Extreme learning machine with affine transformation inputs in an activation function

Cao, Jiuwen; Zhang, Kai; Yong, Hongwei; Lai, Xiaoping; Chen, Badong; Lin, Zhiping

2018

Cao, J., Zhang, K., Yong, H., Lai, X., Chen, B., & Lin, Z. (2019). IEEE Transactions on Neural Networks and Learning Systems, 30(7), 2093-2107. doi:10.1109/TNNLS.2018.2877468

https://hdl.handle.net/10356/136684

https://doi.org/10.1109/TNNLS.2018.2877468

Extreme Learning Machine with Affine Transformation Inputs in an Activation Function

Jiuwen Cao, *Member, IEEE,* Kai Zhang, Hongwei Yong, Xiaoping Lai, *Member, IEEE*
Badong Chen, *Senior Member, IEEE* and Zhiping Lin, *Senior Member, IEEE*

***Abstract*—The extreme learning machine (ELM) has attracted much attention over the past decade due to its fast learning speed and convincing generalization performance. However, there still remains a practical issue to be approached when applying the ELM: the randomly generated hidden node parameters without tuning can lead to the hidden node outputs being nonuniformly distributed, thus giving rise to poor generalization performance. To address this deficiency, a novel activation function with an affine transformation on its input is introduced into the ELM, which leads to an improved ELM algorithm that is referred to as an affine transformation ELM (AT-ELM) in this paper. The scaling and translation parameters of the affine transformation activation function are computed based on the maximum entropy principle in such a way that the hidden layer outputs approximately obey a uniform distribution. Application of the AT-ELM algorithm in nonlinear function regression shows its robustness to the range scaling of the network inputs. Experiments on nonlinear function regression, real-world dataset classification and benchmark image recognition demonstrate better performance for the AT-ELM compared with the original ELM, the regularized ELM and the kernel ELM. Recognition results on benchmark image datasets also reveal that the AT-ELM outperforms several other state-of-the-art algorithms in general.**

***Index Terms*—Extreme learning machine, Affine transformation activation function, Maximum entropy, Regression, Classification**

I. Introduction

ARTIFICIAL neural networks with random parameters were studied many years ago [1]. Recently, the extreme learning machine (ELM) has been developed for feedforward neural networks (FNN) and has become popular due to its fast learning speed, ease of implementation and good generalization performance [2–4]. The ELM was originally proposed to learn a single hidden layer FNN

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

using a training dataset $\{(\mathbf{x}_j, \mathbf{t}_j)\}_{j=1}^{N}$, with $\mathbf{x}_j \in R^m$ and $\mathbf{t}_j \in R^d$ being the j-th input sample and the target output associated with $\mathbf{x}_j$, respectively. In (1), $g(\cdot)$ denotes the activation function, $\mathbf{w}_i \in R^{1\times m}$ and $b_i \in R$ are the input weight and bias of the i-th hidden node, $\boldsymbol{\beta}_i \in R^d$ is the weight vector that connects the i-th hidden node to the output nodes, and $\mathbf{o}_j \in R^d$ is the network output for the j-th input sample. For the convenience of presentation, we call $\{(\mathbf{w}_i, b_i)\}_{i=1}^{L}$ the hidden node parameters in this paper. ELM theory states that the hidden node parameters can be randomly assigned, independent of the applications, and tuning-free [2]. The network training becomes solving a system of linear equations.

The ELM has been extensively studied in terms of its theory [5–8], algorithms [9–15] and applications [16–27]. Although significant achievements have been made, little attention has been paid to the distribution of hidden layer outputs of the ELM, which is essential for network learning. Unlike the ELM, neural networks trained with gradient descent-based algorithms have benefited from studies on data distributions of the network layers, especially in deep network learning [28–31]. The training becomes slow and complicated if the data distribution of each layer changes during learning [31]. Mending the distributions of data across all layers over time not only accelerates the training by saving the cost of parameter adaption to new distributions but also improves the performance [31]. A similar observation states that whitening each layer's inputs guarantees a fast convergence speed [28]. To address the data distribution issue, a number of methods have been developed for deep neural networks (DNNs), including whitening activations [30], batch normalization [31], weight normalization [32], and normalization propagation [33].

Recent studies show that the ELM can be plagued by input data scaling [34]. The good generalization performance is only achievable within an order of magnitude on the input range scale [34], which limits the practical applications of the ELM. The reason is that the input data scaling changes the distribution of the hidden layer net inputs $v = \mathbf{w} \cdot \mathbf{x} + b$ and the resultant hidden node outputs might not be uniformly distributed. Random hidden node parameters without tuning in the ELM could move many of the hidden layer net inputs v into the saturated regime or the linear regime (close to a

J. Cao and X. Lai are with Key Lab for IOT and Information Fusion Technology of Zhejiang, Hangzhou Dianzi University, Zhejiang, 310018, China and J. Cao is with School of Electrical, Information and Media Engineering, University of Wuppertal, 42119 Wuppertal, Germany, E-mail: jwcao@hdu.edu.cn, laixp@hdu.edu.cn.

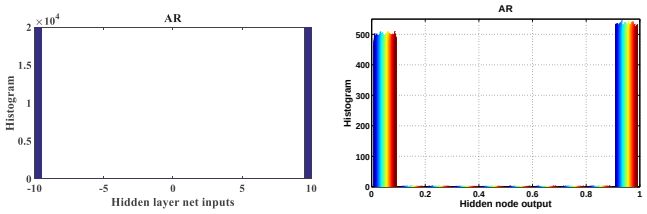
K. Zhang is with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China, E-mail: cskaizhang@gmail.com.

H. Yong is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, E-mail: cshyong@comp.polyu.edu.hk.

B. Chen is with School of Electronic and Information Engineering, Xian Jiaotong University, Xian, 710049, China, E-mail: chenbd@mail.xjtu.edu.cn.

Z. Lin is with School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, E-mail: ezplin@ ntu.edu.sg. Corresponding author with phone no. +65 6790 6857.

This work was supported by the NNSF of China (61503104, 61573123, 91648208) and by the K. C. Wong Education Foundation and DAAD.



(a) Inputs (b) Outputs

Fig. 1. Histograms of hidden node net inputs and outputs for the AR dataset by an ELM.

linear function) of the activation function. This circumstance can lead to poor generalization performance of the network. For illustration, the histograms of the hidden node net inputs v of the AR face database[1] and the associated outputs obtained by the ELM are depicted in Fig. 1 (a) and (b), respectively. The Sigmoid $g(v) = \frac{1}{1+e^{-v}}$ is used as the activation function. As depicted, the hidden node net inputs v are mostly distributed in the region $|v| \geq 10$, and hence, the resultant outputs $g(v)$ overly concentrate around 0 and 1, i.e., the saturated regimes of the Sigmoid.

Existing DNNs usually employ an activation function with nonsaturating nonlinearity to alleviate the aforementioned deficiency of the saturated regime in conventional activation functions, such as the Rectifier Linear Unit (ReLU) [35] and its improvements, including the leaky ReLU [36], parametric ReLU (PReLU) [37] and randomized ReLU (RReLU) [38]. They are effective in speeding up the training and enhancing the performance. The idea of using an activation function with a learnable parameter that is adaptively updated during the training is introduced in PReLU [37], where different rather than fixed activation functions are suggested for different channels in a DNN. However, existing DNNs have the following drawbacks: 1) the optimization of the hidden node parameters relies on the backward propagation (BP) algorithm, which usually suffers from having very slow convergence; 2) the data distribution characteristics of the hidden layers are not well studied for parameter optimization of the activation functions.

To address the aforementioned deficiencies in the existing ELMs, a novel activation function with an affine transformation input is introduced in this paper. The resulting improved ELM algorithm is referred to as an affine transformation ELM (AT-ELM). The scaling and translation parameters of the affine transformation in the activation function, or in short, the affine parameters, can be computed based on the maximum entropy principle such that the distribution of the hidden layer outputs approximates a uniform distribution after mapping. In general, however, there is no closed form solution for the optimal affine parameters. In this paper, a gradient descent-based iterative algorithm is used to optimize the affine parameters. In the special case with zero mean Gaussian hidden layer net inputs, an empirical closed form for estimating the two affine parameters is obtained using the maximum entropy principle. The main contributions of this paper are summarized as follows:

[1]Detailed descriptions on the database are given in Section IV-C

1) An affine transformation is introduced into the activation function. With the affine transformation, the proposed AT-ELM algorithm can adapt its activation function to the distribution of the hidden layer net inputs, and thus, to avoid the issue of having a saturated or linear regime, lead to a robust generalization performance.
2) Estimation schemes for optimal parameters of the affine transformation are developed. With these schemes, the hidden layer outputs in the AT-ELM approximately comply with a uniform distribution, and thus, they have the maximum entropy and contain more information than the nonuniformly distributed hidden layer outputs in the other ELMs.
3) Experiments and comparisons are conducted on many benchmark problems and datasets. The results show that the AT-ELM-based methods have promising performance better than the original ELM, regularized ELM (RELM) [39], RELM_{LOO} (LOO stands for the leave-one-out cross-validation strategy [39]), KELM [3], and several other state-of-the-art algorithms. In addition, employing orthogonal hidden node parameters in AT-ELMs usually improves the generalization performance further.

II. Brief Review of the ELM

A brief review on the ELM is first presented in this section. Then, the deficiencies that arise from having a fixed activation function are discussed.

A. ELM

The ELM was developed to train single hidden layer feed-forward neural networks (SLFN) [2]. The network model (1) can be rewritten using matrix multiplication as follows:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{O}, \tag{2}$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, ..., \boldsymbol{\beta}_L]^T$ and $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_N]^T$ are the output layer weight matrix and network output matrix, $\mathbf{H}$ is the hidden layer output matrix defined by [2]

$$\mathbf{H}(\mathbf{w}_1, \cdots, \mathbf{w}_L, b_1, \cdots, b_L) = G(\mathbf{V}). \tag{3}$$

Here, $G(\mathbf{V})$ is an $N \times L$ matrix, in which the element in the j-th row and i-th column is $g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i)$. $\mathbf{V}$ is the matrix of hidden layer net inputs for all of the training samples

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_N^T \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1 \cdot \mathbf{x}_1 + b_1 & \cdots & \mathbf{w}_L \cdot \mathbf{x}_1 + b_L \\ \vdots & \ldots & \vdots \\ \mathbf{w}_1 \cdot \mathbf{x}_N + b_1 & \cdots & \mathbf{w}_L \cdot \mathbf{x}_N + b_L \end{bmatrix}_{N \times L}. \tag{4}$$

The ELM aims to minimize the training error, i.e.,

$$\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_F^2, \tag{5}$$

where $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_N]^T$ is the target output matrix and $\|\cdot\|_F$ is the the Frobenius norm of the error matrix $\mathbf{H}\boldsymbol{\beta} - \mathbf{T}$. For the case with single output, the error matrix reduces to a vector and the Frobenius norm reduces to the 2-norm of the error vector.

It is stated in [2] that for any infinitely differentiable activation function and randomly generated hidden node parameters with any continuous probability distribution, the SLFN with $L = N$ hidden nodes can approximate N distinctive samples without error with a probability of one. The output weight is thus determined by finding the least-squares solution of the linear equations. The well recognized universal approximation capability of ELM shows that with any bounded nonconstant piecewise continuous hidden activation function and any randomly generated hidden weights and biases, the SLFN output can approximate any continuous function $f(\mathbf{x})$ with appropriate output weights $\boldsymbol{\beta}$ in the sense that $\lim_{L\to\infty} \left\| \sum_{i=1}^{L} \boldsymbol{\beta}_i g(\mathbf{w}_i \cdot \mathbf{x} + b_i) - f(\mathbf{x}) \right\| = 0$. A number of nonlinear activation functions, including Sigmoid, Sine and Radial Basis Function (RBF), are shown to be effective for the ELM.

Following Bartlett's statement that for FNNs, the smaller the output weight's norm is, the better the generalization performance will be [40], RELM minimizes the training error and the norm of the output weight matrix as follows [39]:

$$\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_F^2 + \lambda \|\boldsymbol{\beta}\|_F^2, \tag{6}$$

where λ is the regularization parameter that controls the tradeoff between the two terms in the cost. According to whether the number of training samples is greater than the number of hidden nodes, the closed-form solution of (6) is given as [3]

$$\hat{\boldsymbol{\beta}} = (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T\mathbf{T}, \tag{7}$$

where $\mathbf{I}$ is the identity matrix. When $L > N$, a more computational efficient solution is

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1}\mathbf{T}. \tag{8}$$

B. Deficiencies from having fixed activation functions in the ELM

The activation function is one of the crucial factors in neural networks; it brings nonlinearity into the networks. While using random hidden node parameters in the ELM brings a fast training speed and easy implementation advantages, using a fixed activation function also leads to a deficiency in the ELM. It has been long known that the choice of the activation function is determined by the distributions of the input samples and targets [41]. In the ELM, the hidden node parameters are independent of the input data, and the activation function is fixed during learning. Relying merely on solving the output weights through linear equations could lead to poor generalization performance for irregular distributions, such as the large data scaling mentioned in Section I or the data shift investigated in [34]. It was shown that the *ELM is quite sensitive to the range of input variables*.

For illustration, we take the most frequently used activation function in the ELMs, the Sigmoid, as an example here. The Sigmoid function is popular for its good interpretation as the firing rate of a node, but it recently has fallen out of favor in deep learning (DL) due to its saturated ends and its linear regime [31, 37]. Hence, the ReLU function and its variants have been developed in DL. Let v_{ij} ($i = 1, 2, \cdots, L$ and $j = 1, 2, \ldots, N$) be the i-th element of the hidden layer net input vector $\mathbf{v}_j = \mathbf{W}\mathbf{x}_j + \mathbf{b}$ for the j-th input $\mathbf{x}_j$, where $\mathbf{W} = [\mathbf{w}_1^T, \mathbf{w}_2^T, ..., \mathbf{w}_L^T]^T$ is the hidden node weight matrix, and $\mathbf{b} = [b_1, b_2, ..., b_L]^T$ denotes the bias vector. The Sigmoid function maps $\mathbf{v}_j$ into its output range. If $|v_{ij}|$ is too large, then the output will fall into the saturated regime of the function at either tail, 0 or 1. The gradient $g'(v_{ij})$ then becomes zero, and the v_{ij}-associated input weights and biases cannot be updated using the gradient-based method. In addition, if most of the hidden layer net inputs are located in the region that corresponds to an output range $[0.2, 0.8]$ of the Sigmoid function, then the hidden nodes would be approximately be linear, and thus, the network would lose its universal approximation capability [31]. Moreover, $\mathbf{H}$ could also suffer the problem of having a large condition number if the hidden layer net inputs overly concentrate in any small region or at any fixed value.

III. The Proposed AT-ELM Algorithm

In this paper, we propose a novel ELM algorithm that has affine transformation inputs in the activation function, which is abbreviated as AT-ELM. Rather than tuning the input weights and hidden biases, a novel affine transformation activation is applied into the ELM to adapt to the distribution of hidden layer net inputs through adjusting its affine parameters based on the principle of maximum entropy [42]. This approach, therefore, not only preserves the fast learning speed of the ELM by avoiding the iterative updating of hidden weights and biases, but also attains the largest entropy of the hidden layer output data by enforcing the data to be approximately uniformly distributed in the desired output range of the activation function.

A. Information entropy of the hidden layer output data

In information theory, entropy is a measure of the unpredictability of the information content. It is defined as the expected value of the information contained in an event. The principle of maximum entropy states that *subject to precisely stated prior data, the probability distribution that best represents the current state of knowledge is the one with the largest entropy* [42]. Suppose that the probability of the state x_i for a discrete random variable $\mathcal{X}$ is $p(\mathcal{X} = x_i) = p(x_i)$; then the entropy of $\mathcal{X}$ is

$$\mathcal{H}[p] = \sum_i p(x_i) \log_2 \frac{1}{p(x_i)}, \tag{9}$$

where the logarithm function is a measure of the information content. Maximizing the entropy $\mathcal{H}[p]$ subject to the probability constraint can be solved by a Lagrangian multiplier method, with the Lagrangian function defined by

$$\mathcal{L} = \sum_i p(x_i) \log_2 \frac{1}{p(x_i)} + \mu \left(\sum_i p(x_i) - 1 \right). \tag{10}$$

Maximizing (10) leads to $p(x_i) = \frac{1}{M}$, where M is the total number of states [41]. This circumstance implies that

a uniformly distributed random variable has the maximum entropy [41].

In the ELM, however, because all hidden node parameters are randomly generated, independent of the training data and tuning-free during the training process, the hidden layer outputs are unlikely to obey a uniform distribution. For illustration, we investigate the variance of the hidden layer net inputs in the ELM. Assuming that the elements in $\mathbf{x}$ share the same distribution and are independently and identically distributed (*i.i.d.*), and the elements in $\mathbf{W}$ and $\mathbf{b}$ are usually generated according to a uniform distribution $\mathcal{U}[-1,1]$ or standard normal distribution $\mathcal{N}(0,1)$ and are all mutually independent, the variance $var(\cdot)$ of the hidden layer net inputs can be expressed as

$$var(v_i) = L \cdot var(w_i) E\left(x_i^2\right) + var(b_i), \tag{11}$$

where v_i, w_i, x_i and b_i denote the random variables of the elements in $\mathbf{V}$, $\mathbf{W}$, $\mathbf{x}$ and $\mathbf{b}$ (the derivation of (11) is given in Appendix A). Here, the expectation of input x_i is assumed to be nonzero, which is reasonable for most practical applications (one can verify that the result also holds for zero mean input x_i). If w_i and b_i obey a uniform distribution $\mathcal{U}[-1,1]$, (11) reduces to $var(v_i) = \frac{L}{3} \cdot \left(E\left(x_i^2\right) + 1\right)$. If w_i and b_i satisfy the standard normal distribution, $var(v_i) = L \cdot \left(E\left(x_i^2\right) + 1\right)$. Therefore, the variance of the hidden layer net inputs depends on the number of hidden nodes L and the second-order moment $E\left(x_i^2\right)$ of the network inputs. One can check that the hidden layer output data are unlikely to obey a uniform distribution within the output range of the activation function in ELM because the variance is related to the variance of the inputs and the number of hidden nodes. Thus, the hidden layer output data do not have the maximum entropy in general. To address this issue, we design an activation function with an affine transformation on its input such that after mapping by the new activation function, the hidden layer outputs are approximately uniformly distributed in the output range of the activation function. Hence, the issue of having a saturated regime in ELM can be avoided.

B. Activation function with affine transformation inputs

As mentioned above, in AT-ELM, we maximize the entropy of the hidden layer output data by adjusting the scaling and translation parameters s and t of the affine transformation activation function. Given an activation function $g(\cdot)$ with a desired output range $\mathbb{D} \triangleq: [\mathsf{o}_\ell, \mathsf{o}_u]$ and letting the probability density function (PDF) of the hidden layer net inputs be $p_{\mathbf{v}}(v)$, the objective is to find a proper (s,t) pair such that the net inputs of the hidden layer are mapped into uniformly distributed outputs in $\mathbb{D}$ by $g(s \cdot v + t)$. Specifically, the postactivation value $y = g(s \cdot v + t)$ is required to satisfy the following constraints:

$$E(y) = \frac{\mathsf{o}_\ell + \mathsf{o}_u}{2}, \tag{12}$$

$$var(y) = \frac{\left(\mathsf{o}_u - \mathsf{o}_\ell\right)^2}{12}, \tag{13}$$

where y denotes the random variable of the activation function output, which is also called the hidden layer output [3]. In other words, it requires

$$\int_{-\infty}^{\infty} y p_{\mathbf{y}}(y) dy = \frac{\mathsf{o}_\ell + \mathsf{o}_u}{2}, \tag{14}$$

$$\int_{-\infty}^{\infty} y^2 p_{\mathbf{y}}(y) dy = \frac{\left(\mathsf{o}_u - \mathsf{o}_\ell\right)^2}{12} + \frac{\left(\mathsf{o}_\ell + \mathsf{o}_u\right)^2}{4}, \tag{15}$$

where $p_{\mathbf{y}}(y)$ represents the PDF of y. The PDF $p_{\mathbf{y}}(y)$ can be obtained from the well-known truth that for a continuous random variable $\mathbf{x}$ with PDF $f_{\mathbf{x}}(x)$, the PDF of element y in the random variable $\mathbf{y} = g(x)$ satisfies the following [43]:

$$f_{\mathbf{y}}(y) = \begin{cases} f_{\mathbf{x}}\left(g^{-1}(y)\right) \left|\frac{dg^{-1}(y)}{dy}\right|, & y \in (y_l, y_u), \\ 0, & elsewhere, \end{cases} \tag{16}$$

where $g(\cdot)$ is monotonic and derivable in the domain of definition, the support set of $f_{\mathbf{x}}(x)$ is assumed to be $[x_{\min}, x_{\max}]$, and $y_l = \min\{g(x_{\min}), g(x_{\max})\}$ and $y_u = \max\{g(x_{\min}), g(x_{\max})\}$ correspond to the minimum and maximum in the output range. Given an affine transformation activation function $y = g(s \cdot v + t)$, let $g^{-1}(\cdot)$ be the inverse of $g(\cdot)$. The inverse of $y = g(s \cdot v + t)$ can be expressed as $v = \frac{1}{s} g^{-1}(y) - \frac{t}{s}$. Hence, with (14), (15) and (16), it can be obtained that (s,t) satisfies

$$\int_{-\infty}^{\infty} y p_{\mathbf{v}}\left(\frac{1}{s} g^{-1}(y) - \frac{t}{s}\right) \left|\frac{d\left(\frac{1}{s} g^{-1}(y) - \frac{t}{s}\right)}{dy}\right| dy = \frac{\mathsf{o}_\ell + \mathsf{o}_u}{2},$$

$$\int_{-\infty}^{\infty} y^2 p_{\mathbf{v}}\left(\frac{1}{s} g^{-1}(y) - \frac{t}{s}\right) \left|\frac{d\left(\frac{1}{s} g^{-1}(y) - \frac{t}{s}\right)}{dy}\right| dy = \frac{\mathsf{o}_\ell^2 + \mathsf{o}_u^2 + \mathsf{o}_\ell \mathsf{o}_u}{3}.$$

Unfortunately, from the above equations, it is in general not possible to derive closed-form solutions for the affine parameters s and t. Therefore, estimation algorithms for the affine parameters will be developed in the next subsection. Before presenting the algorithms, we assume that the training and testing data have the same distribution, which is a fundamental assumption in supervised learning.

C. Parameter estimation of the affine transformation activation function

Given a training dataset $\{(\mathbf{x}_j, \mathbf{t}_j)\}_{j=1}^N$, let $\mathbf{V}$ be the hidden layer net input matrix for all of the training samples, as denoted in (4), and let $g(\cdot)$ represent the activation function. We concatenate all of the elements of $\mathbf{V}$ into an $(N \cdot L)$-dimensional vector $\boldsymbol{\nu}_{asc}$ in ascending order as $\boldsymbol{\nu}_{asc} = [\nu_1, \ldots, \nu_k, \ldots, \nu_{N \cdot L}]^T$, where $\nu_1 \leq \ldots \leq \nu_k \leq \ldots \leq \nu_{N \cdot L}$. To ensure that the elements in $g(s \cdot \boldsymbol{\nu}_{asc} + t \cdot \mathbb{1})$ are approximately uniformly distributed in $\mathbb{D} \triangleq: [\mathsf{o}_\ell, \mathsf{o}_u]$, s and t are estimated by

$$\min_{s,t} \|g(s \cdot \boldsymbol{\nu}_{asc} + t \cdot \mathbb{1}) - \hat{\mathbf{o}}\|_2^2, \tag{17}$$

where $\hat{\mathbf{o}}$ is an $(N \cdot L)$-dimensional vector whose elements are uniformly sampled in the output range $\mathbb{D}$, and $\mathbb{1} = [1, \ldots, 1]^T \in R^{N \cdot L}$ is a vector. In other words, the output range $\mathbb{D} \triangleq: [\mathsf{o}_\ell, \mathsf{o}_u]$ is first partitioned into $N \cdot L - 1$ equidistant subintervals with endpoints $\hat{\mathsf{o}}_1, \ldots, \hat{\mathsf{o}}_k, \ldots \hat{\mathsf{o}}_{N \cdot L}$, where $\hat{\mathsf{o}}_1 = \mathsf{o}_\ell$, $\hat{\mathsf{o}}_{N \cdot L} = \mathsf{o}_u$, and $\hat{\mathsf{o}}_1 < \ldots < \hat{\mathsf{o}}_k < \ldots < \hat{\mathsf{o}}_{N \cdot L}$. Then,

the vector $\hat{\mathbf{o}}$ is constructed using all of the aforementioned endpoints, as follows:

$$\hat{\mathbf{o}} = [\hat{\mathsf{o}}_1, \ldots, \hat{\mathsf{o}}_k, \ldots, \hat{\mathsf{o}}_{N\cdot L}]^T. \tag{18}$$

In this paper, we assume that the same affine parameters are used for all hidden nodes. In such a case, there are only 2 parameters to be optimized in (17), for which the gradient descent algorithm can be adopted, which we address as follows:

Denote $\mathcal{E} = \frac{1}{2}\|g(s\cdot\boldsymbol{\nu}_{asc} + t\cdot\mathbb{1}) - \hat{\mathbf{o}}\|_2^2$. The derivatives of $\mathcal{E}$ w.r.t. s and t are

$$\frac{\partial \mathcal{E}}{\partial s} = \sum\nolimits_{k=1}^{N\cdot L}\left[g\left(s\cdot\nu_k + t\right) - \hat{\mathsf{o}}_k\right]\frac{\partial g}{\partial s}, \tag{19}$$

$$\frac{\partial \mathcal{E}}{\partial t} = \sum\nolimits_{k=1}^{N\cdot L}\left[g\left(s\cdot\nu_k + t\right) - \hat{\mathsf{o}}_k\right]\frac{\partial g}{\partial t}. \tag{20}$$

The parameters can be iteratively solved as

$$s(\kappa+1) \longleftarrow s(\kappa) - \eta\frac{\partial \mathcal{E}}{\partial s(\kappa)}, \tag{21}$$

$$t(\kappa+1) \longleftarrow t(\kappa) - \eta\frac{\partial \mathcal{E}}{\partial t(\kappa)}, \tag{22}$$

where κ denotes the iteration step, and η is the step size. More specifically, for the Sigmoid function, (19) and (20) can be computed as $\frac{\partial \mathcal{E}}{\partial s} = -\sum_{k=1}^{N\cdot L}\nu_k\left[g\left(s\cdot\nu_k+t\right) - \hat{\mathsf{o}}_k\right]\left[g\left(s\cdot\nu_k+t\right) - g^2\left(s\cdot\nu_k+t\right)\right]$ and $\frac{\partial \mathcal{E}}{\partial t} = -\sum_{k=1}^{N\cdot L}\left[g\left(s\cdot\nu_k+t\right) - \hat{\mathsf{o}}_k\right]\left[g\left(s\cdot\nu_k+t\right) - g^2\left(s\cdot\nu_k+t\right)\right]$, respectively. One can find that there exist repetitive terms in calculating the derivatives w.r.t. s and t, which can be utilized to reduce the computational complexity.

For almost all real-world applications, $N\cdot L \gg 2$. Thus, there is no need to employ all hidden node net inputs to perform the parameter estimation in (17). To reduce the computational complexity, a uniform downsampling is performed on $\boldsymbol{\nu}_{asc}$. Suppose that N_d $(N_d < N\cdot L)$ samples in $\boldsymbol{\nu}_{asc}$ are used to estimate the affine parameters; the downsampling rate is then equal to $\left\lfloor\frac{N\cdot L}{N_d}\right\rfloor$, where $\lfloor\cdot\rfloor$ is the rounding up function. Then, the newly formed data vector and its corresponding output vector after downsampling can be respectively described by

$$\tilde{\boldsymbol{\nu}}_d = [\tilde{\nu}_1, \ldots, \tilde{\nu}_i, \ldots, \tilde{\nu}_{N_d}]^T, \tag{23}$$

$$\tilde{\mathbf{o}}_d = [\tilde{\mathsf{o}}_1, \ldots, \tilde{\mathsf{o}}_i, \ldots, \tilde{\mathsf{o}}_{N_d}]^T, \tag{24}$$

where $i = 1, \ldots, N_d$. The new derivatives for the estimation of s and t can be derived as follows:

$$\frac{\partial \mathcal{E}}{\partial s} = \sum\nolimits_{i=1}^{N_d}\left[g\left(s\cdot\tilde{\nu}_i + t\right) - \tilde{\mathsf{o}}_i\right]\frac{\partial g}{\partial s}, \tag{25}$$

$$\frac{\partial \mathcal{E}}{\partial t} = \sum\nolimits_{i=1}^{N_d}\left[g\left(s\cdot\tilde{\nu}_i + t\right) - \tilde{\mathsf{o}}_i\right]\frac{\partial g}{\partial t}. \tag{26}$$

The proposed affine parameter estimation approach for the activation function is summarized in **Algorithm 1**.

The above iterative algorithm assumes that there are no constraints on the distribution of the hidden layer net inputs, i.e., it applies to any distributions of the training data and any randomly generated hidden weights and biases. In the special case that the hidden layer net inputs obey the zero mean Gaussian distribution, a closed-form formula for fast estimation of the affine parameters s and t based on **Algorithm 1** can be developed as follows.

Algorithm 1: Estimation algorithm for the affine parameters

Input : Hidden layer net input matrix $\mathbf{V}$, hidden node activation function $g(\cdot)$ with desired output range $\mathbb{D} \triangleq: [\mathsf{o}_\ell, \mathsf{o}_u]$, step size η, a small tolerance $\varepsilon > 0$, the maximum iterations $\mathcal{K}$, the downsampling number N_d $(N_d < N\cdot L)$.

Output : The affine parameters s and t.

```
Concatenate V into a vector ν_asc in the ascending order;
Partition D into (N·L) − 1 equidistant sub-intervals and
  construct the output vector ô using all sub-interval endpoints;
Obtain the downsampled vectors ν̃_d and õ_d from ν_asc and ô
  with the downsampling rate ⌊(N·L)/N_d⌋;
Randomly initialize s, t and compute E;
while (κ < K) or (E ≥ ε) do
    Compute the gradients with (25) and (26);
    s(κ+1) ⟵ s(κ) − η ∂E/∂s(κ);
    t(κ+1) ⟵ t(κ) − η ∂E/∂t(κ);
    E ⟵ (1/2)‖g(s(κ+1)·ν̃_d + t(κ+1)·1) − õ_d‖_F^2;
    κ ⟵ κ + 1;
end
```

For activation functions of the Sigmoid, Tanh, and Hyperbolic Tangent types, it is suggested in [33] that s and t can be computed empirically by simulations. For hidden node net inputs that obey the zero-mean Gaussian distribution $\mathcal{N}\left(0, \sigma^2\right)$, the scaling parameter s can be estimated as

$$s = \frac{1}{\frac{\sigma}{s_e}}, \tag{27}$$

and the translation parameter is directly taken as $t = 0$. Here, s_e is an empirical value of s for hidden node inputs that obey the standard normal distribution $\mathcal{N}\left(0, 1\right)$. Given an activation function with the output range $\mathbb{D}$, s_e can be estimated via maximizing the output entropy while the hidden node net inputs are generated from $\mathcal{N}\left(0, 1\right)$. It is noted that (27) can only be used for hidden node net inputs that obey the zero-mean Gaussian distribution. One can check through simulation that s_e is approximately 1.67 for the Sigmoid. Therefore, we only need to estimate the variance σ for the hidden node net inputs. The fast estimation algorithm presented for the standard derivation σ in [44, 45], i.e.,

$$\sigma = \frac{\text{Median}\left(\text{abs}\left(\mathbf{V}\right)\right)}{0.6745}, \tag{28}$$

is adopted in this paper, where $\text{abs}\left(\mathbf{V}\right)$ is the elementwise absolute value function of $\mathbf{V}$.

It is worthwhile to point out that the proposed **Algorithm 1** is different from the general rescaling for the network input data. The input data rescaling methods normally adjust their inputs in different scales to a common scale, e.g., by normalization. They usually do not change the data density distribution and thus cannot achieve entropy maximization for the hidden layer outputs.

D. The proposed AT-ELM algorithm

Without loss of generality, the RELM algorithm is employed as the basis for the proposed AT-ELM in this paper. To optimize the regularization parameter λ, the leave-one-out (LOO) cross-validation strategy with the predicted residual sum of squares (PRESS) statistic [46] is used in (7) or (8), as in [47, 48]. To speed up the calculation of the PRESS-based mean square error (MSE) in LOO ($\text{MSE}^{\text{PRESS}}$), the computationally efficient algorithm developed in [49] is adopted. A brief review of this algorithm is given below, while for the details, please refer to [49].

The LOO cross-validation approach optimizes λ through partitioning N training samples into N different combinations, with each combination having $N-1$ training instances and one left-out testing sample. The $\text{MSE}^{\text{PRESS}}$ used for the performance evaluation is

$$\text{MSE}^{\text{PRESS}} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\mathbf{t}_j - \mathbf{o}_j}{1-\text{HAT}_j}\right)^2, \tag{29}$$

where HAT_j denotes the j-th diagonal element of the HAT matrix described by [49]

$$\text{HAT} \quad = \quad \mathbf{H}\left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\mathbf{H}^T, \tag{30}$$

or when $L > N$, by the computationally more efficient formula as

$$\text{HAT} \quad = \quad \mathbf{H}\mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I}\right)^{-1}. \tag{31}$$

To reduce the repetitive computation in calculating $\text{MSE}^{\text{PRESS}}$ during the optimization of λ, novel matrix decomposition strategies other than the singular value decomposition (SVD) on $\mathbf{H}$ are introduced in [49] and adopted in this paper.

- When $L \le N$, the decomposition of $\mathbf{H}^T\mathbf{H} = \mathcal{V}\mathbf{D}^2\mathcal{V}^T$ is made, since in this case, the output weight (7), $\text{MSE}^{\text{PRESS}}$ (29) and HAT matrix (30) after certain transformations are irrelevant to the matrix $\mathbf{U}$ in SVD of $\mathbf{H}$ because $\mathbf{H} = \mathbf{U}\mathbf{D}\mathcal{V}^T$. In other words, (7) can be computed as $\hat{\boldsymbol{\beta}} = \mathcal{V}(\mathbf{D}^2+\lambda\mathbf{I})^{-1}\mathcal{V}^T\mathbf{H}^T\mathbf{T}$, and the diagonal elements of HAT can be derived from the row sum of $(\mathbf{H}\mathcal{V}(\mathbf{D}^2+\lambda\mathbf{I})^{-1}) \odot (\mathbf{H}\mathcal{V})$, where $\odot$ is the elementwise multiplication. With such decomposition and transformations, many repetitive calculations in deriving the $\text{MSE}^{\text{PRESS}}$ (29) can be avoided, including $\mathbf{H}\mathcal{V}$ and $\mathcal{V}^T\mathbf{H}^T\mathbf{T}$, because they are independent of λ and can be precalculated.
- When $L > N$, the decomposition of $\mathbf{H}\mathbf{H}^T = \mathbf{U}\mathbf{D}^2\mathbf{U}^T$ is conducted. In this case, (8) and (31) can be expressed as $\hat{\boldsymbol{\beta}} = \mathbf{H}^T\mathbf{U}(\mathbf{D}^2+\lambda\mathbf{I})^{-1}\mathbf{U}^T\mathbf{T}$ and $\text{HAT} = \mathbf{H}\mathbf{H}^T\mathbf{U}(\mathbf{D}^2+\lambda\mathbf{I})^{-1}\mathbf{U}^T$, respectively. The repetitive terms $\mathbf{H}\mathbf{H}^T\mathbf{U}$ and $\mathbf{U}^T\mathbf{T}$ can be precalculated before optimizing the $\text{MSE}^{\text{PRESS}}$.

With the estimated s, t and the LOO cross-validation approach in [49], the proposed AT-ELM algorithm can be described as follows. In the first step, the training dataset is used to estimate s and t via **Algorithm 1** or the fast method (27) (which is only suitable for the hidden node net inputs that obey the zero-mean Gaussian distribution). In the second step, the output weight matrix is derived using (7) or (8), where the optimal λ_{opt} is obtained by the efficient algorithm in [49], which was reviewed above in this subsection. **Algorithm 2** summarizes the proposed AT-ELM.

In addition to using random hidden node parameters, orthogonalization of hidden node parameters can help to improve the generalization performance of ELM [50]. In other words, after randomly generating the hidden node parameters $(\mathbf{w}_i, b_i)$, $i = 1, \ldots, L$, the following orthogonalization is performed:

$$\tilde{\mathbf{W}} \quad = \quad \text{orth}\left(\mathbf{W}\right), \tag{32}$$

$$\tilde{\mathbf{b}} \quad = \quad \text{orth}\left(\mathbf{b}\right), \tag{33}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_L]$, $\mathbf{b} = [b_1, \ldots, b_L]$, $\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_L]$, $\tilde{\mathbf{b}} = \left[\tilde{b}_1, \ldots, \tilde{b}_L\right]$, and $\text{orth}(\cdot)$ represents the orthogonalization operation, which satisfies

$$\tilde{\mathbf{W}}^T\tilde{\mathbf{W}} = \mathbf{I}, \quad \tilde{\mathbf{b}}^T\tilde{\mathbf{b}} = 1. \tag{34}$$

The performance of AT-ELM when utilizing orthogonal random hidden node parameters is also reported in the next section on experiments and discussions.

Algorithm 2: AT-ELM

Input : A training dataset $\{(\mathbf{x}_j, \mathbf{t}_j)\}_{j=1}^N$, hidden node number L, a set of regularization parameters $\lambda \in [\lambda_{\min}, \lambda_{\max}]$, hidden node activation function $g(\cdot)$ with the desired output range $\mathbb{D} \triangleq [\mathsf{o}_\ell, \mathsf{o}_u]$, the downsampling number N_d ($N_d < N \cdot L$) for affine parameter estimation.

Output : The affine parameters s and t, the optimal λ_{opt} and the corresponding output weight matrix $\hat{\boldsymbol{\beta}}$.

Randomly generate the hidden node weight vector and bias $(\mathbf{w}_i, b_i)$, $i = 1, \ldots, L$;
Calculate the hidden layer net input matrix $\mathbf{V}$ via (4);
Estimate the affine parameters s and t with **Algorithm 1** or the fast method (27) for zero-mean Gaussian distribution;
Compute the hidden layer output matrix as

$$\mathbf{H} = g(s \cdot \mathbf{V} + t \cdot \mathbb{1}),$$

where $\mathbb{1} \in R^{N \cdot L}$ is a matrix with all elements equal to 1;
Optimize λ in (6) with the algorithm in [49], ;
if $L \le N$ **then**
    $\hat{\boldsymbol{\beta}} = \mathcal{V} \odot \text{repmat}(1./(\mathbf{d} + \lambda_{opt}), N, 1)\mathbf{F}$;
    with $\mathbf{d} = (\text{diag}(\mathbf{D}^2))^T$, $\mathbf{F} = \mathbf{E}^T\mathbf{T}$, $\mathbf{E} = \mathbf{H}\mathcal{V}$ and $\mathbf{H}^T\mathbf{H} = \mathcal{V}\mathbf{D}^2\mathcal{V}^T$
end
if $L > N$ **then**
    $\hat{\boldsymbol{\beta}} = \mathbf{H}^T(\mathbf{U} \odot \text{repmat}(1./(\mathbf{d} + \lambda_{opt}), N, 1))\mathbf{F}$;
    with $\mathbf{d} = (\text{diag}(\mathbf{D}^2))^T$, $\mathbf{F} = \mathbf{U}^T\mathbf{T}$ and $\mathbf{H}\mathbf{H}^T = \mathbf{U}\mathbf{D}^2\mathbf{U}^T$
end

IV. Experiments and Discussion

Experiments on nonlinear function regression, real-world dataset classification, and benchmark image recognition are conducted in this section to test the effectiveness of the proposed AT-ELM. The affine transformation activation function is taken as the Sigmoid type,

$$g(x, s, t) \quad = \quad \frac{1}{e^{-(s\cdot x + t)} + 1} \tag{35}$$

with s and t denoting the two affine parameters. In general, the affine parameters s and t are assumed to be 1 and 0, respectively, in the original ELM and RELM as well as most of the other artificial neural network models. It is noted that almost all conventional activation functions can be adopted in the proposed AT-ELM algorithms. For illustration, experimental results when using the Hyperbolic Tangent function are also presented in the subsequent Section IV-C for comparison.

Depending on whether **Algorithm 1** or the fast closed form formula (27) (only applied for zero-mean Gaussian distribution) is used, and whether the hidden node weights and biases are orthogonalized, we have the following combinations for the AT-ELM:

- **AT-ELM**$_1$ denotes the AT-ELM algorithm that employs (27) for s and t estimation;
- **AT-ELM**$_2$ denotes the AT-ELM algorithm that employs **Algorithm 1** for s and t estimation;
- **AT-ELM**$_{\mathbf{Orth},1}$ denotes the AT-ELM$_1$ algorithm with orthogonal hidden node parameters;
- **AT-ELM**$_{\mathbf{Orth},2}$ denotes the AT-ELM$_2$ algorithm with orthogonal hidden node parameters.

To show the superiority of our proposed algorithms, comparisons with the original ELM [2], RELM [39], kernel-based ELM (KELM) [3] with Gaussian kernel function, RELM with λ optimized using the LOO method in [49] (in short as RELM$_{\text{LOO}}$), as well as several other state-of-the-art algorithms are made. All of the experiments were conducted in Matlab R2014a on a PC with an Intel(R) Core(TM) i3-3240 processor (3.40 GHz). In **Algorithm 1**, the number of samples used for the affine parameter estimation is set to $N_d = 20000$ given that $N \cdot L > 20000$, but for those experiments with $N \cdot L \leq 20000$, all of the samples in the hidden layer net input matrix $\mathbf{V}$ were used. A discussion on the sensitivity to N_d will be given in Section IV-B.

A. Nonlinear function regression

Potocnik and Govekar show in [34] that with a fixed hidden node activation function, the ELM is quite sensitive to the range scaling of the input variables. To show the efficiency of the proposed AT-ELM on handling the data scaling and shifting, we test the regression performance on two nonlinear functions, namely, the 'SinC' function $f_1(\cdot)$ and a two-dimensional (2D) function $f_2(\cdot)$, in this subsection. The formulations of these two functions are

$$\text{SinC}: \qquad f_1(x) = \begin{cases} \frac{\sin(x)}{x}, & x \neq 0 \\ 1, & x = 0 \end{cases}, \tag{36}$$

$$\text{2D}: \qquad f_2(x_1, x_2) = \left(x_1^2 - x_2^2\right) \sin(0.5x_1). \tag{37}$$

Conventional experiments presented in [2, 51, 52] show that the ELM performs well for input data generated in the range $[-10, 10]$ for both functions. Following [34], the practical regression performance of the proposed algorithm is studied through scaling the input data by different factors γ. In other words, the original inputs are first generated in $[-10, 10]$ and then scaled by the following factors:

$$\gamma = \left\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3, 10^4\right\}. \tag{38}$$

For each scaling factor γ, multiple trials were conducted, and the average performance is compared. For the 'SinC' function, 5000 training samples and 5000 testing samples uniformly distributed in $[-10, 10]$ were generated. For the '2D' function, 40 samples for both x_1 and x_2, which form a 40×40 grid, were generated for training, while another 40×40 grid points were generated for testing. Uniformly distributed noise in $[-0.2, 0.2]$ were added to the training target outputs to test the algorithms' robustness. In these experiments, we assume that the training and testing samples share the same distribution. For the 'SinC' and the '2D' functions, the numbers of hidden nodes that change from 20 to 100 with an interval of 10 and from 100 to 500 with an interval of 100 are tested for all algorithms. The results obtained with the hidden node number that corresponds to the best approximation performance are then reported. The search ranges of λ for the LOO cross-validation algorithm [49] for RELM$_{\text{LOO}}$, **AT-ELM**$_1$, **AT-ELM**$_2$, **AT-ELM**$_{\mathbf{Orth},1}$, and **AT-ELM**$_{\mathbf{Orth},2}$ are $\left\{e^{[-50:0.2:10]}\right\}$, $\left\{e^{[-50:0.2:10]}\right\}$, $\left\{e^{[-50:1:10]}\right\}$, $\left\{e^{[-50:0.2:10]}\right\}$, and $\left\{e^{[-150:0.2:10]}\right\}$, respectively. The root mean square errors (RMSE) between the target outputs and the estimated outputs were computed for comparison.

Tables I and II show the RMSEs for the 'SinC' and '2D' functions obtained by the proposed AT-ELM algorithms (namely, **AT-ELM**$_1$, **AT-ELM**$_2$, **AT-ELM**$_{\mathbf{Orth},1}$, **AT-ELM**$_{\mathbf{Orth},2}$) and the existing ELM, RELM and RELM$_{\text{LOO}}$. The best and second-best results are highlighted by boldface and underlined in the table, respectively. From Table I, when using the fixed Sigmoid, ELM, RELM and RELM$_{\text{LOO}}$ only work well for certain scaling factors, e.g., $\gamma = 1$. In the meantime, the proposed AT-ELM algorithms perform slightly better than the ELM, RELM and RELM$_{\text{LOO}}$ when $\gamma = 1$. With other smaller or larger γ values, the generalization performance of the ELM, RELM and RELM$_{\text{LOO}}$ become poor. Specifically, when γ becomes larger than 100, the three competitive algorithms fail in approximating the 'SinC' function due to large RMSEs. In contrast, the proposed AT-ELM algorithms perform consistently well for all of the tested scaling factors because the activation function can adaptively adjust its affine parameters to the distribution of hidden layer net inputs. At the same time, the AT-ELM also outperforms the other two relevant algorithms, CFWNN-ELM [51] and CWN-E-ELM [52]. The RMSEs on the two functions $f_1(\cdot)$ and $f_2(\cdot)$ by CFWNN-ELM are 0.0187 and 0.3938, and those by CWN-E-ELM are 0.0112 and 0.0534, respectively.

For a demonstration, Figs. 2 (a) and 2 (b) draw the sorted hidden node outputs of the 'SinC' function obtained by the original ELM and the proposed AT-ELM with the scaling factors $\gamma =: 0.1$ and 100, respectively. As shown in Fig. 2 (a), diminishing the inputs with $\gamma = 0.1$ leads to concentrating the hidden layer output in the flat region of the Sigmoid, and amplifying the inputs with $\gamma = 100$ makes almost all of the hidden layer outputs move to the saturated regime of the Sigmoid (specifically, the outputs are close to 1 or 0, as depicted in Fig. 2 (b) and (c)). In contrast, the hidden node outputs obtained by the AT-ELM are evenly distributed within the output range of the Sigmoid. The estimated affine param-

TABLE I
RMSE COMPARISON OF THE 'SINC' FUNCTION.

	ELM	RELM	RELM_{LOO}	**AT-ELM$_1$**	**AT-ELM$_2$**	**AT-ELM$_{\text{Orth},1}$**	**AT-ELM$_{\text{Orth},2}$**
Scale $\gamma =$:							
10^{-2}	0.1453	0.2746	0.1121	0.0072	0.0064	0.0062	**0.0060**
10^{-1}	0.1088	0.2122	0.0476	0.0058	0.0057	0.0058	**0.0056**
1	0.0064	0.099	0.0065	0.0057	0.0058	0.0060	**0.0056**
10	0.0288	0.0476	0.0152	**0.0060**	**0.0060**	0.0062	0.0061
10^2	0.1765	0.1652	0.1755	0.0062	**0.0061**	0.0065	0.0063
10^3	14.277	0.3203	0.3347	0.0089	0.0072	0.0090	**0.0068**
10^4	17.522	0.3341	0.4987	0.0156	**0.0083**	0.0165	0.0085

TABLE II
RMSE COMPARISON OF THE '2D' FUNCTION.

	ELM	RELM	RELM_{LOO}	**AT-ELM$_1$**	**AT-ELM$_2$**	**AT-ELM$_{\text{Orth},1}$**	**AT-ELM$_{\text{Orth},2}$**
Scale $\gamma =$:							
10^{-2}	8.5534	18.473	1.245	0.0465	0.0435	0.0388	**0.0372**
10^{-1}	6.8845	15.237	1.021	0.0477	0.0421	0.0371	**0.0352**
1	0.3654	1.134	0.3865	0.0398	0.0402	0.0373	**0.0364**
10	18.765	19.543	17.965	0.0376	0.0368	0.0362	**0.0358**
10^2	22.867	24.432	21.226	0.0341	0.0356	**0.0328**	0.0340
10^3	1.84e+03	30.566	20.657	0.0347	0.0363	**0.0328**	0.0340
10^4	1.03e+03	24.223	18.665	0.0336	0.0362	**0.0308**	0.0342

TABLE III
TRAINING TIME (SECONDS) COMPARISON OF THE 'SINC' AND '2D' FUNCTION APPROXIMATION.

Scale $\gamma = 1$	ELM	RELM	RELM_{LOO}	**AT-ELM$_1$**	**AT-ELM$_2$**	**AT-ELM$_{\text{Orth},1}$**	**AT-ELM$_{\text{Orth},2}$**
SinC	0.0142	0.0133	0.9114	0.9203	0.3415	0.9224	2.6556
2D	0.0316	0.0313	1.1902	1.2042	0.4385	1.2042	3.4562

eters for the activation function obtained via **Algorithm 1** for the above two γ values are $(s,t) =: (1.1537, -0.0830)$ and $(0.0024, -0.0015)$, respectively. Obviously, to compensate the amplification in the sample inputs, a parameter s, with a value less than 1, has been obtained in the AT-ELM for $\gamma = 100$. On the other hand, when $\gamma = 0.1$, a parameter s, with a value larger than 1, is derived.

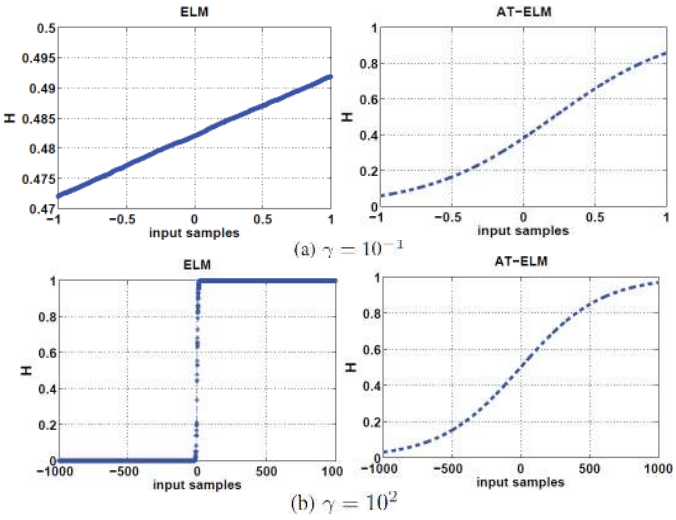


Fig. 2. Hidden node outputs of SinC with $\gamma =: 10^{-1}$ and 10^2, respectively.

In addition to the generalization performance, the training time is another concern in machine learning algorithms. Although the gradient descent method is adopted in **Algorithm 1**, the increase of the computational complexity in AT-ELMs due to the estimation of the affine parameters is very small since only two parameters are required to be optimized. For illustration, the computational complexity is compared in terms of the training time in Table III for $\gamma = 1$. Although the training time spent by the four AT-ELM algorithms is obviously higher than that by the ELM, the time is mainly the cost of searching for the optimal λ by the LOO cross-validation algorithm. Moreover, it is apparent that the larger the set $[\lambda_{\min}, \lambda_{\max}]$ is, the longer the training time will be needed. For example, the proposed **AT-ELM$_1$** has a comparable training time to RELM_{LOO} because they have the same search range. The only difference lies in the adjustment of the affine transformation activation function in **AT-ELM$_1$**. As can be seen from Table III, compared to RELM_{LOO}, the estimation of the affine parameters in **AT-ELM$_1$** takes only $0.9203 - 0.9114 = 0.0089$s for the 'SinC' function. Similarly, **AT-ELM$_{\text{Orth},2}$** spends a longer training time than **AT-ELM$_2$** due to the larger range $[\lambda_{\min}, \lambda_{\max}]$ that it uses. In addition, if the searching range of λ is the same in all four proposed algorithms, the methods that employ **Algorithm 1** for the scaling and translation parameter estimation (**AT-ELM$_2$** and **AT-ELM$_{\text{Orth},2}$**) usually spend a longer time than those (**AT-ELM$_1$** and **AT-ELM$_{\text{Orth},1}$**) that use (27). The reason is that **Algorithm 1** adopts the gradient descend method for the affine parameter estimation, while (27) is only required to calculate the variance of the hidden node net inputs.

As mentioned previously, the parameter t is set to 0 in the case when the hidden node net inputs obey the zero-mean Gaussian distribution. However, it is important to adjust t in other cases. To demonstrate this relationship, an experiment on the 'SinC' function estimation by generating input data with different means that range from 4 to 16 is conducted. The AT-ELM algorithm using the Sigmoid function with or without

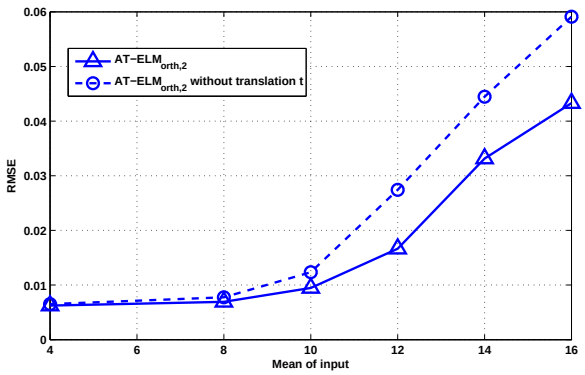


Fig. 3. RMSEs obtained by the AT-ELM with/without the parameter t.

the parameter t is employed for function estimation. Fig. 3 shows a comparison of the RMSEs. As depicted, adjusting t in the AT-ELM provides a smaller RMSE than that without the parameter.

B. Real-world dataset classification

In this section, 34 benchmark classification datasets that cover a wide range of real-world applications are tested for performance comparison. Specifications that include the number of attributes, the numbers of training and testing samples, and the number of classes are shown in Table IV. The first 20 datasets are from the UCI website[2], and the remaining datasets are from the feature selection website[3]. The Handwritten digit dataset is tested using both the partial features, namely, the profile correlation (Pro), Karhunen-Love coefficient (Kar), pixel average (Pix), Zernike moment (Zer), and the whole set of features (All). For all of the datasets, the raw features are used for classifier training and testing. The training and testing samples are randomly shuffled in each trial except for the Protein dataset, whose training and testing datasets are fixed during the experiment. The ELM, RELM, RELM$_{\text{LOO}}$, and kernel-based ELM (KELM) are included for performance comparison. For small size datasets, including Breast Tissue, Diabetic, Heart disease, Bupa liver, Breast Cancer and Wine, the optimal number of hidden nodes is searched within the range $[100 : 50 : 600]$. For those datasets with a very large number of attributes or samples, including Online-news, Handwritten(All), Carcinom, gisette, TOX, and 20Newsgroup, the optimal number of hidden nodes is searched within the range $[500 : 100 : 2500]$. For the remaining datasets, the optimal number of hidden nodes is searched within the range $[500 : 100 : 1500]$. For the LOO cross-validation, the same searching range $\{e^{[-10:0.2:10]}\}$ for the regularization parameter is used in all of the LOO-based approaches. The Gaussian kernel function is adopted in KELM, where the cost parameter and kernel parameter are both optimized within $\{2^{-24}, 2^{-23}, \cdots, 2^{24}, 2^{25}\}$. For all of the algorithms, the best results are reported in Table V for comparison.

[2]http://www.ics.uci.edu/∼mlearn/MLRepository.html
[3]http://featureselection.asu.edu/datasets.php

TABLE IV
SPECIFICATIONS OF THE BENCHMARK CLASSIFICATION DATASETS

Datasets	# Attributes	# Train	#Test	#Classes
Breast Tissue	9	75	31	6
Cardiotocography	21	1701	425	3
Diabetic	19	806	345	2
Heart disease	13	100	170	2
Letter	16	10000	10000	26
Leukemia	7129	38	34	2
Bupa liver	6	200	145	2
Magic	10	10000	9020	2
Online-news	59	31715	7929	2
Optical-digits	64	3823	1797	10
Pendigits	16	7494	3498	10
Protein	56	949	534	10
Satimage	36	3217	3218	6
Breast Cancer	30	300	269	2
Wine	13	100	78	3
Handwritten(Pro)	216	1200	800	10
Handwritten(Kar)	64	1200	800	10
Handwritten(Pix)	240	1200	800	10
Handwritten(Zer)	47	1200	800	10
Handwritten(All)	649	1200	800	10
Lung	3312	122	81	5
Carcinom	9182	104	70	11
gisette	5000	4200	2800	2
TOX	5748	103	68	4
Isolet1	617	936	624	26
Isolet2	617	936	624	26
Isolet3	617	936	624	26
Isolet4	617	935	623	26
Isolet5	617	935	624	26
PC/MAC	3289	1166	777	2
BASE/HOCK	4862	1196	797	2
REL/ALTHE	4322	856	571	2
20Newsgroup	26214	11307	7539	20
WarpAR10P	2400	78	52	10

The average testing accuracy (denoted as Rate) and standard deviations (Std) obtained by the four existing ELM algorithms and the four proposed AT-ELMs are reported in Table V. As seen from the table, the AT-ELMs have higher classification rates than the four competitive algorithms on all 34 classification datasets. Among all of the 34 classification datasets, the AT-ELMs achieve an improvement of more than 5% over the best of the four competitive ELM algorithms on 13 datasets, as indicated by underlined boldface, and a $0.73\% \sim 5\%$ improvement over the best of the four competitive ELM algorithms on 21 datasets, as indicated by boldface. Within the four AT-ELM algorithms, the **AT-ELM$_{\text{Orth},2}$** method performs better than or identically to the remaining three methods on 16 datasets, and the **AT-ELM$_{\text{Orth},1}$** method shows its superiority on 12 datasets. Utilizing the orthogonal hidden node parameters indeed helps enhance the classification performance with respect to the 34 tested classification datasets.

Furthermore, compared to the original ELM, AT-ELM achieves an improvement of more than 16% on Wine, Isolet1, Isolet2, Isolet3, Isolet4, and Isolet5, $10\% \sim 20\%$ on Breast Tissue, Heart disease, Diabetic, Leukemia, Bupa liver, Handwritten(Kar), Handwritten(Pix), Handwritten(All), TOX, PC/MAC, BASE/HOCK, REL/ALTHE, and WarpAR10P, and 0% to 10% on the remaining 15 datasets. The poor performance of the original ELM is mainly due to the hidden node outputs being mostly located in the saturated regions or the flat region of the Sigmoid for these databases. Fig.

TABLE V
PERFORMANCE COMPARISON (RATE, STD%) ON BENCHMARK DATASETS USING THE SIGMOID FUNCTION.

Datasets	ELM		RELM		RELM_{LOO}		KELM		**AT-ELM$_1$**		**AT-ELM$_2$**		**AT-ELM$_{\text{Orth},1}$**		**AT-ELM$_{\text{Orth},2}$**	
	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std
Breast Tissue	57.45	8.22	59.03	7.88	62.15	6.88	49.56	9.23	74.88	5.51	73.55	5.98	74.44	6.67	**75.12**	6.44
Cardiotocography	85.95	1.63	86.74	1.39	88.86	1.44	89.02	1.28	90.77	1.28	90.46	1.23	91.23	1.22	**91.36**	1.28
Diabetic	65.88	2.65	66.83	2.43	69.95	2.45	68.87	2.66	77.22	1.98	**79.42**	1.79	78.55	1.58	79.13	1.35
Heart disease	64.88	3.76	64.86	3.52	69.85	3.12	64.88	3.54	82.03	2.32	82.22	2.65	**82.94**	2.42	81.88	2.44
Letter	88.96	0.31	88.97	0.35	88.82	0.36	89.21	0.28	92.62	0.26	92.62	0.24	92.72	0.22	**92.78**	0.21
Leukemia	88.45	4.53	88.86	4.99	91.02	4.34	90.88	4.65	97.12	3.55	97.08	3.44	**100**	2.61	97.11	2.66
Bupa liver	58.22	4.11	61.05	3.98	67.21	3.22	66.87	3.46	74.76	2.34	74.21	2.88	**74.87**	2.52	73.87	3.20
Magic	77.25	0.42	77.34	0.35	78.53	0.36	78.32	0.43	84.54	0.29	84.85	0.32	84.87	0.28	**85.23**	0.28
Online-news	58.22	0.51	58.12	0.44	59.32	0.70	58.87	0.68	62.41	0.62	62.77	0.98	62.54	0.54	**62.78**	0.58
Optical-digits	96.33	0.41	96.85	0.38	97.21	0.35	96.55	0.32	98.33	0.22	98.12	0.24	**98.53**	0.22	98.15	0.20
Pendigits	96.97	0.32	97.05	0.22	97.22	0.21	97.45	0.20	98.21	0.12	98.14	0.11	98.14	0.12	**98.25**	0.10
Protein	87.21	1.12	87.32	0.22	90.13	0.54	89.43	0.66	91.21	0.76	90.58	0.70	**91.43**	0.65	90.82	0.86
Satimage	78.88	0.86	79.65	0.75	80.22	0.68	81.22	0.65	87.48	0.59	**87.76**	0.49	87.43	0.56	87.55	0.56
Breast Cancer	88.21	2.33	88.96	1.56	88.87	1.82	89.32	2.03	**96.76**	0.89	96.65	0.92	96.67	0.90	96.45	0.98
Wine	66.56	6.87	67.80	6.35	68.23	6.78	70.23	5.87	96.10	2.33	96.10	2.24	96.22	2.34	**96.25**	2.10
Handwritten(Pro)	91.32	1.21	91.88	1.24	92.46	0.98	92.55	0.86	98.46	0.49	98.45	0.65	98.41	0.47	**98.52**	0.48
Handwritten(Kar)	86.78	1.29	86.55	1.26	96.66	0.61	96.25	0.53	97.34	0.41	**97.39**	0.54	97.36	0.43	**97.39**	0.42
Handwritten(Pix)	86.32	1.32	89.22	1.54	96.67	0.58	95.43	0.58	97.43	0.32	**97.75**	0.35	97.38	0.42	97.70	0.24
Handwritten(Zer)	78.21	1.56	78.66	1.43	79.87	1.32	78.88	1.55	84.86	0.68	84.78	0.80	84.92	0.67	**85.01**	0.65
Handwritten(All)	89.21	0.65	90.43	0.87	95.76	0.96	96.21	0.56	98.88	0.37	**99.02**	0.34	99	0.37	99	0.38
Lung	89.77	3.45	89.89	4.21	92.56	2.44	93.11	2.54	96.12	1.54	96.87	1.24	95.99	1.79	**96.42**	1.65
Carcinom	86.87	3.22	87.67	3.34	88.22	3.46	89.02	3.12	95.94	2.08	96.22	1.43	**96.65**	2.32	96.30	1.91
gisette	90.88	0.65	90.83	0.51	95.02	0.44	95.31	0.22	96.45	0.34	96.44	0.33	96.50	0.39	**96.54**	0.32
TOX	78.02	5.12	78.56	4.53	78.97	4.24	79.02	3.48	92.54	4.28	92.77	4.02	92.87	4.12	**93.34**	3.22
Isolet1	45.33	2.40	51.43	2.55	93.66	0.87	93.55	0.76	94.46	0.68	94.40	1.01	94.37	0.81	**94.55**	0.97
Isolet2	46.32	2.55	50.56	2.42	92.31	0.97	93.04	0.77	94.18	0.78	94.21	0.74	**94.44**	0.89	94.30	0.94
Isolet3	44.65	3.87	45.76	1.98	90.21	0.97	90.77	0.87	91.66	0.76	91.75	0.89	**91.76**	0.77	91.59	0.73
Isolet4	42.11	2.43	45.76	2.87	89.42	1.23	89.97	1.55	90.66	1.23	90.88	1.21	90.87	1.10	**91.00**	0.96
Isolet5	44.88	2.76	48.77	3.76	91.23	0.87	91.33	0.98	91.80	0.86	91.87	1.21	**91.95**	0.83	91.67	0.76
PC/MAC	81.67	1.52	82.43	1.44	89.77	1.32	89.87	1.26	89.42	1.11	89.45	1.21	90.02	1.00	**90.12**	1.09
BASE/HOCK	80.23	1.22	82.31	2.34	94.43	0.67	94.21	0.77	95.31	0.47	95.47	0.65	**95.77**	0.87	95.61	0.81
REL/ALTHE	70.32	2.11	71.87	2.23	84.67	1.78	84.55	1.89	86.32	1.44	86.56	1.23	**86.71**	1.21	86.44	1.29
20Newsgroup	79.55	0.43	79.86	0.34	80.21	0.44	80.31	0.25	82.33	0.45	82.56	0.32	**82.71**	0.24	82.66	0.45
WarpAR10P	87.21	5.46	87.43	5.31	89.66	4.55	91.45	2.87	97.60	2.72	**97.72**	2.55	97.54	2.43	96.87	2.66

4 compares the hidden node outputs in the ELM and AT-ELM on Breast Tissue and Diabetic, respectively. As expected, using the fixed Sigmoid, the hidden node outputs in ELM on Breast Tissue are highly concentrated in the tail regions. While for the Diabetic dataset, the hidden node outputs are mainly distributed in either the tail or the center flat regions. In contrast, the curves of the hidden node outputs obtained by the proposed AT-ELM remain similar to the shape of the Sigmoid because they are evenly distributed within the range $[0, 1]$. Although the LOO cross-validation approach helps enhance the classification performance of the ELM for some datasets, the classification rate is only slightly enhanced for most datasets or even becomes worse in a few cases. Mere optimization over λ is usually unable to overcome the difficulty that arises from using the fixed activation functions mentioned in Section II-B. In this section, a comparison of the computational complexity is omitted due to page limitations.

The sensitivity to the number of hidden nodes for all of the algorithms and the downsampling number N_d for the affine parameter estimation is also tested in this section. Fig. 5 depicts the performance of the ELM, RELM, RELM_{LOO}, KELM, and the proposed **AT-ELM$_1$**, in terms of the number of hidden nodes on the Heart disease, Diabetic, and Breast Tissue datasets. The performances of the remaining three AT-ELMs is not presented in the figure because they are close to that of **AT-ELM$_1$**. As shown in the figure, employing different numbers of hidden nodes can affect the recognition accuracy, but the best accuracy by the proposed **AT-ELM$_1$** is much higher than those by the ELM, RELM, RELM_{LOO}, and KELM. Fig. 6 presents the

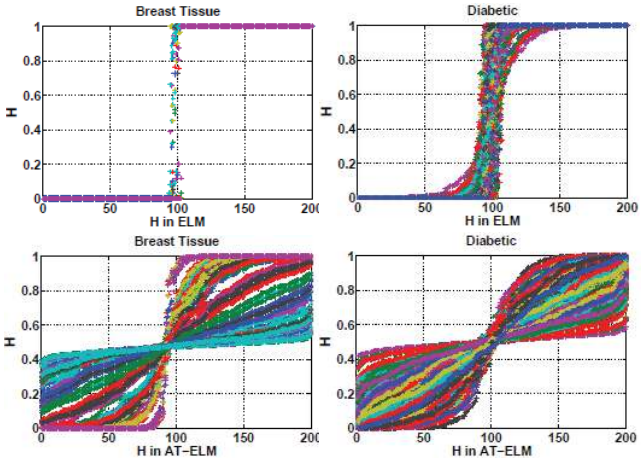


Fig. 4. Hidden node outputs in the ELM and AT-ELM on Breast Tissue and Diabetic datasets.

recognition accuracy obtained by the proposed **AT-ELM$_2$** and **AT-ELM$_{\text{Orth},2}$** w.r.t. different N_d values used in **Algorithm 1** for the affine parameter estimation. Seven different N_d values $N_d = [500, 1000, 5000, 10000, 15000, 20000, 25000]$ are tested on the Handwritten(Kar) and Isolate1 datasets. As depicted, the variations in the recognition accuracy for different N_d values are slight, which indicates that **AT-ELM$_2$** and **AT-ELM$_{\text{Orth},2}$** are not sensitive to N_d. In fact, only two parameters must be estimated. Any number of samples that well cover the output range of the activation function should be very acceptable.

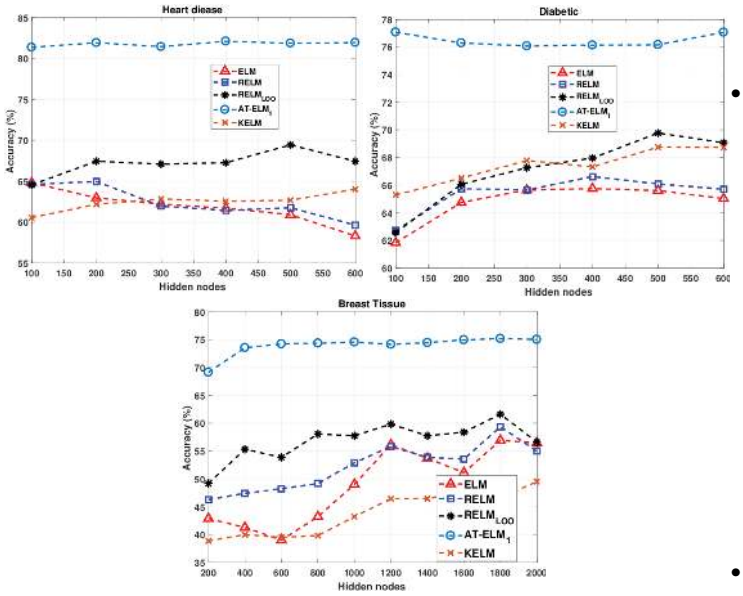
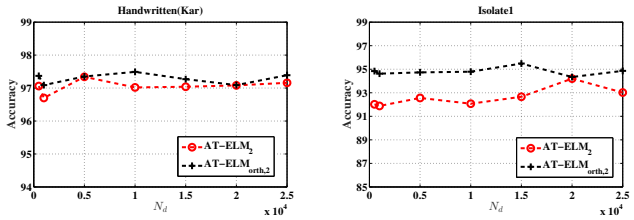


Fig. 5. Performance comparisons of different algorithms for different numbers of hidden nodes on the Heart disease, Diabetic, and Breast Tissue datasets.



(a) Handwritten(Kar) (b) Isolate1

Fig. 6. Recognition performance w.r.t. different N_d values.

C. Image recognition

Further experiments on 6 benchmark image databases are presented in this section. Except that the experiment on USPS has only one setup, those on the other 5 databases have two setups, which are described in detail in the following

- **AR:** The AR face database consists of 2600 image samples collected from 100 individuals. We tested the performance on this database under two setups. In the first setup, all of the 2600 images were adopted, and 540 features extracted from each image were used. The first 8, 10, 12, 14 and 16 images per person were used for training, and the remaining samples were employed for testing. In the second setup, only 1400 images (the first 14 samples per person) were adopted, and 300 eigenface features extracted from each image were used in the experiment. Half of the 1400 images were used for model training, and the remaining images were adopted for performance testing.
- **USPS:** The USPS database includes a total of 9298 handwritten samples for the ten digits 0∼9, collected from disjoint writers. The pixels of each 16×16 grayscale image were directly used as the features for digit representation. In the experiment, 7291 images were used for classifier training, and the remaining images were used for testing.
- **ORL:** The ORL face database has two datasets of the same images taken from 40 different face subjects, one set with a 16×16 grayscale and the other set with a 32×32 grayscale. For each subject, 10 image samples taken at different times with varying lighting, facial expressions and facial details were collected. On this database, the experiment has two setups. In both setups, pixels of the grayscale images are adopted as the features. In the first setup, 2, 3, 4, 5, 6, 7 and 8 samples per category of the dataset with the 32×32 grayscale images were selected for the classifier learning, and the remaining samples were used for testing. In the second setup, 60% of the images were randomly selected for the model training, and the remaining 40% were used for testing for each of the two datasets.
- **Yale:** The Yale face database also has two datasets, which are constituted by the same 165 images that belong to 15 individuals, one dataset with a 32×32 grayscale and the other with a 64×64 grayscale. For each individual, 11 image samples are captured under different expressions or configurations, such as happy, sad, sleepy, with/without glasses, and so on. On this database, the experiment also has two setups. In the first setup, 3, 4, 5, 6, 7 and 8 images per category from the dataset with the 32×32 grayscale were selected to train the model, and the remaining images were used for testing. In the second setup, 60% of the images were randomly chosen for the model training and the remaining 40% were used for testing for each of the two datasets.
- **Yale B:** The extended Yale B face database consists of 2414 samples, collected from 38 different individuals. There are approximately 64 near-frontal images under different illuminations for each person in the database. Each image has been cropped and resized to 32×32, and the pixels are employed as the features. The two experimental setups on this database are as follows. In the first setup, 5, 10, 20, 30, 40 and 50 images per category were used for training, and the remaining images were adopted for testing. In the second setup, 60% of the images were randomly chosen from the dataset and were employed for classifier training, and the remaining images were used for testing.
- **MNIST:** The MNIST database contains binary images of handwritten digits. The original database includes 60,000 training and 10,000 testing samples from hundreds of disjoint writers. The experiment on this database was also conducted under two setups, where for each image, the grayscale pixels were used as the features. In the first setup, the first 2,000 training samples and the first 2,000 testing samples were used. In the second setup, the first 10,000 training samples and all of the 10,000 testing samples were adopted.

Similar to Section IV-B, the optimal number of hidden nodes for all ELM-based algorithms is searched within a certain range. The detailed range for MNIST with 10,000

TABLE VI
PERFORMANCE COMPARISON (RATE, STD%) ON BENCHMARK IMAGE DATASETS.

Datasets	ELM		RELM		RELM_{LOO}		**AT-ELM$_1$**		**AT-ELM$_2$**		**AT-ELM$_{\text{Orth},1}$**		**AT-ELM$_{\text{Orth},2}$**	
	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std
Sigmoid														
AR	72.43	1.44	73.66	1.11	86.54	0.87	92.31	0.65	92.66	0.57	92.29	0.50	**92.67**	0.50
USPS	92.77	0.80	92.66	0.77	92.46	0.73	**93.76**	0.58	93.44	0.76	93.22	0.65	93.47	0.66
ORL (32×32)	72.44	4.23	73.21	3.78	87.25	2.56	95.28	1.44	94.97	1.67	95.21	1.66	**95.36**	1.44
ORL (64×64)	90.24	3.45	91.44	1.43	92.43	2.56	96.72	1.21	96.94	1.04	**97.12**	0.77	96.88	0.56
Yale (32×32)	72.43	4.55	73.67	5.42	74.55	3.88	82.87	4.32	82.87	4.76	83.65	4.46	**84.12**	4.27
Yale (64×64)	82.66	3.43	82.45	4.21	82.33	4.67	89.88	3.11	91.21	2.56	90.34	3.22	**91.76**	2.88
Yale B	70.76	1.88	71.12	1.87	89.21	1.44	97.11	0.73	97.21	0.66	**97.54**	0.56	97.51	0.45
MNIST (2000)	82.55	0.68	82.31	0.57	87.88	0.57	88.43	0.57	**88.79**	0.34	87.95	1.03	88.12	0.77
MNIST (10000)	93.02	0.13	92.88	0.18	93.45	0.10	**94.97**	0.10	94.90	0.08	94.88	0.12	94.87	0.10
Hyperbolic Tangent														
AR	71.25	1.33	71.44	1.11	84.66	0.95	91.90	0.76	92.34	0.72	91.87	0.68	**92.88**	0.65
USPS	92.44	0.87	92.08	1.21	91.87	0.97	**93.87**	0.65	93.55	0.89	93.45	0.67	93.67	0.78
ORL (32×32)	71.87	4.56	73.25	3.87	88.45	2.34	95.66	1.34	95.98	1.32	95.80	1.42	**96.22**	1.22
ORL (64×64)	91.22	2.12	91.78	2.24	92.76	2.41	96.12	1.55	96.21	1.23	96.02	1.48	**96.32**	1.21
Yale (32×32)	73.27	4.34	74.88	5.82	74.89	4.45	80.55	4.42	82.87	5.34	80.97	4.55	**83.66**	5.10
Yale (64×64)	82.32	5.21	82.98	4.55	81.77	3.68	89.55	2.78	90.33	3.21	89.78	4.28	**90.44**	3.24
Yale B	70.25	1.66	71.71	1.56	89.77	1.04	97.11	0.58	97.62	0.55	97.68	0.55	**97.95**	0.54
MNIST (2000)	81.55	0.77	81.28	1.22	87.44	0.87	**89.33**	0.58	88.54	0.88	89.12	0.76	88.43	0.57
MNIST (10000)	92.87	0.23	92.87	0.22	93.34	0.21	94.88	0.13	94.87	0.09	**94.94**	0.12	94.88	0.09

TABLE VII
TRAINING TIME COMPARISON (SECONDS) ON BENCHMARK IMAGE DATASETS (SIGMOID).

Datasets	ELM	RELM	RELM_{LOO}	**AT-ELM$_1$**	**AT-ELM$_2$**	**AT-ELM$_{\text{Orth},1}$**	**AT-ELM$_{\text{Orth},2}$**
AR	1.3822	1.4355	3.574	3.6745	3.8786	3.6754	3.8957
USPS	1.9780	1.9867	11.273	12.121	11.254	11.457	12.477
ORL (32×32)	0.0556	0.0621	0.1342	0.1455	0.5632	0.1476	0.4421
ORL (64×64)	0.0988	0.0975	0.1822	0.2134	0.7734	0.2254	0.6859
Yale (32×32)	0.0187	0.0195	0.0563	0.0457	0.4432	0.0564	0.5433
Yale (64×64)	0.0431	0.0455	0.0877	0.0978	0.5524	0.0857	0.4367
Yale B	1.3455	1.3476	4.578	4.869	5.211	4.973	5.479
MNIST (2000)	1.4453	1.4768	5.436	5.556	5.967	5.674	6.241
MNIST (10000)	12.478	12.675	45.324	46.758	46.893	46.566	46.784

TABLE VIII
PERFORMANCE COMPARISON (RATE, STD%) OF STATE-OF-THE-ART ALGORITHMS ON BENCHMARK IMAGE DATASETS.

Datasets	KELM [3]		SBELM [9]		ECS-ELM [10]		DQLS-SVM [53]		**AT-ELM$_1$**		**AT-ELM$_2$**		**AT-ELM$_{\text{Orth},1}$**		**AT-ELM$_{\text{Orth},2}$**	
	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std	Rate	Std
AR	87.22	0.77	90.45	0.68	92.7	0.65	**92.82**	0.52	92.31	0.65	92.66	0.57	92.29	0.50	92.67	0.50
USPS	92.87	0.78	92.44	0.70	92.56	0.68	93.22	0.58	**93.76**	0.58	93.44	0.76	93.22	0.65	93.47	0.66
ORL (32×32)	87.02	1.74	94.44	1.67	93.45	1.72	94.88	1.65	95.28	1.44	94.97	1.67	95.21	1.66	**95.36**	1.44
ORL (64×64)	94.76	1.21	95.21	1.14	95.86	0.98	96.21	0.85	96.72	1.21	96.94	1.04	**97.12**	0.77	96.88	0.56
Yale (32×32)	75.87	5.22	80.11	4.87	82.34	4.57	83.22	4.43	82.87	4.32	82.87	4.76	83.65	4.46	**84.12**	4.27
Yale (64×64)	84.57	3.86	83.66	3.78	86.24	3.12	90.15	3.16	89.88	3.11	91.21	2.56	90.34	3.22	**91.76**	2.88
Yale B	91.22	1.02	92.75	0.87	95.62	0.76	97.05	0.64	97.11	0.73	97.21	0.66	**97.54**	0.56	97.51	0.45
MNIST (2000)	87.43	0.89	85.88	1.04	87.34	0.97	88.22	0.76	88.43	0.57	**88.79**	0.34	87.95	1.03	88.12	0.77
MNIST (10000)	93.55	0.21	93.76	0.18	94.92	0.14	**95.22**	0.07	94.97	0.10	94.90	0.08	94.88	0.12	94.87	0.10

training samples is $[2000 : 200 : 4000]$; for AR, USPS, ORL (64×64), and Yale (64×64) is $[1000 : 200 : 3000]$; for Yale (32×32), Yale B, and MNIST with 2,000 training samples is $[400 : 200 : 2000]$; and for ORL (32×32) is $[100 : 100 : 1000]$. Both the Sigmoid and Hyperbolic tangent functions served as the activation function.

Fig. 7 shows the performance comparison on the AR (random feature), ORL (32×32), Yale (32×32) and Yale B databases, all under the first setups. It is obvious that the AT-ELM-based algorithms consistently outperform the three competitive ELM methods on all four datasets. It is worthwhile to point out that the ELM and RELM fail to recognize the AR faces with random features, as shown in Fig. 7 (a). The main reason for the failure is that both the ELM and RELM suffer the same hidden node net input distribution issue as depicted in Figs. 2 and 4. Although the LOO cross-validation in RELM improves the classification accuracy, the recognition performance is still worse than those of AT-ELMs for all cases. At the same time, a similar observation to those in Sections IV-A and IV-B is shown in Fig. 7, i.e., adopting orthogonal random hidden node parameters in the AT-ELM usually achieves the best results among all of the four AT-ELM-based algorithms.

In addition, the average classification rates obtained in experiments on all of the 6 benchmark databases under setups other than those used for comparison in Fig. 7 are given in Table VI. The best and second-best results are highlighted by boldface and underlined in the table. It is apparent that for the Sigmoid function, the proposed AT-ELMs always achieve the best recognition performance. On 5 of the total 9 datasets, the AT-ELM-based algorithms offer $5.45\% \sim 8.25\%$ improvements over the best results obtained by the four competitive ELM methods, while on the remaining 4 datasets, the increments in the recognition rates by AT-ELMs are approximately

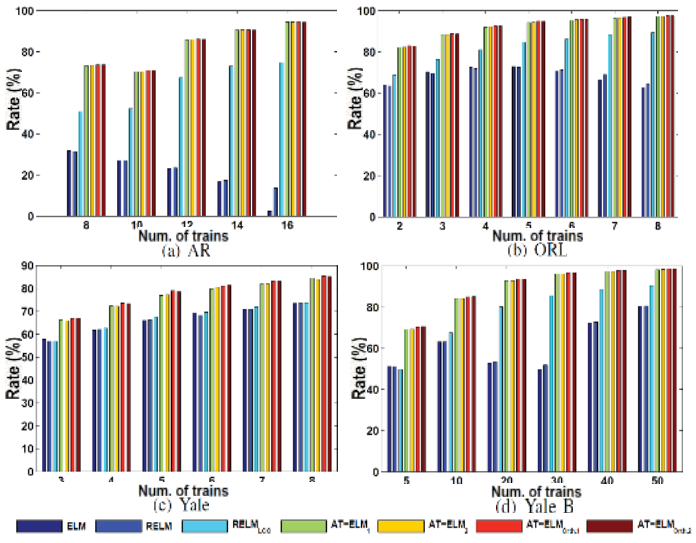


Fig. 7. Classification rate on the AR (random feature), ORL (32×32), Yale (32×32) and extended Yale B databases with the Sigmoid function.

$0.89\% \sim 2.36\%$. To show the computational complexity of the proposed AT-ELMs, the training time on all of the image datasets is recorded in Table VII for comparison. Similar to the observations in Table III of Section IV-A, compared to the computational time spent by searching the optimal regularization parameter λ in LOO, the time for updating the affine parameters in **Algorithm 1** is very short.

In addition to the Sigmoid function, we also tested the affine transformation hyperbolic tangent function as the activation function in our AT-ELM algorithms in all of the experiments in Sections IV-A$\sim$IV-C. The affine transformation hyperbolic tangent function is expressed as $f(x, s, t) = \frac{\sinh(s \cdot x + t)}{\cosh(s \cdot x + t)}$, where $\sinh$ and $\cosh$ denote the hyperbolic sine and hyperbolic cosine functions, respectively. For each experiment, similar observations to those with the Sigmoid function were obtained. Due to page limitations, only the results of the benchmark image recognition datasets are reported in Table VI. As expected, the proposed algorithms achieve similar performance and enhancement by using the affine transformation hyperbolic tangent function.

To further demonstrate the superiority of the proposed AT-ELMs, we compare the recognition performance on the benchmark image datasets with several other state-of-the-art algorithms, including the KELM [3], the sparse Bayesian ELM (SBELM) [9], the evolutionary cost-sensitive ELM (ECS-ELM) [10] and the data density-dependent quantized least squares SVM (DQLS-SVM) [53]. The SBELM improves the original ELM by first estimating the marginal likelihood of the network outputs and then pruning the redundant hidden neurons during learning [9]. The ECS-ELM enhances the generalization performance by introducing an adaptive evolutionary cost-sensitive matrix to quantify misclassified samples in ELM [10]. The DQLS-SVM algorithm employs a data density-dependent vector quantization method for input sample representation and adopts the LS-SVM for quantized data classification. To have a fair comparison, the number of hidden nodes used in the KELM, SBELM, and ECS-ELM are the same as that in the AT-ELMs used in this subsection. The Gaussian kernel function is adopted in the KELM and DQLS-SVM, where the cost parameter and kernel parameter are both optimized within $\left\{2^{-24}, 2^{-23}, \cdots, 2^{24}, 2^{25}\right\}$. For SBELM and the four proposed AT-ELMs, the Sigmoid function is adopted. As suggested in [53], in DQLS-SVM, the shrinkage parameter δ that controls the number of points within the region of the code center for density representation, is set to the value of 1% for the entire sample number in each dataset. For all of the datasets, the training and testing sample ratio is set to the same value given at the beginning of this subsection. Table VIII shows the comparison of the average recognition rate (Rate), where the best and second-best results are highlighted by boldface and underlined, respectively. One can observe that the proposed AT-ELMs show the best recognition performance in 7 of the 9 datasets, and the DQLS-SVM has the highest recognition rate in the remaining two datasets.

V. Conclusions

Neglecting the hidden layer net inputs distribution and using a fixed activation function in the extreme learning machine (ELM) can lead to poor generalization performance. The reason behind this effect is that using random parameters without tuning in the ELM can move most hidden layer net inputs into the saturated or linear regions of the activation function. To overcome these deficiencies, an effective ELM algorithm with an affine transformation activation function, or in short, the affine transformation ELM (AT-ELM), has been developed in this paper. This algorithm enforces the hidden layer outputs to be approximately uniformly distributed within the output range of the activation function by adjusting the scaling and translation parameters of the affine transformation based on the maximum entropy theory. Extensive experiments on nonlinear function regression, real-world benchmark dataset classification and image recognition have been conducted. The results show that the AT-ELM-based algorithms outperform the original ELM, RELM, KELM, and RELM_{LOO}. Experiments on 6 benchmark image recognition datasets also show that the proposed AT-ELMs outperform several other state-of-the-art algorithms. In addition, among all AT-ELMs, employing orthogonal hidden node parameters has usually achieved the best generalization performance.

Appendix A
Derivation of (11)

Since all of the elements in $\mathbf{W}$, $\mathbf{x}$ and $\mathbf{b}$ are *i.i.d.* and mutually independent, for each element v_i in the hidden layer net input matrix $\mathbf{V}$, the expectation is expressed as

$$\begin{aligned} E(v_i) &= E(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) \\ &= E(\mathbf{w}_i \cdot \mathbf{x}_j) + E(b_i) \\ &= L \cdot E(w_i) E(x_i) + E(b_i). \end{aligned}$$

For w_i and b_i obeying the uniform distribution $\mathcal{U}[-1, 1]$ or standard normal distribution, we have $E(w_i) = 0$ and

$E(b_i) = 0$. Thus, $E(v_i) = 0$. The variance of v_i is then computed as $var(v_i) = E\left(v_i^2\right)$, which is

$$\begin{aligned} E\left(v_i^2\right) &= E\left[\left(\mathbf{w}_i \cdot \mathbf{x}_j\right)^2\right] + var(b_i) \\ &= \sum_{i=1}^{L} E\left(w_i^2\right) E\left(x_i^2\right) + var(b_i) \\ &= L \cdot var(w_i) E\left(x_i^2\right) + var(b_i). \end{aligned}$$

REFERENCES

[1] B. Igelnik and Y. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1320–1329, 1995.

[2] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[3] G.-B. Huang, "What are extreme learning machines? filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle," *Cognitive Computation*, vol. 7, pp. 263–278, 2015.

[4] G.-B. Huang, Z. Bai, L. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 18–29, 2015.

[5] X. Liu, S. B. Lin, and Z. B. Xu, "Is extreme learning machine feasible? a theoretical assessment (part I)," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 7–20, 2015.

[6] S. B. Lin, X. Liu, J. Fang, and Z. B. Xu, "Is extreme learning machine feasible? a theoretical assessment (part II)," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 21–34, 2015.

[7] J. C. Principe and B. Chen, "Universal approximation with convex optimization: Gimmick or reality?," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 68–77, 2015.

[8] C. M. Wong, C. M. Vong, P. K. Wong, and J. Cao, "Kernel-based multilayer extreme learning machines for representation learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 3, pp. 757–762, 2018.

[9] J. Luo, C. M Vong, and P.-K. Wong, "Sparse bayesian extreme learning machine for multi-classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 836–843, 2014.

[10] L. Zhang and D. Zhang, "Evolutionary cost-sensitive extreme learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–16, 2017.

[11] A. Akusok, K. Bjork, Y. Miche, and A. Lendasse, "High-performance extreme learning machines: A complete toolbox for big data applications," *IEEE Open Access*, vol. 3, pp. 1011–1025, 2015.

[12] A. Iosifidis, A. Tefas, and I. Pitas, "Graph embedded extreme learning machine," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 311–324, 2016.

[13] Y. Yang, Y. Wang, Q. M. Jonathan Wu, X. Lin, and M. Liu, "Progressive learning machine: A new approach for general hybrid system approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1855–1874, Sept 2015.

[14] N. D. Vanli, M. O. Sayin, I. Delibalta, and S. S. Kozat, "Sequential nonlinear learning for distributed multiagent systems via extreme learning machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 546–558, March 2017.

[15] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809–821, 2016.

[16] M. Duan, K. Li, X. Liao, and K. Li, "A parallel multiclassification algorithm for big data using an extreme learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–15, 2017.

[17] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum class variance extreme learning machine for human action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 11, pp. 1968–1979, 2013.

[18] Y. Bazi, N. Alajlan, F. Melgani, H. AlHichri, S. Malek, and R. R. Yager, "Differential evolution extreme learning machine for the classification of hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 6, pp. 1066–1070, 2014.

[19] W. M. Czarnecki, "Weighted tanimoto extreme learning machine with case study in drug discovery," *IEEE Computational Intelligence Magazine*, vol. 10, no. 3, pp. 19–29, 2015.

[20] H. Zou, B. Huang, X. Lu, H. Jiang, and L. Xie, "A robust indoor positioning system based on the procrustes analysis and weighted extreme learning machine," *IEEE Transactions on Wireless Communications*, vol. 15, no. 2, pp. 1252–1266, 2016.

[21] J. Cao, Y. Zhao, X. Lai, M. Ong, C. Yin, Z. Koh, and N. Liu, "Landmark recognition with sparse representation classification and extreme learning machine," *Journal of the Franklin Institute*, vol. 352, no. 10, pp. 4528–4545, 2015.

[22] J. Cao, J. Hao, X. Lai, C.-M. Vong, and M. Luo, "Ensemble extreme learning machine and sparse representation classification algorithm," *Journal of the Franklin Institute*, vol. 353, no. 17, pp. 4526–4541, 2016.

[23] J. Cao, T. Chen, and J. Fan, "Landmark recognition with compact bow histogram and ensemble ELM," *Multimedia Tools and Applications*, vol. 75, no. 5, pp. 2839–2857, 2016.

[24] Z. Huang, Y. Yu, J. Gu, and H. Liu, "An efficient method for traffic sign recognition based on extreme learning machine," *IEEE Transactions on Cybernetics*, vol. 47, no. 4, pp. 920–933, 2017.

[25] J. Cao, W. Wang, J. Wang, and R. Wang, "Excavation equipment recognition based on novel acoustic statistical features," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4392–4404, 2017.

[26] H. Liu, Y. Yu, F. Sun, and J. Gu, "Visual-tactile fusion for object recognition," *IEEE Transactions on Automation Science and Engineering*, vol. PP, no. 99, pp. 1–13, 2016, DOI:10.1109/TASE.2016.2549552.

[27] J. Cao, W. Huang, T. Zhao, J. Wang, and R. Wang, "An enhance excavation equipments classification algorithm based on acoustic spectrum dynamic feature," *Multidimensional Systems and Signal Processing*, vol. 28, no. 3, pp. 921–943, 2017.

[28] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *G. Orr and K. Muller (eds), Neural Networks: Tricks of the trade, Springer*, 1998.

[29] W. Simon and N. Hermann, "A convergence analysis of log-linear training," in *J. Shawe-Taylor, R. Zemel, P. Barlett, F. Pereira and K. Weiberger (eds.), Advances in Neural Information Processing Systems*, 2011, pp. 657–665.

[30] W. Simon, R. Alexander, S. Ralf, and N. Hermann, "Mean-normalized stochastic gradient for large-scale deep learning," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2014, pp. 180–184.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR, arXiv:1502.03167*, 2015.

[32] T. Salimans and D. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *CoRR, arXiv:1602.07868*, 2016.

[33] D. Arpit, Y. Zhou, B. Kota, and V. Govindaraju, "Normalization propagation: A parametric technique for removing internal covariate shift in deep networks," *eprint arXiv:1603.01431*, 2016.

[34] P. Potocnik and E. Govekar, "Practical considerations in training extreme learning machines," in *Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (EANN), Rhodes Island, Greece, Sep. 25-28*, 2015, pp. 1–5.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Conference and Workshop on Neural Information Processing Systems*, 2012, vol. 25.

[36] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, 2013.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[38] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolution network," in *International Conference on Machine Learning*, 2015.

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

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

[41] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[42] E. T. Jaynes, "Information theory and statistical mechanics," *Physical Review. Series II*, vol. 106, no. 4, pp. 620–630, 1957.

[43] O. Kallenberg, *Foundations of Modern Probability*, Berlin: Springer

Verlag, 2001.
[44] D. Donoho and I. Johnston, "Ideal spatial adaptive via wavelet shrinkage," *Biometrika*, vol. 81, pp. 425–455, 1994.
[45] Y. Ghanbari and M. Karami-Mollaei, "A new approach for speech enhancement based on the adaptive thresholding of the wavelet packets," *Speech Communication*, vol. 48, pp. 927–940, 2006.
[46] R. Myers, *Classical and Modern Regression with Applications*, Duxbury, 1990.
[47] Y. Miche, M. Heeswijk, P. Bas, O. Simula, and A. Lendasse, "TROP-ELM: A double-regularized ELM using LARS and tikhonov regularization," *Neurocomputing*, vol. 74, pp. 2413–2421, 2011.
[48] M. Heeswijk and Y. Miche, "Binary/ternary extreme learning machines," *Neurocomputing*, vol. 149, pp. 187–197, 2015.
[49] J. Cao, K. Zhang, M. Luo, C. Yin, and X. Lai, "Extreme learning machine and adaptive sparse representation for image classification," *Neural Networks*, vol. 81, pp. 91–102, 2016.
[50] L. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with ELMs for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.
[51] J. Cao, Z. Lin, and G.-B. Huang, "Composite function wavelet neural networks with extreme learning machine," *Neurocomputing*, vol. 73, no. 7-9, pp. 1405–1416, 2010.
[52] J. Cao, Z. Lin, and G.-B. Huang, "Composite function wavelet neural networks with differential evolution and extreme learning machine," *Neural Processing Letters*, vol. 33, no. 3, pp. 251–265, 2011.
[53] S. Nan, L. Sun, B. Chen, Z. Lin, and K.-A.n Toh, "Density-dependent quantized least squares support vector machine for large data sets," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 1, pp. 94–106, 2017.

Jiuwen Cao received the B.Sc. and M.Sc. degrees from the School of Applied Mathematics, University of Electronic Science and Technology of China, Chengdu, China, in 2005 and 2008, respectively, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, in 2013. From 2012 to 2013, he was a Research Fellow with NTU. He has been an Associate Professor with Hangzhou Dianzi University, China, since 2013. He has published over 90 papers in top-tier journals and conferences. His current research interests include machine learning, neural networks, and array signal processing. He is an Associate Editor of *IEEE Transactions on Circuits and Systems I: Regular paper*, *Multidimensional Systems and Signal Processing*, and *Memetic computing*. He has served as a Guest Editor of *Journal of the Franklin Institute* and *Multidimensional Systems and Signal Processing*. He served as the Program Chair of the 5th International Conference on ELM, Singapore, in 2014, and the Publication Chair of the 6th∼9th International Conference on ELM, in 2015 to 2018, respectively, the Publication Chair of the IEEE ISPACS 2017.

Kai Zhang received the M.Sc. degree in applied mathematics from China Jiliang University, Hangzhou, China, in 2014. He is currently pursuing the Ph.D. degree in computer science and technology at Harbin Institute of Technology, Harbin, China, under the supervision of Prof. Wangmeng Zuo and Prof. Lei Zhang. His research interests include machine learning and image processing.

Hongwei Yong received the B.Sc. and M.Sc degree from Xian Jiaotong University, Xian, China, in 2013 and 2016, respectively. Now he studies at The Hong Kong Polytechnic University for his Ph.D. degree. His current research interests include deep learning, video analysis, and image modeling.

Xiaoping Lai received the B. S. and M. S. degrees in physics from Shandong University in June 1985 and June 1988, and the Ph. D. degree in applied mathematics from Shandong University in June 2000. From 1988 to 2008, he was with Shandong University, China, where he became a professor in 2001, was the Director of the Department of Information Science and Control Engineering for 2003-2004, and was the Dean of the School of Information Engineering for 2004-2007. Since 2008, he has been with Hangzhou Dianzi University, China, as a distinguished professor of Control Science and Engineering, where he was the Dean of the Automation School for 2010-2015. Dr. Lai has served as an Associate Editor of *Multidimensional Systems and Signal Processing* since June 2014. His general research interests lie in the areas of optimization methods and applications in digital signal processing and control systems. His current research focuses on optimal designs of one- and multidimensional digital filters and filter banks, machine learning, and modeling and optimization for big data.

Badong Chen received the Ph.D. degree in computer science and technology from Tsinghua University in 2008. Currently he is a professor at the Institute of Artificial Intelligence and Robotics (IAIR), Xian Jiaotong University. His research interests are in signal processing, machine learning and their applications to cognitive science and neural engineering. He has published 2 books, 4 chapters, and over 200 papers in various journals and conference proceedings. Dr. Chen is an IEEE Senior Member, a Technical Committee Member of *IEEE SPS Machine Learning for Signal Processing (MLSP)* and *IEEE CIS Cognitive and Developmental Systems (CDS)*, and an associate editor of *IEEE Transactions on Cognitive and Developmental Systems*, *IEEE Transactions on Neural Networks and Learning Systems* and *Journal of The Franklin Institute*, and has been on the editorial board of *Entropy*.

Zhiping Lin received the B.Eng. degree in control engineering from South China Institute of Technology, Canton, China in 1982 and the Ph.D. degree in information engineering from the University of Cambridge, England in 1987. Since 1999, he has been with Nanyang Technological University (NTU), Singapore. Dr. Lin was the Editor-in-Chief of *Multidimensional Systems and Signal Processing* for 2011-2015. He was an Associate Editor of *Circuits, Systems and Signal Processing* for 2000-2007 and an Associate Editor of *IEEE Transactions on Circuits and Systems - II* for 2010-2011. He was a reviewer for *Mathematical Reviews* for 2011-2013. He is currently a Subject Editor of the *Journal of the Franklin Institute*. His research interests include multidimensional systems and signal processing, statistical and biomedical signal processing, and machine learning. He is the co-author of the 2007 Young Author Best Paper Award from the IEEE Signal Processing Society, Distinguished Lecturer of the IEEE Circuits and Systems Society for 2007-2008, and received the Best Paper Awards at ELM 2015 and ELM 2017.