	0.0017	48.60	**78.49%**	0.0005	65.95	78.21%	0.0009	41.07
Caltech101(30.tr) (500)	58.62%	0.0034	3.21	48.95%	0.0104	113.40	76.61%	0.0013	107.81	**76.94%**	0.0010	94.36
Connect4 (2000)	80.65%	0.0049	30.90	75.11%	0.0038	110.57	**81.54%**	0.0008	44.03	80.79%	0.0017	21.71
Covtype (1000)	80.07%	0.0021	35.33	70.07%	0.0017	43.63	**81.47%**	0.0010	120.87	81.32%	0.0741	97.10
Flower102(500)	11.43%	0.0028	1.65	14.55%	0.0061	83.90	35.70%	0.0089	81.01	**39.30%**	0.0051	74.08
IJCNN (2000)	94.37%	0.0020	31.14	90.50%	<0.0001	71.05	**95.14%**	0.0014	54.44	94.80%	0.0009	37.10
Olive face(500)	63.75%	0.0741	0.18	67.13%	0.0344	15.90	96.63%	0.0242	0.66	**97.50%**	0.0102	0.38
Scene15(500)	58.24%	0.0141	1.67	55.58%	0.0164	79.10	79.25%	0.0035	79.90	**79.51%**	0.0057	67.65
w3a (2000)	98.10%	0.0003	4.15	97.30%	0.0010	50.19	**98.37%**	0.0004	13.22	98.34%	0.0003	7.97
Yale face (500)	78.00%	0.0650	0.28	66.00%	0.0457	13.70	**90.67%**	0.0311	0.44	87.56%	0.0215	0.27

in these figures, the proposed method provides much better reconstructions than do DBN and PCA. In Fig. 5(c), we use a 530432(input)-1000-500-100 DBN and our method [network architecture: 530432(input)-1000-100] to extract codes for all training samples in Car sets. As seen in Fig. 5(c), the proposed method clearly outperforms PCA and DBN. In Fig. 9(d), we test our method [784(input)-200-20] and DBN [784(input)-500-250-100-20] to extract features for all the testing samples in the Mnist dataset. Different from other reconstruction results, the DBN produces a better visualization of the data than do PCA or our proposed method. The same experimental result are also obtained for USPS [see Fig. 6(b)]. Because USPS and Mnist datasets are "high sparse datasets," these experimental results show that DBN is more suitable for some "high sparse" datasets, while our proposed method is more suitable for general image datasets.

Furthermore, we carry out comparative experiments between our proposed method and other unsupervised feature extraction methods. Table VI and Fig. 6 display the performance comparison of DBN, LPP, SAE, LGE, OLGE, and the proposed method. As we can see, our proposed algorithms consistently outperform most of the other feature extraction algorithms on these image datasets.

2) Unsupervised Expanded Representation: Different from dimension reduction problems, in this section we increase the dimensions of datasets to increase testing accuracy. Tables VII and VIII and Fig. 7 show the results of using the original datasets without any distortions to test the performance of our method with respect to DBN, SAE, and ML-ELM. Furthermore, because ML-ELM is a feature representation method without iterative tuning, in this test we also carry out some experiments between

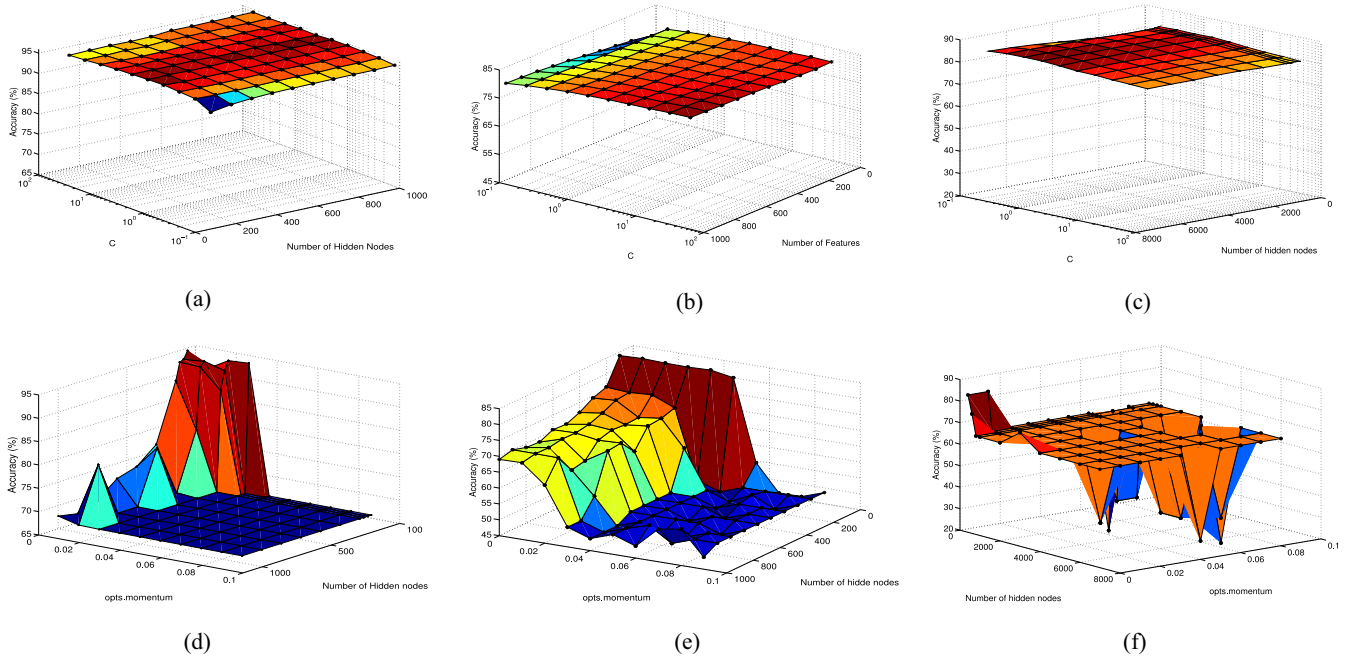


Fig. 8. Sensitive performance on Codrna, Covtype, and Connect dataset. Unlike the performance of DBN, Performance of our method is not sensitive to the user-specified parameters, and good testing accuracies can be achieved. (a)–(c) Sensitive performance with our proposed method on Codrna, Covtype, and Connect. (d)–(f) Sensitive performance with DBN on Codrna, Covtype, and Connect.

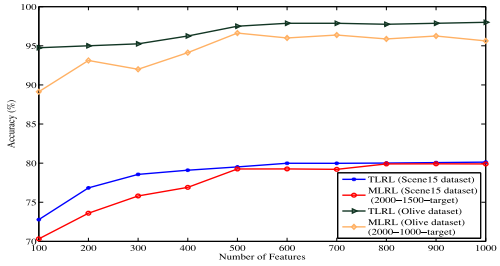


Fig. 9. Compared experimental results between TLRL and MLRL on USPS and W3a.

ML-ELM and our method without iterative tuning. As seen in Tables VII and VIII and Figs. 7 and 8, the proposed method has several interesting advantages different from other methods as follows.

1) The proposed method provides a comparable or better performance and has the least amount of required training time with respect to DBN, SAE, ELM, and Incremental ELM. If fine-tuning is not required, our method with random weights also provides a comparable performance to ML-ELM and runs several times faster than DBN and SAE.
2) Contrary to other learning methods, which have many parameters and are sensitive to the parameters, Fig. 8 shows that parameters in our method are not sensitive to generalization performance. The user can choose these parameters randomly at the outset without affecting generalization performance in the learning process.
3) Table VIII and Fig. 9 show that for supervised/unsupervised dimension reduction, TLRL provides a better generalization performance than MLRL. In other words, the best parameter c equals 2. However, for supervised/unsupervised expanded representation, MLRL provides a better generalization performance than TLRL.

V. Conclusion

This paper proposes an efficient representation learning algorithm based on ML-ELM, which has also been rigorously proven. The proposed method has several interesting features different from traditional representation learning methods.

1) Contrary to some feature learning methods, which only work for one type of problem, this paper provides a unified representation learning platform with unsupervised/supervised and compressed/expanded representation learning. The experimental results show that the proposed method can provide a similar or much better generalization performance compared to other representation learning methods.
2) Different from NN-based representation methods such as DBN, SAE, and ML-ELM, in which the combination of parameters is very sensitive to the performance, experimental results show that parameters in our method are not sensitive to generalization performance. In essence, the proposed methods give a new forward encoding learning method that is completely different from BP-based multilayer networks.
3) There still exists considerable space for us to explore the proposed methods in the near future. Strictly speaking, from a network architecture point of view, this paper simply uses one type of input form. Multifeatures have been extensively investigated by using local receipt field. Further exploration is required to construct the more

complex stacked network of MLRL by performing subsequent local connections on the output of the previous combinatorial layer. Furthermore, the proposed method needs an invertible activation function. Thus, it is worth investigating how to deal with this limitation.

ACKNOWLEDGMENT

The authors would like to thank the editor and anonymous reviewers for their invaluable suggestions, which have been incorporated to improve the quality of this paper.

REFERENCES

[1] X. F. He, M. Ji, C. Zhang, and H. Bao, "A variance minimization criterion to feature selection using Laplacian regularization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 10, pp. 2013–2025, Oct. 2011.

[2] D. Cai, X. F. He, K. Zhou, J. Han, and H. Bao, "Locality sensitive discriminant analysis," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, Hyderabad, India, Jan. 2007, pp. 708–713.

[3] D. Cai, X. F. He, and J. Han, "Isometric projection," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, Vancouver, BC, Canada, Jul. 2007, pp. 528–533.

[4] X. F. He, D. Cai, and J. Han, "Learning a maximum margin subspace for image retrieval," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 2, pp. 189–201, Feb. 2008.

[5] X. He, D. Cai, S. Yan, and H.-J. Zhang, "Neighborhood preserving embedding," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Beijing, China, Oct. 2005, pp. 1208–1213.

[6] D. Cai, X. F. He, and J. Han, "Spectral regression for efficient regularized subspace learning," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, vols. 1–6. Rio de Janeiro, Brazil, 2007, pp. 214–221.

[7] X. F. He and P. Niyogi, "Locality preserving projections," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 16. Vancouver, BC, Canada, 2004, pp. 153–160.

[8] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1530, 2006.

[9] R. Collobert, F. Sinz, J. Weston, and L. Bottou, "Large scale transductive SVMs," *J. Mach. Learn. Res.*, vol. 7, pp. 1687–1712, Aug. 2006.

[10] 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.

[11] Z. Xu, I. King, M. R. T. Lyu, and R. Jin, "Discriminative semi-supervised feature selection via manifold regularization," *IEEE Trans. Neural Netw.*, vol. 21, no. 7, pp. 1033–1047, Jul. 2010.

[12] S. Wang, Q. Huang, S. Jian, and Q. Tian, "S^3MKL: Scalable semi-supervised multiple Kernel learning for real-world image applications," *IEEE Trans. Multimedia*, vol. 14, no. 4, pp. 1259–1274, Aug. 2012.

[13] S. M. Xiang, F. P. Nie, and C. S. Zhang, "Semi-supervised classification via local spline regression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 11, pp. 2039–2053, Nov. 2010.

[14] X. Y. Zhou and C. P. Li, "Semi-supervised classification based on smooth graphs," in *Database Systems for Advanced Applications* (LNCS 3882). Berlin, Germany: Springer, 2006, pp. 757–766.

[15] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[16] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advanced in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2007.

[17] 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, pp. 3371–3408, Dec. 2010.

[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, Dec. 2013.

[19] G. B. Huang, Z. Bai, L. L.C. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 18–29, May 2015.

[20] S. Wang, F. Chung, J. Wu, and J. Wang, "Least learning machine and its experimental studies on regression capability," *Appl. Soft Comput.*, vol. 21, pp. 677–684, Aug. 2014.

[21] 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, Dec. 2006.

[22] S. Suresh, K. Dong, and H. J. Kim, "A sequential learning algorithm for self-adaptive resource allocation network classifier," *Neurocomputing*, vol. 73, nos. 16–18, pp. 3012–3019, Oct. 2010.

[23] X. Z. Wang, A. X. Chen, and H. M. Feng, "Upper integral network with extreme learning mechanism," *Neurocomputing*, vol. 74, no. 16, pp. 2520–2525, Sep. 2011.

[24] X. Z. Wang, Q. Y. Shao, Q. Miao, and J. H. Zhai, "Architecture selection for networks trained with extreme learning machine using localized generalization error model," *Neurocomputing*, vol. 102, pp. 3–9, Feb. 2013.

[25] A. Fu, C. Dong, and L. Wang, "An experimental study on stability and generalization of extreme learning machines," *Int. J. Mach. Learn. Cybern.*, vol. 6, no. 1, pp. 129–135, Feb. 2015.

[26] Y. M. Yang *et al.*, "Data partition learning with multiple extreme learning machines," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1463–1475, Aug. 2015.

[27] A. Riccardi, F. Fernandez-Navarro, and S. Carloni, "Cost-sensitive AdaBoost algorithm for ordinal regression based on extreme learning machine," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1898–1909, Oct. 2014.

[28] G. R. Feng, Y. Lan, X. P. Zhang, and Z. X. Qian, "Dynamic adjustment of hidden node parameters for extreme learning machine," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 279–288, Feb. 2015.

[29] G. B. Huang, H. M. Zhou, X. J. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[30] R. Zhang, Y. Lan, G. B. Huang, Z. B. Xu, and Y. C. Soh, "Dynamic extreme learning machine and its approximation capability," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2054–2065, Dec. 2013.

[31] G. B. Huang, "What are extreme learning machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle," *Cogn. Comput.*, vol. 7, no. 3, pp. 263–278, Jun. 2015.

[32] J. Luo, C.-M. Vong, and P.-K. Wong, "Sparse Bayesian extreme learning machine for multi-classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 4, pp. 836–843, Apr. 2014.

[33] Z. Bai, G.-B. Huang, D. Wang, H. Wang, and M. Westover, "Sparse extreme learning machine for classification," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1858–1870, Oct. 2014.

[34] A. Baradarani, Q. M. J. Wu, and M. Ahmadi, "An efficient illumination invariant face recognition framework via illumination enhancement and DD-DTCWT filtering," *Pattern Recognit.*, vol. 46, no. 1, pp. 57–72, Jan. 2013.

[35] W. Jun, W. Shitong, and F.-L. Chung, "Positive and negative fuzzy rule system, extreme learning machine and image classification," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 4, pp. 261–271, Dec. 2011.

[36] J. W. Cao and Z. P. Lin, "Extreme learning machines on high dimensional and large data applications: A survey," *Math. Prob. Eng.*, vol. 2015, pp. 1–12, Mar. 2015.

[37] J. W. Cao and L. L. Xiong, "Protein sequence classification with improved extreme learning machine algorithms," *Biomed. Res. Int.*, vol. 2014, no. 1, pp. 1–12, Mar. 2014.

[38] G. B. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," *Cogn. Comput.*, vol. 6, no. 3, pp. 376–390, Sep. 2014.

[39] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[40] Y. M. Yang, Y. N. Wang, and X. F. Yuan, "Bidirectional extreme learning machine for regression problem and its learning effectiveness," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1498–1505, Sep. 2012.

[41] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, 2000.

[42] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, USA, Jun. 2008, pp. 1–8.

[43] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Kyoto, Japan, Sep./Oct. 2009, pp. 221–228.

[44] D. Cai, X. F. He, J. W. Han, and H. J. Zhang, "Orthogonal Laplacianfaces for face recognition," *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3608–3614, Nov. 2006.

[45] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, USA, Jun. 2008, pp. 1–8.
[46] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognition natural scene categories," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. New York, NY, USA, Jun. 2006, pp. 2169–2178.
[47] J. van Gemert, C. J. Veenman, A. W. M. Smeuldes, and J. Geusebroek, "Visual word ambiguity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 7, pp. 1271–1283, Jul. 2010.
[48] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 1794–1801.
[49] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, New York, NY, USA, Jun. 2006, pp. 2126–2136.
[50] J. Wang *et al.*, "Locality-constrained linear coding for image classification," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 3360–3367.
[51] J. Yu, D. Tao, Y. Rui, and J. Cheng, "Pairwise constraints based multiview features fusion for scene classification," *Pattern Recognit.*, vol. 46, no. 2, pp. 483–496, Feb. 2013.
[52] J. C. van Gemert, J.-M. Geusebroek, C. Veenman, and A. W. M. Smeulders, "Kernel codebooks for scene categorization," in *Proc. IEEE Eur. Conf. Comput. Vis.*, Marseille, France, 2008, pp. 696–709.
[53] J. Wright, A. Y. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.
[54] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
[55] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2651–2664, Nov. 2013.
[56] L. Liu, L. Wang, and X. Liu, "In defense of soft-assignment coding," in *Proc. IEEE Int. Conf. Comput. Vis.*, Barcelona, Spain, 2011, pp. 2486–2493.
[57] Y. Boureau, J. Ponce, and Y. Lecun, "A theoretical analysis of feature pooling in vision algorithms," in *Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, Jun. 2010, pp. 1–8.
[58] Y. Boureau, N. L. Roux, F. Bach, J. Ponce, and Y. Lecun, "Ask the locals: Multi-way local pooling for image recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 2651–2658.
[59] F. S. Khan, J. van de Weijer, and M. Vanrell, "Modulating shape features by color attention for object recognition," *Int. J. Comput. Vis.*, vol. 98, no. 1, pp. 49–64, May 2012.
[60] R. Khan *et al.*, "Discriminative color descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, Jun. 2013, pp. 2866–2873.
[61] B. Ni, M. Xu, J. Tang, S. Yan, and P. Moulin, "Omni-range spatial contexts for visual classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 3514–3521.
[62] S. L. Yang, M. Chen, D. Pomerleau, and R. Sukthankar, "Food recognition using statistics of pairwise local features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 2249–2256.
[63] B.-K. Bao, G. Liu, R. Hong, S. Yan, and C. Xu, "General subspace learning with corrupted training data via graph embedding," *IEEE Trans. Image Process.*, vol. 22, no. 11, pp. 4380–4393, Nov. 2013.
[64] M. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis. Graph. Image Process.*, Bhubaneswar, India, Dec. 2008, pp. 722–729.
[65] X. Yuan and S. Yan, "Visual classification with multi-task joint sparse representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 3493–3500.
[66] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, pp. 3056–3062, 2007.

Yimin Yang (S'10–M'13) received the Ph.D. degree in electrical engineering from the College of Electrical and Information Engineering, Hunan University, Changsha, China, in 2013.

He is currently a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada. He has authored or coauthored over 20 refereed papers. His current research interests include artificial neural networks, hybrid system approximation, and image feature selection.

Dr. Yang has been serving as a Reviewer for international journals of his research field, such as the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, the *Neurocomputing*, and the *Neural Networks*.

Q. M. Jonathan Wu (M'92–SM'09) received the Ph.D. degree in electrical engineering from the University of Wales, Swansea, U.K., in 1990.

From 1995 to 2005, he was affiliated with the National Research Council of Canada, where he became a Senior Research Officer and a Group Leader. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada. He has published over 300 peer-reviewed papers in computer vision, image processing, intelligent systems, robotics, and integrated microsystems. His current research interests include 3-D computer vision, active video object tracking and extraction, interactive multimedia, sensor analysis and fusion, and visual sensor networks.

Dr. Wu holds the Tier 1 Canada Research Chair in Automotive Sensors and Information Systems. He is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the *Cognitive Computation*, and the *International Journal of Robotics and Automation*. He has served on technical program committees and international advisory committees for many prestigious conferences.