Neural Process Lett
https://doi.org/10.1007/s11063-018-9869-6

Optimizing Extreme Learning Machine via Generalized Hebbian Learning and Intrinsic Plasticity Learning

Chao Chen[1] · **Xinyu Jin**[1] · **Boyuan Jiang**[1] · **Lanjuan Li**[2]

Abstract Traditional extreme learning machine (ELM) has random weights between input layer and hidden layer, this kind of random feature mapping brings non-discriminative feature space and unstable classification accuracy, which greatly limits the performance of the ELM networks. Therefore, to get the well-pleasing input weights, two biologically inspired, unsupervised learning methods were introduced to optimize the traditional ELM networks, namely the generalized hebbian algorithm (GHA) and intrinsic plasticity learning (IPL). The GHA is able to extract the principal components of the input data of arbitrary size, while the IPL tunes the probability density of the neuron's output towards a desired distribution such as exponential distribution or weber distribution, thereby maximizing the networks information transmission. With the incorporation of the GHA and IPL approach, the optimized ELM networks generates a discriminative feature space and preserves much more characteristic of the input data, accordingly, achieving a better task performance. Based on the above two unsupervised methods, a simple, yet effective hierarchical feature mapping extreme learning machine (HFMELM) is further proposed. With almost no information loss in the layer-wise feature mapping process, the HFMELM is able to learn the high-level representation of the input data. To evaluate the effectiveness of the proposed methods, extensive experiments on several datasets are presented, the results show that the proposed methods significantly outperform the traditional ELM networks.

✉ Xinyu Jin
jinxy@zju.edu.cn

Chao Chen
chench@zju.edu.cn

Boyuan Jiang
byjiang@zju.edu.cn

Lanjuan Li
ljli@zju.edu.cn

1 Institution of Information Science and Electrical Engineering, Zhejiang University, Hangzhou 310037, Zhejiang, China

2 State Key Laboratory for Diagnosis and Treatment of Infectious Diseases, First Affiliated Hospital, College of Medicine, Zhejiang University Hangzhou, Hangzhou, China

Published online: 26 July 2018

Keywords Extreme learning machine · Random feature mapping · Generalized hebbian learning · Intrinsic plasticity learning · Hierarchical feature mapping ELM

1 Introduction

In the recent decades, numerous architectures of neural networks have been proposed. Among them, the feed-forward neural networks are widely studied and applied because of its simple structure and good generalization performance. It has been proved that multi-layer feed-forward neural networks with non-linear activation function can approximate any continuous function [1]. On this basis, S. Tamura et al. proved that single hidden layer feed-forward neural networks (SLFNs) have the same approximate ability as the multi-layer feed-forward neural networks [2]. After that, a special feed-forward neural networks, the extreme learning machine (ELM) first proposed by Huang [3], which determines its input weights randomly, has become a very popular classifier due to its fast learning speed, satisfactory performance and little human intervention [4]. Different from the traditional feed-forward neural networks, the ELM networks does not adopt the back-propagation based iterative algorithm to update the input weights, instead, the input weights are randomly initiated and do not need to be tuned, which can be considered as random feature mapping or random projection. Theoretically, Huang et al. proved that the ELM with randomly generated input weights and output weights calculated by regularized least square can approximate any continuous function. Further more, without updating the parameters, the ELM tends to achieve faster and better task performance than those of multi-layer perceptrons and SVMs [4]. Since it was first put forward, theories and applications of ELM have been extensively studied. Various extensions have been applied to the original ELM model to make it more efficient and suitable for specific applications. Liu X proposed a multi-kernel based ELM, which combines the ELM with kernel method [5]. Huang extended ELM to semi-supervised and unsupervised learning in [6]. A multi-layer ELM was proposed in [7], which learns hierarchical features and achieves higher generalization performance. Considering the computational cost and spatial requirements, an online version of ELM, called online sequential ELM (OS-ELM) was proposed and studied in [8–10]. The pointwise ELM and pairwise ELM were first introduced to be beneficial for relevance ranking problems in [11]. Zong et al proposed the weighted ELM (W-ELM) to handle the imbalanced data problem [12]. Ensemble learning based ELM has also been studied in [13,14]. In addition, ELM model has also been applied in various application tasks [15–18]. In spite of the desirable generalization ability and fast learning speed, moreover, considerable excellent achievements in many applications, ELM algorithm still has some shortcomings [19–21], specifically:

(a) Due to the random initialization of the input weights, the performance of the ELM networks may be unstable [19,20].
(b) The feature space is non-discriminative and the hidden neurons of the original ELM are incompact [19,21]. i.e., the completely random connection weights in the hidden layer do not always represent the discriminative features, such that the traditional ELM has to generate a great number of hidden nodes to meet a desirable generalization performance.

2 Related Works

All the above works mainly focused on the following issues: (1) How to extend the traditional ELM to other specific applications, such as unsupervised learning [6], online sequential

learning [8,9], imbalanced data problem [9] and so on. (2) How to optimize the ELM networks by additional statistical properties via adding various regularization terms, such as GEELM [17], MCVELM [15], Dropout ELM [22], etc. (3) How to extend the single hidden layer ELM networks to multi-layer networks and learn deep representations via ELM frameworks [7,23,24].

However, a very important aspect of the ELM networks that has not received much attention so far is how to exactly initialize the connection weights from the input layer to the hidden layer. Actually, compared with the traditional SLFNs, the most notable contribution of the ELM networks is that it replaces the supervised tuned connection weights with unsupervised random projection approach from the input layer to hidden layer. In this way, the output weights can be calculated by solving a linear system to get the global optimal solutions. Therefore, despite simplicity of the ELM networks, its performance is greatly limited by the random projection approach, which leads to unstable task performance and non-discriminative feature space [19]. Intuitively, a question can be raised that: Does the random projection produce the best input weights? If not, is there any other effective unsupervised methods to generate the well-suited input weights?

The answer is obviously affirmative. It is not until recently that some methods have been developed to determine a suitable input weights. A self-adjusting extreme learning machine (SA-ELM) was proposed in [20], in which the input-weights and the bias of hidden layer of ELM are adjusted with "teaching phase" and "learning phase" to minimize the objective function values based on the idea of the ameliorated teaching learning based optimization. A constrained ELM (CELM) algorithm is proposed in [19], some results have been obtained by constraining the input weights from the closed set of difference vectors of between-class samples. Li et al. [25] proposed a tuning ELM improved by artificial bee colony (ABC), which obtains the optimal input weights and bias, thus improves the performance of the ELM. Han et al. used modified PSO to optimize the input weights and the bias of the hidden layer of the ELM networks [26]. In [27], the adaptive differential evolution algorithm is combined with ELM to optimize the parameters of the hidden layer of ELM. Inspired by the intrinsic plasticity, a novel method under the notion of batch intrinsic plasticity(BIP) was introduced to optimize the ELMs [28,29]. The idea of the BIP is to adapt the slope and the bias of the hidden neurons such that the output of the hidden neurons is forced to approximate exponential distributions, which has been confirmed to maximize the networks' information transmission.

Inspired by the biologically plausible mechanism known as hebb's rule, we incorporate the generalized hebbian learning algorithm (GHA) into the ELM networks. It has been proved that the GHA can extract principal components of the input data [30]. Therefore, in order to get the well-suited input weights and obtain a discriminative feature space, we can replace the random feature mapping with the GHA approach, moreover, maximize the information transmission by tuning the output of the hidden neurons to desired distribution with the intrinsic plasticity learning (IPL) method.

The rest of this paper is organized as follows. In Sect. 3, the traditional ELM networks is introduced, and we explain the random weights from the input layer to the hidden layer from the perspective of random projection theory. The intrinsic plasticity learning and the generalized hebbian learning algorithm incorporated ELMs are described in detail in Sect. 4. In addition, the hierarchical feature mapping ELM (HFMELM) networks is further proposed in Sect. 5. In Sect. 6, we evaluate the performance of our proposed method in several commonly used datasets. The conclusions are given in the last section.

3 The ELM, Regularization and Random Feature Mapping

In this section, we first review the traditional ELM algorithm and output regularization. In addition, we explain the ELM from the perspective of random projection. As a special SLFN, the ELM networks has random connection weights between the input layer and the hidden layer, and the output weights are calculated by regularized least square. Compared with traditional back propagation based learning algorithm, the ELM with a closed-form solution is remarkably efficient and tends to reach a global optimum, thereby, achieves better task performance.

(1) Basic ELM: Given a training data set with N samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)|\boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{y}_i \in \mathbb{R}^m, i = 1, 2, \cdots N\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the n-dimensional input data, $\boldsymbol{y}_i \in \mathbb{R}^m$ is its corresponding labels. Assuming that L is the number of hidden neurons, $g(x)$ is activation function, such that the ELM model can be described as:

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i \cdot g(\boldsymbol{w}_i \boldsymbol{x}_j + b_i) = o_j \quad j = 1, 2, \cdots, N \tag{1}$$

where $\boldsymbol{w}_i = [w_{i1}, w_{i2}, \cdots, w_{id}]$ is the connection weights between the input-layer and the i-th hidden node, $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]$ is output weights connecting the i-th hidden node and the output nodes, b_i is the bias of the i-th hidden node. The ELM model can approximate these samples with zero error, means that

$$\sum_{j=1}^{N} \|o_j - y_j\| = 0 \tag{2}$$

i.e., there exists $\boldsymbol{\beta}_i$, $\boldsymbol{w}_i$ and b_i, such that

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i \cdot g(\boldsymbol{w}_i \boldsymbol{x}_j + b_i) = y_j \quad j = 1, 2, \cdots, N \tag{3}$$

The above equ. (3) can be written compactly as:

$$\boldsymbol{H}\boldsymbol{\beta} = \boldsymbol{Y} \tag{4}$$

where $\boldsymbol{H}$ is the output of hidden neurons, $\boldsymbol{Y}$ is the desired output matrix and $\boldsymbol{\beta}$ is output weights matrix.

$$\boldsymbol{H} = \begin{bmatrix} g(w_1 x_1 + b_1) & \dots & g(w_L x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(w_1 x_N + b_N) & \dots & g(w_L x_N + b_L) \end{bmatrix} \tag{5}$$

$$\boldsymbol{Y}_{N \times m} = \begin{bmatrix} \boldsymbol{y}_1^T \\ \vdots \\ \boldsymbol{y}_N^T \end{bmatrix} \tag{6}$$

Since the input weights $\boldsymbol{w}$ and bias b are determined randomly, the only free parameters to be optimized are the output weights $\boldsymbol{\beta}$, which can be obtained by computing the generalized pseudo-inverse

$$\boldsymbol{\beta} = \boldsymbol{H}^{\dagger}\boldsymbol{Y} \tag{7}$$

(2) Regularized ELM: In order to incorporate the additional statistical properties into the ELM networks, a large number of output regularization methods have been proposed

Algorithm 1 Regularized ELM Networks

Input: Training samples $\boldsymbol{x} \in \mathbb{R}^{N \times d}$
Output: Parameters of the ELM model $\boldsymbol{w} \in \mathbb{R}^{L \times d}$, $\boldsymbol{b} \in \mathbb{R}^{N \times L}$, $\boldsymbol{\beta} \in \mathbb{R}^{L \times c}$
1: Randomly generate the input weights $\boldsymbol{w} \in \mathbb{R}^{d \times L}$ and bias $b_0 \in \mathbb{R}^{L}$
2: Construct bias matrix $\boldsymbol{b} = [b_0; b_0; \cdots; b_0] \in \mathbb{R}^{N \times L}$
3: Calculate the output matrix of the hidden neurons $\boldsymbol{H} = g(\boldsymbol{x}\boldsymbol{w}^T + \boldsymbol{b})$, where $g(\cdot)$ is activation function.
4: Obtain the output weights by calculating $\boldsymbol{\beta} = (\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{Y}$

[15,17,22], among which, the most commonly practical one tries to minimize the norm of the output weights $\boldsymbol{\beta}$, which can be expressed as:

$$L_\beta = \frac{1}{2} \parallel \boldsymbol{\beta} \parallel^2 + \frac{C}{2} \parallel \boldsymbol{Y} - \boldsymbol{H}\boldsymbol{\beta} \parallel^2 \tag{8}$$

where C is the trade-off parameters between training errors and the norm of output weights. The free parameters can be effectively solved by setting the derivatives of $L_{\boldsymbol{\beta}}$ with respect to $\boldsymbol{\beta}$ to be zero, the output weights $\boldsymbol{\beta}$ are obtained by

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

where $\boldsymbol{I}$ is identity matrix and C is regularization coefficient, the regularized ELM algorithm can be concluded as Algorithm 1, which is the baseline of our proposal.

(3) Random Feature Mapping of ELM Networks: For the given n points in the d-dimensional space $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, In order to reduce the dimensions of the data matrix and speed up the computations, the random projection method defines a random projection matrix $\boldsymbol{R} \in \mathbb{R}^{d \times k}$, and maps the original data into a k-dimensional subspace by multiplying the original data matrix with the projection matrix, as shown below

$$\boldsymbol{B} = \boldsymbol{A}\boldsymbol{R} \in \mathbb{R}^{n \times k} \tag{10}$$

Generally, $k \ll d$. The theoretical basis of the dimensionality reduction based on random projection is Johnson-Lindenstrauss's JL theorem [31].

JL Theorem: Given $\varepsilon > 0$, and an integer n, let k be a positive integer such that $k \geq k_0 = O(\varepsilon^{-2}\log(n))$. For every set $\mathbb{P}$ of n points in $\mathbb{R}^d$, there exists a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$, such that for all $u, v \in \mathbb{P}$.

$$(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2 \tag{11}$$

The JL theorem states that the set of n points in d-dimensional Euclidean space could be projected onto the k-dimensional subspace, and the distance between the pair of points is almost invariant.

In the traditional ELM networks, the connection weights between the input layer and the hidden layer are determined randomly. This process can be regarded as non-linear random projection or random feature mapping. Not the same as the random projection algorithm used for data dimensionality reduction introduced in the JL theorem, the random feature mapping in the ELM networks has two main advantages. First of all, it is non-linear because the hidden layer has a non-linear activation function. Secondly, in the ELM networks, the number of hidden nodes is generally more than that of neurons in the input layer. It means that the random feature mapping maps the input data onto a high-dimensional feature space, which makes the original data more separable with almost no time consumption. However,

in spite of the convenience of the random feature mapping, the feature space may be non-discriminative and the hidden nodes are incompact in the traditional ELM networks [19]. For these reasons, in our proposed method, the GHA and IPL are incorporated into the traditional ELM networks to optimize the parameters of the hidden layer of ELM.

4 Proposed Method

In the previous section, the traditional ELM algorithm was introduced and we explained it from the perspective of random projection. Despite the fact that the random projection with a non-linear activation function can make the original data more separable in the feature space, it is undeniable that the traditional ELM algorithm exists the shortcomings of non-discriminative feature space and limited performance [19–21,25]. Therefore, in this section, we propose two biologically inspired unsupervised learning methods, generalized hebbian learning algorithm (GHA) and intrinsic plasticity learning (IPL), to improve the performance of the ELM networks. The generalized hebbian learning algorithm which is able to generate a principal subspace of the input data is designed to produce a discriminative feature mapping. In this way, the number of hidden nodes could be reduced while keeping the same performance as the ELM networks with the random feature mapping. Different from the random feature mapping or GHA, the intrinsic plasticity learning doesn't produce a new feature mapping matrix (input weights), instead, it tunes the output of hidden nodes to approximate a desired distribution with high entropy, such as exponential distribution or weber distribution [32], which will maximize the network's information transmission. With the help of the GHA and IPL method, the optimized ELM networks which is referred to as principal subspace learning ELM (PSLELM) tends to obtain a preferable feature mapping from the input layer to the hidden layer, to this end, it achieves a better performance.

4.1 Intrinsic Plasticity Learning

Intrinsic plasticity (IP) as an unsupervised, biologically inspired adaptation rule was first introduced in [33], and incorporated into ELM in [28,29]. The benefits of using intrinsic plasticity, that tunes the probability density of a neuron's output towards an exponential distribution, thereby realizing an information maximization, have already been demonstrated [34]. Considering a two layer full-connection network shown as Fig. 1 (input layer and hidden layer), giving the training data $\boldsymbol{x} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}) \in \mathbb{R}^{N\times d}$ and the input weights $\boldsymbol{w}$. Let $\boldsymbol{w}_i = [w_{i1}, w_{i2}, \ldots, w_{id}]$ denotes the connection weights between the input-layer and the i-th hidden node. the goal of the intrinsic plasticity learning (IPL) is to learn a set of parameters $\{a_i, b_i\}$ where a_i and b_i are the slope and bias for the i-th output neuron, such that the desired distribution for the neuron's outputs $h_i(k) = f(a_i s_i(k) + b_i)$ could be realized, where $f(a_i x + b_i)$ is the activation function for the output node i, $s_i(k) = \boldsymbol{x}^{(k)}\boldsymbol{w}_i^T$ denotes the response of the i-th neurons for the input $\boldsymbol{x}^{(k)}$, and the $\boldsymbol{s}_i = \boldsymbol{x}\boldsymbol{w}_i^T$ denotes the output of the i-th neuron for all the training set $\boldsymbol{x} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)})$. Therefore, to implement the intrinsic plasticity learning method in ELM networks, for each hidden neurons, targets $\boldsymbol{t}_{N\times 1} = (t_1, t_2, \ldots, t_N)^T$ are drawn from the weber distribution [32] and normalized to 0-1 first, then they should be sorted in ascending order such that $t_1 < t_2 < \cdots < t_N$. Since the output of each neurons $\boldsymbol{s}_i = (s_i(1), s_i(2), \ldots, s_i(N))$ should be mapped onto the right targets $\boldsymbol{t}$, we sort the $\boldsymbol{s}_i$ as $s_i(1) < s_i(2) < \cdots < s_i(N)$ in the same way. After that, the optimal slope a_i and bias b_i should be calculated, such that the actual output $\boldsymbol{h}_i$ of i-th hidden node could

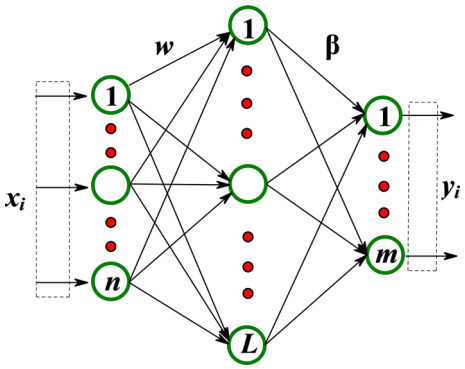

Fig. 1 The architecture of Extreme learning machine. The input weights are initialized randomly, only the output weights update during the training phase

be approximately mapped onto the desired output $\boldsymbol{t}$. The optimal parameters $\boldsymbol{v}_i = (a_i, b_i)^T$ could be easily derived by minimizing the following objective function

$$\min_{\boldsymbol{v}_i} ||f(\boldsymbol{\Phi}(\boldsymbol{s}_i) \cdot \boldsymbol{v}_i) - \boldsymbol{t}|| \tag{12}$$

i.e.

$$\min_{\boldsymbol{v}_i} ||\boldsymbol{\Phi}(\boldsymbol{s}_i) \cdot \boldsymbol{v}_i - f^{-1}(\boldsymbol{t})|| \tag{13}$$

where $\boldsymbol{\Phi}(\boldsymbol{s}_i)_{N\times 2} = (\boldsymbol{s}_i^T, \underbrace{(1, 1, \cdots, 1)}_{\text{with N 1}}^T)$, the optimal solution can therefore, like an ELM networks, be solved by calculating the Moore-Penrose pseudo-inverse:

$$\boldsymbol{v}_i = (a_i, b_i)^T = \boldsymbol{\Phi}^{\dagger}(\boldsymbol{s}_i) \cdot f^{-1}(\boldsymbol{t}) \tag{14}$$

Generally, the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ is deployed as the activation function, then the inverse function is $f^{-1}(\boldsymbol{t}) = -\ln(\frac{1}{\boldsymbol{t}} - 1)$. Despite that the above procedure should be performed for every neuron, the intrinsic plasticity learning shows no significant difference in the computational time due to its closed-form solution.

As shown in Algorithm 2, the intrinsic plasticity learning algorithm learns a slope and bias for each hidden neurons, in this respect, the outputs of the i-th hidden neuron approximate the target distribution after adapting by the activation function $h_i(k) = f(a_i s_i(k) + b_i)$. Compared with the random feature mapping in the traditional ELM networks, the intrinsic plasticity learning incorporated ELM (IPLELM) maximizes the information transmission from the input layer to the hidden layer, thereby, leads to better performance.

4.2 Generalized Hebbian Learning

Hebbian learning rule, also called as Hebb's rule is an important biologically inspired learning rule in neural science, which was first introduced by Donald Hebb in his 1949 book "The Organization of Behavior" [35]. The theory is often summarized as "Cells that fire together, wire together", which is described as a method to adjust the synaptic weights in artificial neurons. The Hebb's rule indicates that the connection weight between two neurons increases if the two neurons activate simultaneously, and reduces if the two neurons activate separately. According to the above description, the Hebb's rule can be generalized as $\Delta \boldsymbol{w} = \eta_{Hebb} \boldsymbol{x}(n) y(n)$, where $\boldsymbol{x}(n) \in \mathbb{R}^d$ and $y(n)$ are the input and single output of the synaptic respectively, η_{Hebb} is the small learning rate. To this end, the connection weights between the two layers can be updated as $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta_{Hebb} \boldsymbol{x}(n) y(n)$, where the

Algorithm 2 Intrinsic Plastic Learning

Input: Training data $\boldsymbol{x}$, input weights $\boldsymbol{w}$, num of hidden nodes L
Output: Slop and bias for every neuron $\{a_i, b_i\}_{i=1\sim L}$

for all hidden neurons i **do**
  $\boldsymbol{s}_i = \boldsymbol{x}\boldsymbol{w}_i^T$
  Generate target $\boldsymbol{t} = \{t_1, t_2, \cdots, t_N\}$ from desired distribution f_{des}
  Normalize $\boldsymbol{t}$ to (0,1) by calculating $\boldsymbol{t} = \dfrac{\boldsymbol{t}}{1.001 \cdot max(\boldsymbol{t})}$
  Sort the target and actual output as descending order $\boldsymbol{t} \leftarrow sort(\boldsymbol{t}), \boldsymbol{s} \leftarrow sort(\boldsymbol{s})$
  Construct $\boldsymbol{\Phi}(\boldsymbol{s}_i) = (\boldsymbol{s}_i^T, \underbrace{(1, 1, \cdots, 1)}_{\text{with N 1}}^T)$
  Calculating $\boldsymbol{v}_i = (a_i, b_i)^T = \boldsymbol{\Phi}^{\dagger}(\boldsymbol{s}_i) \cdot f^{-1}(\boldsymbol{t})$
return $\boldsymbol{a} = [a_1, a_2, \cdots, a_L], \boldsymbol{b} = [b_1, b_2, \cdots, b_L]$

$\boldsymbol{w}(n)$ and $\boldsymbol{w}(n+1)$ denote the connection weights at step n and step $n+1$ respectively. It has been proved that, the output $y(n)$ will converge to the largest principal component of the input when the connection weights of the two-layer networks update according to the Hebb's rule. However, the form of the learning rule will lead to infinite increase of the synaptic weights which is unacceptable. To solve this problem, a normalization term introduced in [36] shows that

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta y(n)[\boldsymbol{x}(n) - y(n)\boldsymbol{w}(n)] \tag{15}$$

where the negative term $-y(n)\boldsymbol{w}(n)$ guaranties the stability of the learning rule. The detailed analysis can be found in [36].

Biologically plausible though the hebbian learning is, it can only extract the largest principal component of the input. Fortunately, various extensions of the hebb's rule have been made to make it more suitable for multi-layer networks. The most well-known variant is generalized hebbian algorithm (GHA) proposed by Sanger [37], which is able to extract the principal components of the input data of arbitrary size. The GHA could be achieved by a two-layer fully connected feed-forward neural networks with L output neurons. Denoting that the synaptic weights between the input node i and the output node j as w_{ji}, one can compute the response of the output neuron j for the input set $\{x_i(n)|i = 1, 2, \cdots, d\}$ at training times n as

$$y_j(n) = \sum_{i=1}^{d} w_{ji}(n)x_i(n), \quad j = 1, 2, \cdots, L. \tag{16}$$

According to the generalized hebbian learning algorithm [37], the connection weights can be update as $w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}$, where

$$\Delta w_{ji} = \eta[y_j(n)x_i(n) - y_j(n)\sum_{k=1}^{j} w_{ki}y_k(n)], \quad i = 1, 2, \cdots, d; \quad j = 1, 2, \cdots, L \tag{17}$$

To be consistent with the Hebb's rule, we rewrite it as

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta y_j(n)[\boldsymbol{x}'(n) - y_j(n)\boldsymbol{w}_j(n)] \tag{18}$$

where $\boldsymbol{x}'(n)$ denotes the modified $\boldsymbol{x}(n)$

$$\boldsymbol{x}'(n) = \boldsymbol{x}(n) - \sum_{k=1}^{j-1} \boldsymbol{w}_k(n)y_k(n) \tag{19}$$

Algorithm 3 Generalized Hebbian Learning

Input: Training data, num of hidden nodes L, and num of training samples N
Output: Feature mapping weights $\boldsymbol{w}$

Initialize input weights $\boldsymbol{w}$ randomly
repeat
  for $n = 1$ to N **do**
    for $j = 1$ to L **do**
      $\boldsymbol{x}'(n) = \boldsymbol{x}(n) - \sum_{k=1}^{j-1} \boldsymbol{w}_k(n) y_k(n)$
      $\Delta \boldsymbol{w}_j(n) = \eta y_j \boldsymbol{x}'(n) - \eta y_j^2(n) \boldsymbol{w}_j(n)$
      $\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \Delta \boldsymbol{w}_j(n)$
until Convergency
Obtain feature mapping weights $\boldsymbol{w}$

The whole procedure of the GHA method is presented in **algorithm 3**. It's worth noting that the iterative procedure will be terminated once the number of iterative reaches T_{max} which is determined according to multiple experimental results, or the input weights $\boldsymbol{w}$ tends to convergency. Note that the GHA method is quite similar with the standard principal component analysis (PCA), both of them are used for extracting the principal components of the original data. PCA may be the most commonly used dimensionality reduction and feature extraction method, however, it has some limitations. First of all, the standard PCA networks can only achieve linear transformation of the input data, which is unacceptable for a neural realization. Secondly, PCA networks cannot usually separate independent subsignals from their linear mixture. Besides, the PCA method is only effective for gaussian data and stationary, linear processing operations [37]. Conversely, the component subspace learning derived from the perspective of hebb's rule realized a non-linear PCA learning which is much more favorable for a neural realization.

4.3 Proposed PSLELM Networks

Based on the above two unsupervised methods, the principal subspace learning (PSLELM) networks was proposed, which incorporated the IPL and GHA into the conventional ELM networks. In the proposed PSLELM networks, the GHA method was used for determining a feature mapping from the input layer to the hidden layer. Besides, the IPL method was applied for tuning the output of each hidden neurons to approximate the target distribution, which maximizes the information transmission from the input layer to the hidden layer. Compared with the traditional ELM networks with randomly initialized input weights, our proposal mainly has the following three advantages. First of all, the GHA approach which maps the input data to its principal subspace is capable of generating a discriminative feature space and preserving much more characteristic of the input data. Secondly, for each hidden neuron, the parameters of the activation function were trained to map the outputs onto the desired weber distribution. By this means, it leads to a non-linear principal subspace feature mapping and maximizes the information transmission as well. And lastly, as has been demonstrated in [38], it is noteworthy that the interaction of the intrinsic plasticity and the hebb's rule allow the neurons to discover heavy-tailed directions in the input. The details of the proposed PSLELM networks are described as Algorithm 4. It should be noted that in order to apply activation function in one-shot version, the slop and bias should be replicated as a matrix form first. i.e., $\boldsymbol{A} = [\boldsymbol{a}; \boldsymbol{a}; \cdots; \boldsymbol{a}] \in \mathbb{R}^{N \times L}$ and $\boldsymbol{B} = [\boldsymbol{b}; \boldsymbol{b}; \cdots; \boldsymbol{b}] \in \mathbb{R}^{N \times L}$. Then, the IPL

procedure can be achieved by $\boldsymbol{H} = f(\boldsymbol{A} \circ \boldsymbol{H}_0 + \boldsymbol{B})$, where "$\circ$" denotes the element-wise multiplication operation.

Algorithm 4 Training of PSLELM Networks

Input: Training data $\boldsymbol{x}$, num of hidden nodes L
Output: Input weights $\boldsymbol{w}$, slope and bias matrices $\boldsymbol{A}$,$\boldsymbol{B}$, and output weights $\boldsymbol{\beta}$
1: Initialize input weights $\boldsymbol{w}_0$ randomly
2: Obtain the input weights $\boldsymbol{w}$ according to the **Algorithm 3**
3: Calculate $\boldsymbol{H}_0 = \boldsymbol{x}\boldsymbol{w}^T$
4: Calculate the slope and bias $\boldsymbol{a}$,$\boldsymbol{b}$ according to **Algorithm 2**
5: Repmat $\boldsymbol{a}$,$\boldsymbol{b}$ as $A = [\underbrace{\boldsymbol{a}; \boldsymbol{a}; \cdots ; \boldsymbol{a}}_{withN\boldsymbol{a}}]_{N\times L}$ and $B = [\underbrace{\boldsymbol{b}; \boldsymbol{b}; \cdots ; \boldsymbol{b}}_{withN\boldsymbol{b}}]_{N\times L}$
6: Calculate the output matrix of the hidden neurons $\boldsymbol{H} = f(\boldsymbol{A} \circ \boldsymbol{H}_0 + \boldsymbol{B})$
7: Calculate the output weights $\boldsymbol{\beta} = (\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{Y}$
8: **return** $\boldsymbol{w}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\beta}$

5 Hierarchical Feature Mapping ELM Networks

From the perspective of optimizing the feature mapping from the input layer to the hidden layer, we have proposed two unsupervised approaches, namely the generalized hebbian learning algorithm (GHA) and intrinsic plasticity learning (IPL), to promote the feature learning ability of the feature mapping. However, due to the shallow structure of the baseline and our proposal, the feature learning ability may not be effective for high dimension data, e.g. natural images. It's not until recently that some new ELM-based hierarchical learning frameworks have been proposed to address this issue [7,23,24]. In this section, based on the GHA and the IPL method, we proposed a simple, yet effective approach to extend the single hidden layer ELM networks to Hierarchical Feature Mapping ELM Networks, which referred to as HFMELM.

The proposed HFMELM, aiming to learn deep representation with ELM networks, is a stacked model with more than one feature mapping layer, whereas the baseline or our improved ELM has only one random feature mapping layer or one optimized feature mapping layer. In order to extend the single hidden layer ELM networks to Hierarchical Feature Mapping ELM networks, several unsupervised feature mapping blocks are stacked to be a deep feature mapping block in our scheme. As we have mentioned above, there are two kinds of feature mapping blocks deployed in the HFMELM, which are IPL feature mapping block and PSL feature mapping block respectively. Unlike the existing deep learning frameworks, such as DBN and auto-encoder [39], where all the hidden layers are initialized randomly and trained with BP-based method, our proposed HFMELM is composed of several independent unsupervised feature learning layers and a least square regression layer. With this method, the networks can be established layer by layer in a feed-forward way. More importantly, without the supervised training step, the computational complexity of the HFMELM is in the same order of magnitude as the traditional ELM. The detailed description of the HFMELM networks shown as Fig. 2.

Fig. 2(A) shows the two single hidden layer ELM networks, which correspond to the proposed two ELM networks with improved input weights. The W_{IPL} denotes the feature mapping of the IPLELM, or called as IPL-layer. i.e., the connection weights are randomly determined, and the outputs of the hidden neurons are then adapted by the IPL method. In this way, The IPL-layer maps the input data onto a high dimension feature space, at the

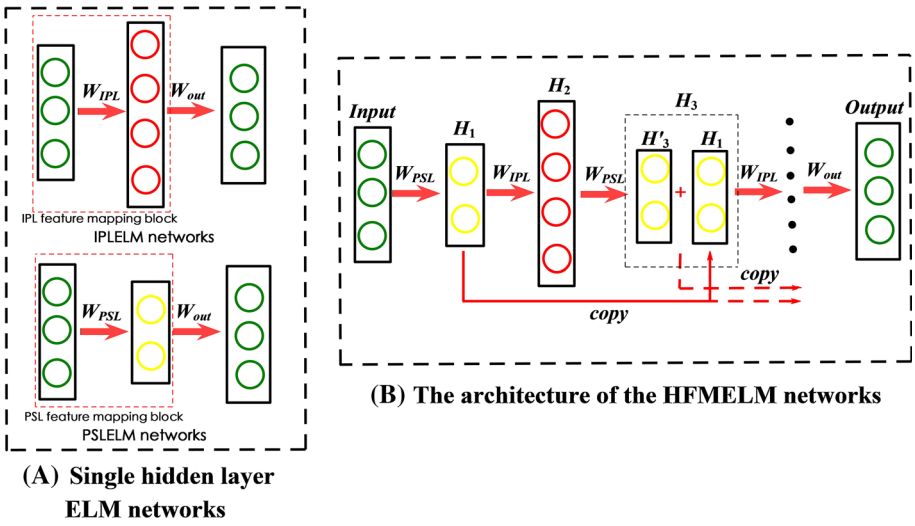

Fig. 2 Illustration of the architecture of the IPLELM, PSLELM, and HFMELM networks. A. Two kind of single hidden layer ELM networks, including IPLELM and PSLELM networks. We call the feature mapping from the input layer to hidden layer as IPL incorporated random feature mapping block (IPL feature mapping block or IPL-layer) and principal subspace learning block (PSL feature mapping block or PSL-layer) respectively. Note that W_{IPL} denotes the feature mapping of the IPLELM, or called as IPL-layer, while W_{PSL} denotes the feature mapping of the PSLELM network, or called as PSL-layer. B. The architecture of the HFMELM networks, which is composed of several stacked IPL feature mapping blocks and PSL feature mapping blocks

meanwhile, maximizing the information transmission by mapping the output to a desired distribution. The W_{PSL} denotes the feature mapping of the PSLELM, or called as PSL-layer. i.e., the connection weights are generated by the GHA method, and the outputs of the hidden neurons are then adapted by the IPL method. Hence, the PSL-layer maps the original data to its principal subspace, therefore extracting the principal component of the former layer. As illustrated in Fig. 2b, the HFMELM is composed of several feature mapping blocks. Due to the non-linearity of each feature mapping, the HFMELM has much better non-linearity fitting ability than those of single hidden layer ELMs. After the layer by layer feature mapping, the deep representations of the original data are obtained. It is worth noting that, as the networks going deeper, the information of the input could be vanish and "wash out" by the time it reaches the end of the networks [40]. Therefore, to ensure maximum information flow within the layer-wise feature mapping process, as can be seen from Fig. 2b, each PSL-layer (the second layer and fourth layer in Fig. 2) obtains additional inputs from all preceding PSL-layers and pass on to all subsequent PSL-layers. The effectiveness of the feature reusing in the deep networks has also been successfully verified in many models, such as Densely connected CNN [40] and Multi-Layer ELM [23]. As shown in Fig. 2b, the features output from the first PSL-layer $\boldsymbol{H}_1$ should be incorporated into the features output from the second PSL-layer $\boldsymbol{H}'_3$.

$$\boldsymbol{H}_3 = \boldsymbol{H}'_3 + \lambda \boldsymbol{H}_1 \tag{20}$$

where λ is the trade-off parameters between the two features. Note that when $\lambda \to 0$, the HFMELM networks becomes a traditional stacked model without feature reusing, when $\lambda \to \infty$, the HFMELM networks degrade into a PSLELM networks. The output layer of the HFMELM networks is the same as the traditional ELM networks. Compared with the single hidden layer ELM networks, the proposed HFMELM shows some favorable merits. On the

Algorithm 5 Training of HFMELM Networks

Input: Training data $\boldsymbol{x}$, num of hidden nodes $\boldsymbol{L} = \{L_1, L_2, L_3\}$, trade-off parameters λ and C.
Output: Connection weights of each feature mapping layer $\boldsymbol{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$, replicate matrix of slope and bias of each layer $\boldsymbol{A} = \{\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3\}$, $\boldsymbol{B} = \{\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{B}_3\}$, and output weights $\boldsymbol{\beta}$.
1: Obtain $\boldsymbol{w}_1$ according to **Algorithm 3**, and calculate $\boldsymbol{H}_1 = \boldsymbol{x}\boldsymbol{w}_1^T$.
2: Calculate and construct $\boldsymbol{A}_1$, $\boldsymbol{B}_1$, and adapt the output by $\boldsymbol{H}_1 = f(\boldsymbol{A}_1 \circ \boldsymbol{H}_1 + \boldsymbol{B}_1)$. (Refer to the **Algorithm 4**, Step4-Step6).
3: Randomly initialize $\boldsymbol{w}_2$ and calculate $\boldsymbol{H}_2 = \boldsymbol{H}_1\boldsymbol{w}_2^T$.
4: Calculate and construct $\boldsymbol{A}_2$, $\boldsymbol{B}_2$, and adapt the output by $\boldsymbol{H}_2 = f(\boldsymbol{A}_2 \circ \boldsymbol{H}_2 + \boldsymbol{B}_2)$.
5: Obtain $\boldsymbol{w}_3$ according to **Algorithm 3**, and calculate $\boldsymbol{H}_3 = \boldsymbol{H}_2\boldsymbol{w}_3^T$.
6: Calculate and construct $\boldsymbol{A}_3$, $\boldsymbol{B}_3$, and adapt the output by $\boldsymbol{H}_3 = f(\boldsymbol{A}_3 \circ \boldsymbol{H}_3 + \boldsymbol{B}_3)$.
7: Feature reusing $\boldsymbol{H}_3 = \boldsymbol{H}_3 + \lambda\boldsymbol{H}_1$.
8: Calculate the output weights $\boldsymbol{\beta} = (\frac{\boldsymbol{I}}{C} + \boldsymbol{H}_3^T\boldsymbol{H}_3)^{-1}\boldsymbol{H}_3^T\boldsymbol{Y}$.
9: **return** $\boldsymbol{W}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\beta}$.

one hand, with the layer-wise principal component extracting, the HFMELM is capable of capturing the high-level representation of the original data. On the other hand, the HFMELM is stacked layer by layer and trained independently. In this respect, one can stack the IPL-layer and PSL-layer in different ways. The complete training procedure of the HMFELM is summarized in Algorithm 5.

6 Performance Evaluation and Analysis

In this section, to evaluate the effectiveness of our proposed methods, we compare the classification performance of our proposals with the traditional regularized ELM, IPLELM and the CELM [19] in various tasks, including four widely used face recognition databases, handwritten digits recognition (mnist) and object recognition (cifar-10). It is worth noting that the HFMELM networks evaluated in the following experiments is mainly shows in Fig. 2b. It has three hidden layers, a total of 5 layers altogether with the input layer and output layer, specifically, it has two PSL feature mapping layers and one IPL feature mapping layer, i.e. Input-layer $\rightarrow$ PSL-layer $\rightarrow$ IPL-layer $\rightarrow$ PSL-layer $\rightarrow$ Output-layer. Moreover, the two PSL feature mapping layers have the same number of hidden nodes (both are range from 50 to 1000) and the number of neurons in the IPL layer is set to be 10000 in all the experiments. In all methods, the input weights and bias are initialized in the same way, all of them are randomly initialized in the range of $(-1,1)$, and the number of iterations T_{max} in the algorithm 3 is set as 20 in all experiments. Regarding the optimal hyper-parameters, they are determined by applying multiple experiments using grid search strategy. Specifically, the trade-off coefficient C are finely tuned in the range of {0.001, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 1000}, and the trade-off parameters λ in the HFMELM is searched in the range of {0, 0.1, 0.2, 0.3,$\cdots$,0.9, 1, 2, 10} and set to be 0.5 at last. Note that the optimal coefficient C may vary with the number of hidden nodes. During experiments, the hyper-parameters of different methods are fixed to be the optimal values to get the best results. In all the following simulations, the testing environments are listed as below: Intel E5-2609 with 256G memory, CentOS Linux 7, python 2.7.

6.1 Experiments in Face Recognition Tasks

Face image databases used in our experiments include the following four face databases, namely ORL face database, Yale face database, extended Yale B face database and the

Table 1 Basic information of the benchmarking datasets

Datasets	Num of images	Num of classes	Cropped image size	Training mode
ORL	400	40	64×64	10-fold cv
Yale	165	100	100×100	10-fold cv
YaleB	5760	38	96×84	10-fold cv
AR	2600	100	124×90	10-fold cv
Mnist	60,000	10	28×28	Train/test
Cifar-10	60,000	10	$32 \times 32 \times 3$	Train/test

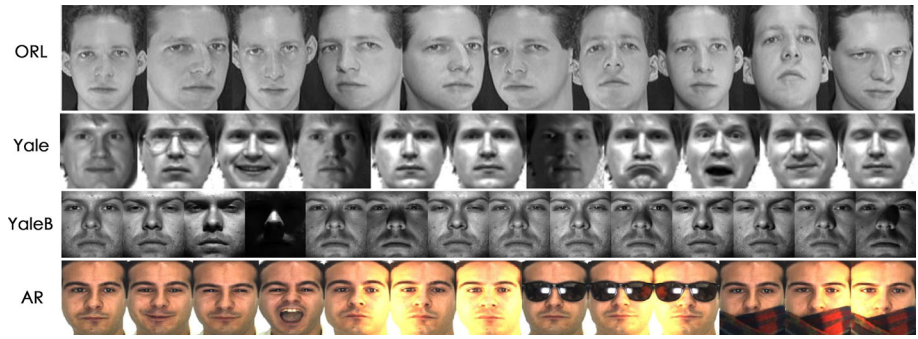

Fig. 3 Selected facial images from the four face database. From top to bottom: ORL face database, Yale face database, extend Yale face B (YaleB) database and cropped AR database respectively

cropped AR face database. The basic information of all the databases is listed in Table 1. Besides, a series of facial images from each database have been displayed in Fig. 3. The ORL face database consists of 400 face images of 40 distinct subjects with 10 images for each, the images were taken at different times, varying the lighting, facial expression and facial details, moreover, all the images were taken against a dark homogeneous background with the subjects in an upright, frontal position. The Yale face database contains 165 gray-scale individuals with 11 images for per subject, one per different facial expression or configuration: center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised, and wink. The extended Yale face database B, which is referred to as YaleB for ease of notation, contains 5760 single light source images of 10 individuals each seen under 576 viewing conditions (9 poses $\times$ 64 illumination conditions). The cropped AR face database consists of 2600 frontal images of 100 subjects (50 males and 50 females), which contain different facial expressions, illumination conditions (no additional light, additional left light, additional right light, additional two lights) and occlusions (sun glasses, make-up and scar). In order to reduce the dimension of the input data and increase the computation speed, all the color face images have been converted to grayscale and resized to the suitable size shown as Table 1.

For each face database, we performed 10-fold cross validation classification. Before classification, all the samples have been normalized to be of zero means and unit variance. Besides, for all the experiments, the Relu function is used as the activation function, and weber distribution is used as the target distribution in the IPL process. The number of the hidden nodes is selected from 50 to 1000 (50, 80, 100, 200, 300, 400, 500, 1000). In order

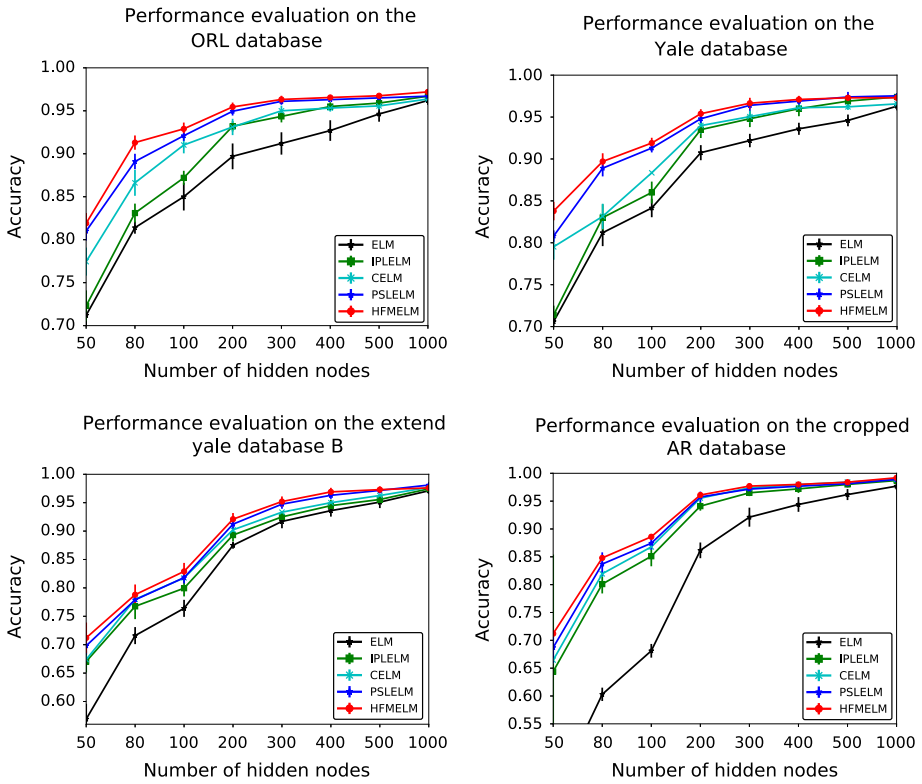

Fig. 4 Performance evaluation of our proposed methods on the four face databases

to get a stable and reliable classification accuracy, every experiment is executed 10 times for each database, and the average result and its standard deviation are illustrated in the Fig. 4.

As shown in the Fig. 4, for each database, the average accuracy as well as the standard deviation are presented. It should be noted that, for the experiment conducted on the YaleB and the cropped AR database, the actual standard deviation is too small to be seen. For ease of comparison, the standard deviation reported in the figure has been amplified for a same scale. As can be seen, it is obviously that both the proposed PSLELM and HFMELM almost consistently outperform the IPLELM and CELM, and significantly outperform the traditional regularization ELM by a large margin, especially in the case that the number of hidden nodes is small. In the meantime, the proposed methods achieve a much more stable classification accuracy than the traditional ELM networks, which can be reflected by the standard deviation. Thereby, we can draw the conclusion that the performance of the ELM networks can be significantly improved with the incorporation of the principal subspace learning and the intrinsic plasticity learning. Further more, in spite of the fact that the HFMELM is very simple and easy to achieve, it performs best as presented.

6.2 Performance Evaluation on mnist and cifar-10 Database

In the last subsection, we mainly evaluated the proposed methods on several facial recognition databases. In this section, we also evaluated the proposed methods on handwritten digit

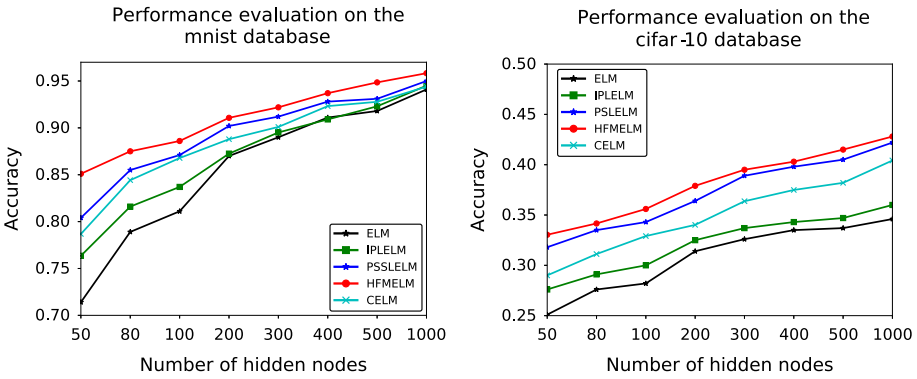

Fig. 5 Performance evaluation of the proposed methods on the mnist dataset (left) and the cifar-10 dataset (right). Note that the feature dimension of the Cifar-10 dataset (raw pixel) used in our experiments is 3072. Hence, either the traditional ELM networks with hundreds of hidden nodes or the SVM and other classifiers can hardly achieve a satisfactory performance on the samples with such a high-dimensional input space

recognition task (mnist) and objection recognition task (cifar-10), which are widely used for the performance evaluation of deep learning related algorithm. The mnist dataset consists of 60,000 training images and 10,000 test images. All the samples are digits $0-9$ in a total of 10 classes, with the size of 28×28. The cifar-10 dataset contains 60,000 color images in 10 classes, in which the training set and test set consist of 50,000 images and 10,000 images respectively. Due to the large number of the training samples, a long time and large memory space is required to compute the pseudo inverse. Thus, we randomly select 30,000 samples for training, and 10,000 samples for testing.

Similar to the experiments conducted on the facial database, the raw images are vectorized first and preprocessed to be of zero means and unit variance, and the number of hidden nodes is range from 50 to 1000. Due to the vast amount of training samples and test samples, we adopt the train/test classification model to evaluate the performance. As can be seen in the Fig. 5, either on the mnist dataset or on the cifar-10 dataset, the proposed PSLELM and HFMELM consistently outperform the baselines by a large margin. Especially in the cifar-10 dataset, the HFMELM outperforms the tradition ELMs by almost 7 percentage points on average, moreover, the HFMELM with 50 hidden nodes attains the approximate accuracy with the traditional ELM with 1000 hidden nodes.

7 Conclusions

Due to the random initialization of the input weights, the performance of the ELM networks has been greatly limited. In this paper, we attempt to optimize the connection weights between the input layer and the hidden layer. In this respect, we adopt the principal subspace learning approach to generate a much more discriminative feature space. Besides, the information transmission from the input layer to the hidden layer has been maximized with the integration of the intrinsic plasticity learning. It has been shown that, with the help of the PSL and IPL methods, the proposed PSLELM networks significantly outperforms the classic ELM. In addition, based on the PSL and IPL approach, a novelty stacked ELM networks (HFMELM) is further proposed, which is composed of several IPL feature mapping blocks and PSL feature mapping blocks. Without any information loss in the layer-wise feature mapping

process, the HFMELM networks combines the low-level features and high-level features of the original data and achieves better performance.

Acknowledgements This research is supported by the National Science and Technology Major Projects (No. 2013ZX03005013), and the Opening Foundation of the State Key Laboratory for Diagnosis and Treatment of Infectious Diseases (No. 2014KF06).

References

1. Leshno M, Lin VY, Pinkus A, Schocken S (1993) Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw 6(6):861–867
2. Huang G-B, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9(1):224–229
3. Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In 2004 IEEE international joint conference on proceedings neural networks, vol 2. IEEE, pp 985–990
4. Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B (Cybern) 42(2):513–529
5. Liu X, Wang L, Huang G-B, Zhang J, Yin J (2015) Multiple kernel extreme learning machine. Neurocomputing 149:253–264
6. Huang G, Song S, Gupta JN, Wu C (2014) Semi-supervised and unsupervised extreme learning machines. IEEE Trans Cybern 44(12):2405–2417
7. Tang J, Deng C, Huang G-B (2016) Extreme learning machine for multilayer perceptron. IEEE Trans Neural Netw Learn Syst 27(4):809–821
8. Liang N-Y, Huang G-B, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Networks 17(6):1411–1423
9. Mirza B, Lin Z, Toh K-A (2013) Weighted online sequential extreme learning machine for class imbalance learning. Neural Process Lett 38(3):465–486
10. Mirza B, Kok S, Dong F (2016) Multi-layer online sequential extreme learning machine for image classification. In Proceedings of ELM-2015 Volume 1. Springer, Berlin pp 39–49
11. Zong W, Huang G-B (2014) Learning to rank with extreme learning machine. Neural Process Lett 39(2):155–166
12. Zong W, Huang G-B, Chen Y (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing 101:229–242
13. Cao J, Lin Z, Huang G-B, Liu N (2012) Voting based extreme learning machine. Inf Sci 185(1):66–77
14. Liu N, Wang H (2010) Ensemble based extreme learning machine. IEEE Signal Process Lett 17(8):754–757
15. Iosifidis A, Tefas A, Pitas I (2013) Minimum class variance extreme learning machine for human action recognition. IEEE Trans Circuits Syst Video Technol 23(11):1968–1979
16. Kasun LLC, Yang Y, Huang G-B, Zhang Z (2016) Dimension reduction with extreme learning machine. IEEE Trans Image Process 25(8):3906–3918
17. Iosifidis A, Tefas A, Pitas I (2016) Graph embedded extreme learning machine. IEEE Trans Cybern 46(1):311–324
18. Nguyen TV, Mirza B (2017) Dual-layer kernel extreme learning machine for action recognition. Neurocomputing 260:123–130
19. Zhu W, Miao J, Qing L (2014) Constrained extreme learning machine: a novel highly discriminative random feedforward neural network. In 2014 international joint conference on neural networks (IJCNN). IEEE, pp 800–807
20. Niu P, Ma Y, Li M, Yan S, Li G (2016) A kind of parameters self-adjusting extreme learning machine. Neural Process Lett 44(3):813–830
21. Huang G-B, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(16–18):3460–3468
22. Iosifidis A, Tefas A, Pitas I (2015) Dropelm: Fast neural network regularization with dropout and dropconnect. Neurocomputing 162:57–66
23. Yu W, Zhuang F, He Q, Shi Z (2015) Learning deep representations via extreme learning machines. Neurocomputing 149:308–315
24. Zhou H, Huang G-B, Lin Z, Wang H, Soh YC (2015) Stacked extreme learning machines. IEEE Trans Cybern 45(9):2013–2025

25. Li G, Niu P, Ma Y, Wang H, Zhang W (2014) Tuning extreme learning machine by an improved artificial bee colony to model and optimize the boiler efficiency. Knowl-Based Syst 67:278–289
26. Han F, Yao H-F, Ling Q-H (2013) An improved evolutionary extreme learning machine based on particle swarm optimization. Neurocomputing 116:87–93
27. Cao J, Lin Z, Huang G-B (2012) Self-adaptive evolutionary extreme learning machine. Neural Process Lett 36(3):285–305
28. Neumann K, Steil JJ (2011) Batch intrinsic plasticity for extreme learning machines. In International conference on artificial neural networks. Springer, Berlin pp 339–346
29. Klaus Steil J (2013) Optimizing extreme learning machines via ridge regression and batch intrinsic plasticity. Neurocomputing 102:23–30
30. Sanger TD (1989) Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Netw 2(6):459–473
31. Johnson WB, Lindenstrauss J (1984) Extensions of lipschitz mappings into a hilbert space. Contemp Math 26(189–206):1
32. Li C (2011) A model of neuronal intrinsic plasticity. IEEE Trans Auton Ment Dev 3(4):277–284
33. Triesch J (2005) Synergies between intrinsic and synaptic plasticity in individual model neurons. Adv Neural Inf Process Syst 1417–1424
34. Schrauwen B, Wardermann M, Verstraeten D, Steil JJ, Stroobandt D (2008) Improving reservoirs using intrinsic plasticity. Neurocomputing 71(7–9):1159–1171
35. Hebb DO (2005) The organization of behavior: a neuropsychological theory. Psychology Press, Hove
36. Oja E, Karhunen J, Wang L, Vigario R (1996) Principal and independent components in neural networks-recent developments. Proceedings VII Italian Workshop Neural Networks WIRN 95:16–35
37. Karhunen J, Joutsensalo J (1995) Generalizations of principal component analysis, optimization problems, and neural networks. Neural Netw 8(4):549–562
38. Triesch J (2014) Synergies between intrinsic and synaptic plasticity mechanisms. Neural Comput 19(4):885–909 s
39. Schlkopf B, Platt J, Hofmann T (2006) Greedy layer-wise training of deep networks. In: International conference on neural information processing systems, pp 153–160
40. Huang G, Liu Z, Weinberger KQ, van der Maaten L (2017) Densely connected convolutional networks. Proc IEEE Conf Comput Vis Pattern Recognit 1(2):3