SCIENCE CHINA
Information Sciences

• REVIEW •
Special Focus on High-Speed Signal Processing

February 2015, Vol. 58 020301:1–020301:16
doi: 10.1007/s11432-014-5269-3

Extreme learning machines: new trends and applications

DENG ChenWei[1], HUANG GuangBin[2*], XU Jia[1] & TANG JieXiong[1]

[1]*School of Information and Electronics, Beijing Institute of Technology, Beijing* 100081*, China;*
[2]*School of Electronic Engineering, Nanyang Technological University, Singapore*

Received October 29, 2014; accepted November 24, 2014

Abstract Extreme learning machine (ELM), as a new learning framework, draws increasing attractions in the areas of large-scale computing, high-speed signal processing, artificial intelligence, and so on. ELM aims to break the barriers between the conventional artificial learning techniques and biological learning mechanism and represents a suite of machine learning techniques in which hidden neurons need not to be tuned. ELM theories and algorithms argue that "random hidden neurons" capture the essence of some brain learning mechanisms as well as the intuitive sense that the efficiency of brain learning need not rely on computing power of neurons. Thus, compared with traditional neural networks and support vector machine, ELM offers significant advantages such as fast learning speed, ease of implementation, and minimal human intervention. Due to its remarkable generalization performance and implementation efficiency, ELM has been applied in various applications. In this paper, we first provide an overview of newly derived ELM theories and approaches. On the other hand, with the ongoing development of multilayer feature representation, some new trends on ELM-based hierarchical learning are discussed. Moreover, we also present several interesting ELM applications to showcase the practical advances on this subject.

Keywords extreme learning machine, fast learning, high-speed and real-time signal processing, large-scale computing, big-data, feature representation

Citation Deng C W, Huang G B, Xu J, et al. Extreme learning machines: new trends and applications. Sci China Inf Sci, 2015, 58: 020301(16), doi: 10.1007/s11432-014-5269-3

1 Introduction

Feedforward neural networks (NNs) [1–5] have been widely used in various areas of machine learning in the past decades. NN theories [6–9] show that single hidden layer feedforward neural networks (SLFNs) with additive or radial basis function (RBF) hidden nodes can work as universal approximators provided that all the parameters of the networks are adjustable. Theoretically, SLFNs could approximate nonlinear mappings from the input samples and establish models for a large amount of natural or artificial signals that conventional techniques may fail to handle with. The most representative training method for SLFNs is called "back propagation" (BP) algorithm, which calculates the gradient of a loss function with respect to all the weights in the network and updates the weights for minimizing the loss function. However, the parameter tuning of BP-based NNs is usually time consuming and cannot handle the overfitting problem.

* Corresponding author (email: egbhuang@ntu.edu.sg)

It is not surprising that the training process may take several hours/days or even more time using BP-like gradient-based methods. Thus, SLFNs or BP-based NNs are feasible but usually "expensive and costly".

Nowadays, we are entering the era of "big-data", and as the development of high-speed signal processing, fast and efficient feature learning and signal representation is becoming an emergent research topic. Extreme learning machine (ELM) [10–12] is one of the leading trends for fast learning. Unlike the other traditional learning algorithms, for example, BP-based NNs [1], or support vector machine (SVM) [13], the parameters of hidden layers of ELM are randomly established and need not be tuned, thus the training of hidden nodes can be established before the inputs are acquired. Also, Huang et al. [14–16] have proved that the SLFNs with randomly generated hidden neurons and output weights computed by ridgre regression still maintain the universal approximation capability of SLFNs, even the input weights are randomly assigned and prefixed.

Therefore, ELM stands out from other learning methods with the following unique characteristics: extremely fast training, good generalization, and universal approximation capability. In other words, ELM achieves stable solutions for the SLFNs and has been demonstrated to have excellent learning accuracy and speed in various applications, such as ship detection [17], super resolution [18], image quality assessment [19,20], and human action recognition [21].

Recently, ELM has been extended to different research fields and gained great progresses. An incremental ELM (I-ELM) is proposed in [22], where the hidden nodes are incrementally constructed and the output weights are determined accordingly. Another ELM variant has also been developed in the case of online sequential learning [23,24], and the input data can be processed sequentially, that is, one-by-one or chunk-by-chunk with fixed or varying chunk size. Huang et al. [25] extend ELM for both semisupervised and unsupervised tasks based on the manifold regularization, and the unlabeled or partially labeled samples are clustered using ELM.

However, the aforementioned works fail to avoid the shortcoming of SLFNs. That is, when dealing with natural scenes (e.g., visual and acoustic signals) or practical applications (e.g., image classification, voice recognition), the original ELM and/or its variants cannot achieve a satisfactory feature learning performance, due to its inherent shallow architecture of SLFNs. A multilayer learning architecture using ELM-based autoencoder as its building block is developed in [26]. The original inputs are decomposed into multiple hidden layers, and the outputs of the previous layer are used as the inputs of the current one. The autoencoders are stacked layer by layer in the hierarchical structure, and the encoded outputs are fed into the last layer for decision making using the supervised least mean square optimization.

From the above-mentioned discussions, it is obvious that the theories and applications of ELM have been extensively studied in the past few years, though ELM itself is still a new learning technique. Therefore, a systematic and up-to-date review of the ELM theoretical and practical developments will be of great interest to the related research community. In this paper, we aim at a comprehensive overview of the current research status in the area of ELM and introduce its new research trends, as well as the state-of-the-art applications.

The rest of this paper is organized as follows. Section 2 introduces the current research status of ELM, including the fundamental theories and implementations of ELM, and other enhanced ELM variants. Section 3 describes the new leading trend in ELM: multilayer learning with meaningful representations. Section 4 introduces several real-world applications incorporating ELM theories, while Section 5 concludes this paper.

2 Current status: single layer no-propagating learning

Over the past few decades, conventional signal computing and artificial intelligence techniques meet some inevitable bottlenecks in terms of algorithmic learning: (1) how to avoid the time-consuming computation while the data dimension is increasing rapidly; and (2) how to establish an estimation before any meaningful observations (like human does). As a newly emerging learning architecture, ELM provides fast solutions and sheds some light upon the said issues that how to effectively deal with these computational bottlenecks. It adopts a single layer scheme that no propagation is needed and can be solved by simple

least square method. Also, its input weights are randomly assigned and fully independent of the input samples.

To help others gain a better understanding of the concept and strategy of ELM, a review of the current works on ELM is presented in this section. The original ELM is to be first introduced, which includes the theory of universal approximation capability for randomly generated network and the implementation details of ELM. Then, its typical variants are discussed, including incremental learning, sequential learning, and semisupervised/unsupervised learning.

2.1 Original ELM

Suppose that SLFNs with n hidden nodes can be represented by the following equation:

$$f_n(x) = \sum_{i=1}^{n} G_i(x,\ a_i, b_i) * \beta_i,\ a_i \in R^d,\ b_i,\ \beta_i \in R, \tag{1}$$

where $G_i(\cdot)$ denotes the ith hidden node activation function, a_i is the input weight vector connecting the input layer to the ith hidden layer, b_i is the bias weight of the ith hidden layer, and β_i is the output weight. For additive nodes with activation function g, G_i is defined as follows:

$$G_i(x, a_i, b_i) * \beta_i = g(a_i \times x + b_i), \tag{2}$$

and for RBF nodes with activation function g, G_i is defined as

$$G_i(x, a_i, b_i) * \beta_i = g(b_i \|x - a_i\|). \tag{3}$$

Huang et al. [14] have proved that the SLFNs are able to approximate any continuous target functions over any compact subset $X \in R^d$ with the above random initialized adaptive or RBF nodes. Let $L^2(X)$ be a space of functions f on a compact subset X in the d-dimensional Euclidean space R^d such that $|f|^2$ are integrable, that is, $\int_X |f(x)|^2 \mathrm{d}x < \infty$. For $u, v \in L^2(X)$, the inner product $\langle u, v\rangle$ is defined by

$$\langle u, v\rangle = \int_X u(x)v(x)\mathrm{d}x. \tag{4}$$

The norm in $L^2(X)$ space is denoted as $\|\cdot\|$, and the closeness between network function f_n and the target function f is measured by the $L^2(X)$ distance:

$$\|f_L - f\| = \left(\int_X |f_n(x) - f(x)|^2 \mathrm{d}x\right)^2. \tag{5}$$

Theorem 1. Given any bounded nonconstant piecewise continuous function g: $R \to R$, if span $\{G(a, b, x)$: $(a, b) \in R^d \times R\}$ is dense in L^2, for any target function f and any function sequence $g_L(x) = G(a_L, b_L, x)$ randomly generated based on any continuous sampling distribution, $\lim_{n\to\infty} \|f - f_n\| = 0$ holds with probability one if the output weights β_i are determined by ordinary least square to minimize $\|f(x) - \sum_{i=1}^{L} \beta_i g_i(x)\|$.

The theorem above shows that randomly generated networks with the outputs being solved by least mean square are able to maintain the universal approximation capability, if and only if the activation function g is nonconstant piecewise and span $\{G(a, b, x) : (a, b) \in R^d \times R\}$ is dense in L^2.

Based on this theorem, ELM can be established for fast learning. Given a training set $N = \{(x_i, t_i)|x_i \in R^n,\ t_i \in R^m,\ i = 1, \ldots, L\}$, where x_i are the training data, t_i represents the class label of each sample, and L denotes the number of hidden nodes. The ELM training algorithm can be summarized as follows [11]:

(1) Randomly assign the hidden node parameters: the input weights w_i and biases b_i, $i = 1, \ldots, L$.

(2) Calculate the hidden layer output matrix H.

(3) Obtain the output weight

$$\beta = H^{\dagger}T, \tag{6}$$

where $T = [t_1, \ldots, t_N]^{\mathrm{T}}$ and $H^{\dagger}$ is the Moore-Penrose (MP) generalized inverse of matrix H.

The orthogonal projection method [27] can be efficiently used for the calculation of MP inverse: $H^{\dagger} = (H^{\mathrm{T}}H)^{-1}H^{\mathrm{T}}$, if $H^{\mathrm{T}}H$ is nonsingular; or $H^{\dagger} = H^{\mathrm{T}}(H^{\mathrm{T}}H)^{-1}$, if HH^{T} is nonsingular. According to the ridge regression theory [28], it was suggested that a positive value $1/\lambda$ be added to the diagonal of $H^{\mathrm{T}}H$ or HH^{T} in the calculation of the output weights β, and by doing so, the resultant solution is more stable and has better generalization performance. That is, to improve the stability of ELM, we can have

$$\beta = H^{\mathrm{T}}\left(\frac{1}{\lambda} + HH^{\mathrm{T}}\right)^{-1} T, \tag{7}$$

and the corresponding output function of ELM is:

$$f(x) = h(x)\beta = h(x)H^{\mathrm{T}}\left(\frac{1}{\lambda} + HH^{\mathrm{T}}\right)^{-1} T, \tag{8}$$

or we can have

$$\beta = \left(\frac{1}{\lambda} + HH^{\mathrm{T}}\right)^{-1} H^{\mathrm{T}}T, \tag{9}$$

and the corresponding output function of ELM is

$$f(x) = h(x)\beta = h(x)\left(\frac{1}{\lambda} + HH^{\mathrm{T}}\right)^{-1} H^{\mathrm{T}}T. \tag{10}$$

Table 1 shows the performance comparison among SVM, LS-SVM [29], and ELM with Gaussian kernel (please refer to [11] for more details). It can be seen that ELM achieves much better generalization performance in multiclass classification cases than those of SVM and LS-SVM.

2.2 Incremental ELM (I-ELM)

An incremental version of ELM learning algorithm is proposed in [14], denoted as I-ELM. It adopts an incremental construction method that the hidden nodes can be incrementally added (before the training there is no node in the network).

The training of I-ELM uses an estimated parameter based on the training samples and the residual error:

$$\beta_n = \frac{E \times H^{\mathrm{T}}}{H \times H^{\mathrm{T}}} = \frac{\sum_{p=1}^{N} e(p)h(p)}{\sum_{p=1}^{N} h^2(p)}, \tag{11}$$

where $h(p)$ is the output of new hidden node for the pth input training sample, and $e(p)$ is the corresponding residual error before this new hidden node is added. $H = [h(1), \ldots, h(N)]^{\mathrm{T}}$ is the activation vector of the new node for the whole N samples, and $E = [e(1), \ldots, e(N)]^{\mathrm{T}}$ is the residual vector before this new hidden node is added. Practically, a maximum number of hidden nodes are usually given rather than reaching zero approximation error.

Based on the above estimation, the incremental constructive method for I-ELM can be summarized as follows: Given a training set $N = \{(x_i, t_i) | x_i \in R^n,\ t_i \in R^m,\ i = 1, \ldots, L\}$, activation function $g(x)$, maximum node number $\tilde{N}_{\max}$, and expected learning accuracy ϵ.

Step 1 Initialization:

Let $\tilde{N} = 0$ and residual error $E = t$.

Step 2 Incremental Learning: while $\tilde{N} < \tilde{N}_{\max}$ and $\|E\| > \epsilon$,

(a) increase the number of hidden nodes $\tilde{N} = \tilde{N} + 1$;

(b) assign random input weight $a_{\tilde{N}}$ and bias $b_{\tilde{N}}$ for the newly added hidden node;

(c) calculate the output weight $\beta_{\tilde{N}}$ for the new hidden nodes:

$$\beta_{\tilde{N}} = \frac{E \times H_{\tilde{N}}^{\mathrm{T}}}{H_{\tilde{N}} \times H_{\tilde{N}}^{\mathrm{T}}}; \tag{12}$$

(d) update the corresponding residual error and go to Step 2(a):

$$E = E - \beta_{\tilde{N}} \times H_{\tilde{N}}. \tag{13}$$

Table 1 Performance comparison of SVM, least square SVM (LS-SVM), and ELM: multiclass data sets

Datasets	SVM			LSSVM			Extreme learning machine								
							Gaussian kernel			Sigmoid additive node			Multiquadrics RBF node		
	Testing		Training	Testing		Training	Testing		Training	Testing		Training	Testing		Training
	Rate (%)	Dev. (%)	Time (s)	Rate (%)	Dev. (%)	Time (s)	Rate (%)	Dev. (%)	Time (s)	Rate (%)	Dev. (%)	Time (s)	Rate (%)	Dev. (%)	Time (s)
										$f(x) = h(x)H^{\mathrm{T}}(\frac{1}{C} + HH^{\mathrm{T}})^{-1}T$					
Iris	95.12	2.45	0.075	96.28	2.36	0.0021	96.04	2.37	0.0022	97.6	2.29	0.0156	97.33	2.12	0.0161
Glass	67.83	4.67	0.2871	67.22	5.04	0.0097	68.41	4.81	0.0026	67.12	4.99	0.0262	66.89	4.97	0.0264
Wine	98.37	1.41	0.075	97.63	1.82	0.0043	98.48	1.7	0.0019	98.47	1.81	0.0222	98.57	1.26	0.0206
Ecoli	86.56	3.65	0.2469	85.93	2.82	0.0244	87.48	2.8	0.008	87.23	2.88	0.053	87.79	2.74	0.054
Vowel	56.28	0	2.172	52.81	0	0.3290	58.66	0	0.0688	53.73	0.91	0.2187	54.75	0.89	0.2231
Vehicle	84.37	1.71	1.5144	83.19	1.93	0.2029	83.16	1.89	0.0831	83.48	1.78	0.2557	83.95	1.85	0.2547
										$f(x) = h(x)(\frac{1}{C} + HH^{\mathrm{T}})^{-1}H^{\mathrm{T}}T$					
Segment	96.53	0.64	14.30	96.12	0.75	4.302	96.53	0.54	0.9272	96.07	0.69	1.889	95.54	0.41	1.070
Satimage	89.75	0	698.4	90.05	0	82.67	92.35	0	15.70	89.8	0.32	2.808	89.06	0.38	2.803
DNA	92.86	0	7732	93.68	0	6.359	96.29	0	2.156	93.81	0.24	1.586	94.81	0.32	1.597
*Adult	92.87	0.26	302.9	93.12	0.27	335.838	97.41	0.13	41.89	93.51	0.15	0.7881	93.96	0.15	1.4339
*Shuttle	99.74	0	2864.0	99.82	0	24767.0	99.91	0	4029.0	99.64	0.01	3.3379	99.65	0.02	5.5455
*USPS	96.14	0	12460	96.76	0	59.1357	98.9	0	9.2784	96.28	0.28	0.6877	97.25	0.24	0.9008

2.3 Online sequential ELM (OS-ELM)

In some real-time applications, the input data need to be updated online. In this scenario, we need to accordingly update the network with newly obtained samples. As the training data are grouped one-by-one or chunk-by-chunk, online sequential ELM (OS-ELM) [23] is preferred and retraining is not required whenever a new chunk of data are received. The sequential implementations of the least square solution of OS-ELM are as follows:

Step 1 Initialization:

Initialize the training using a small chunk of initial data samples $N_0 = \{(x_i, t_i)\}_{i=1}^{N_0}$ from the given training set $N = \{(x_i, t_i)|x_i \in R^n,\ t_i \in R^m,\ i = 1, \ldots, L\}$.

(a) Randomly generate the hidden node parameters $(a_i, b_i),\ i = 1, \ldots, L$.

(b) Calculate the initial hidden layer output matrix H_0.

(c) Estimate the initial output weight $\beta^{(0)} = P_0 H_0^{\mathrm{T}} T_0$, where $P_0 = (H_0^{\mathrm{T}} H_0)^{-1}$ and $T_0 = [t_1, \ldots, t_{N_0}]^{\mathrm{T}}$.

(d) Set $k = 0$.

Step 2 Sequential Learning:

(a) Present the $(k+1)$th chunk of new observations: $N_{k+1} = \{(x_i, t_i)\}_{i=\sum_{j=0}^{k} N_j + 1}^{\sum_{j=0}^{k+1} N_j}$, where N_{k+1} denotes the number of observations in the $(k+1)$th chunk.

(b) Calculate the partial hidden layer output matrix H_{k+1} for the $(k+1)$th chunk of data N_{k+1}.

(c) Calculate the output weight $\beta^{(k+1)}$:

$$P_{k+1} = P_k - P_k H_{k+1}^{\mathrm{T}} (I + H_{k+1} P_k H_{k+1}^{\mathrm{T}})^{-1} H_{k+1} P_k, \tag{14}$$

$$\beta^{(k+1)} = \beta^{(k)} + P_{k+1} H_{k+1}^{\mathrm{T}} (T_{k+1} - H_{k+1} \beta^{(k)}). \tag{15}$$

(d) Set $k = k + 1$, go to Step 2(a).

2.4 Semisupervised ELM (SS-ELM) and unsupervised ELM(US-ELM)

The original ELM and its aforementioned variants are primarily used for supervised learning tasks such as classification and regression. However, for some other tasks, for example, information retrieval and fault diagnosis, it may be time consuming and hard to collect large amount of labeled data. To tackle this problem, semisupervised ELM (SS-ELM) and unsupervised ELM(US-ELM) [25] are developed to deal with the cases that only few samples are labeled or the entire training set is unlabeled. In this subsection, we briefly introduce the implementation details of SS-ELM and US-ELM and compare them with relevant state-of-the-art approaches.

2.4.1 *SS-ELM*

In semisupervised case, we have few labeled data and plenty of unlabeled data. SS-ELM incorporates the manifold regularization to leverage unlabeled data to improve the classification accuracy when the labeled data are scarce. The SS-ELM algorithm is summarized as follows:

Given a training set that consists of labeled samples: $N_l = \{(x_i, t_i)|x_i \in R^n,\ t_i \in R^m,\ i = 1, \ldots, L\}$ and unlabeled samples: $N_u = \{(x_i)|x_i \in R^n,\ i = 1, \ldots, U\}$.

Step 1 Construct the graph Laplacian L [30] from both N_l and N_u.

Step 2 Initiate an ELM with N_h hidden nodes with random input weights and biases and calculate the output matrix H.

Step 3 Choose the hyperparameter C and λ.

Step 4 **if** the number of N_h is less than the number of N_l, compute the output weight β:

$$\beta = (H^{\mathrm{T}}(I_{N_h} + H^{\mathrm{T}} C H + \lambda H^{\mathrm{T}} L H)^{-1} H^{\mathrm{T}} C T, \tag{16}$$

where I is an identity matrix of dimension N_h, H is the output weight of the hidden nodes, and T is the label of the labeled samples N_l.

Table 2 Comparison of proposed SS-ELM, pure supervised and state-of-the-art semisupervised algorithms

Data set	Subset	SVM	ELM	TSVM	LapRLS	LapSVM	SS-ELM
G50C	μ	9.33±2.00	8.91±2.29	5.43±1.11	6.03±1.32	5.52±1.15	5.24±1.17
	ν	9.83±3.46	8.20±3.05	4.67±1.81	6.17±3.66	5.67±2.67	4.07±0.95
	τ	10.06±2.80	9.20±2.07	4.96±1.37	6.54±2.11	5.51±1.65	4.96±1.53
COIL20 (B)	μ	16.23±2.63	11.28±1.95	13.68±3.09	8.07±2.05	8.31±2.19	6.04±1.26
	ν	18.54±6.20	11.30±2.29	12.25±3.99	7.92±3.96	8.13±4.01	5.95±1.68
	τ	15.93±3.00	12.22±2.00	15.19±2.52	8.59±1.90	8.68±2.04	7.33±2.15
USPST (B)	μ	17.00±2.74	17.49±2.44	22.75±2.68	8.87±1.88	8.84±2.20	9.24±2.79
	ν	18.17±5.94	17.40±2.01	21.40±3.78	10.17±4.55	8.67±4.38	9.40±2.07
	τ	17.10±3.21	17.98±2.86	22.06±2.52	9.42±2.51	9.68±2.48	10.98±2.99
COIL20	μ	29.49±2.24	29.23±1.68	24.61±3.16	10.35±2.30	10.51±2.06	10.92±2.05
	ν	31.46±7.79	29.25±2.81	21.25±3.63	9.79±4.94	9.79±4.94	10.75±3.02
	τ	28.98±2.74	29.19±3.61	23.83±3.61	11.30±2.17	11.44±2.39	11.16±1.97
USPST	μ	23.84±3.26	23.91±2.36	18.92±2.81	15.12±2.90	14.36±2.55	13.51±1.89
	ν	24.67±4.54	24.00±2.33	17.53±4.29	14.67±3.94	15.17±4.04	13.47±2.65
	τ	23.60±2.32	23.85±2.71	18.92±3.19	16.44±3.53	14.91±2.83	13.85±2.41

else compute the output weight β:

$$\beta = H^{\mathrm{T}}(I_{N_u+N_l} + CHH^{\mathrm{T}} + \lambda LHH^{\mathrm{T}})^{-1}CT, \tag{17}$$

where the identity matrix I is now with dimension equal to the whole number of samples $(L+U)$.

The performance comparison of different semisupervised methods is shown in Table 2. It is obvious that SS-ELM outperforms the supervised SVMs and ELMs consistently on all data sets, which demonstrates that SS-ELM is able to explore the unlabeled data effectively to achieve significantly better performance than pure supervised learning algorithms. We can also observe that SS-ELM yielded comparable accuracy with the three state-of-the-art semisupervised learning algorithms.

2.4.2 *US-ELM*

In the unsupervised setting, the entire training data are unlabeled, and our target is to find the underlying structure of the original data. Theoretically, the formation of US-ELM is similar to that of SS-ELM, which can be presented as follows:

Given an unlabeled training sample set: $N_u = \{(x_i)|x_i \in R^n,\ i = 1, \ldots, U\}$ and the expected output dimension is N_o.

Step 1 Construct the graph Laplacian L from N_u.

Step 2 Initiate an ELM with N_h hidden nodes with random input weights and biases and calculate the output matrix H.

Step 3 **if** the number of N_h is less than the number of training sample U, find the generalized eigenvector $v_2, v_3, \ldots, v_{N_o+1}$ through the N_o+1 smallest eigenvalues by the following equation:

$$(I_{N_h} + \lambda H^{\mathrm{T}}LH)v = \gamma H^{\mathrm{T}}Hv. \tag{18}$$

Let $\beta = [\tilde{v}_2, \tilde{v}_3, \ldots, \tilde{v}_{N_o+1}]$, where $\tilde{v}_i = v_i/\|Hv_i\|$, $i = 2, \ldots, N_o+1$.

else find the generalized eigenvector $u_2, u_3, \ldots, u_{N_o+1}$ through the N_o+1 smallest eigenvalues by the following equation:

$$(I_{N_u} + \lambda LHH^{\mathrm{T}})u = \gamma HH^{\mathrm{T}}u. \tag{19}$$

Let $\beta = H^{\mathrm{T}}[\tilde{u}_2, \tilde{u}_3, \ldots, \tilde{u}_{N_o+1}]$, where $\tilde{u}_i = u_i/\|HH^{\mathrm{T}}u_i\|$, $i = 2, \ldots, N_o+1$.

Table 3 Performance comparison of the proposed US-ELM

Data set	Accuracy	k-means	DA	SC	LE	USELM
IRIS	Average	82.28±13.93	89.69±14.01	76.16±8.92	80.85±13.82	86.06±15.92
	Best	89.33	97.33	84.00	89.33	97.33
WINE	Average	94.47±3.85	95.24±0.48	93.32±11.36	96.63±0	96.63±0
	Best	96.63	95.51	96.63	96.63	96.63
SEGMENT	Average	60.53±5.86	59.06±4.03	65.64±4.72	62.39±5.03	64.22±5.64
	Best	67.10	62.21	77.10	68.27	74.50
COIL20	Average	57.92±5.44	41.43±1.15	42.21±4.88	64.76±7.82	87.58±3.47
	Best	70.56	44.86	55.21	80.00	90.35
USPST	Average	65.42±3.22	69.04±4.81	69.37±10.39	72.44±6.20	75.78±5.90
	Best	72.45	80.57	84.00	81.32	87.39
YALEB	Average	38.46±3.74	32.24±1.70	42.32±3.37	41.96±3.52	44.08±3.24
	Best	47.27	35.76	47.27	49.09	50.91
ORL	Average	50.86±3.03	35.06±1.36	52.53±2.74	52.94±3.20	59.26±3.54
	Best	58.75	38.00	57.25	61.75	67.50

The average and the best clustering accuracies of these algorithms are reported in Table 3. From the results, we can conclude that US-ELM has obtained satisfying results on all the data sets. Considering the criterion of best clustering accuracy, US-ELM yielded the best results among the five algorithms on six out of the seven data sets.

3 New trend: multilayer learning with high level representations

Feature learning is currently one of the most attractive leading trends in machine learning. Learning meaningful representations of the input data helps to extract useful information when building classifiers or other predictors. In the case of probabilistic models, a good representation can capture the posterior distribution of the observed inputs and is expected to significantly improve the learning performance.

In the existing works, ELM is also extended to a multilayer hierarchical structure, which is based on an unsupervised learning method, i.e., ELM autoencoder (ELM-AE). In this section, we introduce the autoencoder in detail and give an overview of the ELM multilayer framework.

3.1 ELM autoencoder (ELM-AE)

Mathematically, an autoencoder maps the input data $x \in [0,1]^d$ into a higher level representation, and then uses latent representation $y \in [0,1]^{d'}$ through a deterministic mapping $y = h_\theta(x) = g(Wx + b)$, parameterized by $\theta = \{W, b\}$, where $g(\cdot)$ is the activation function, W is a $d' \times d$ weight matrix, and b is a bias vector. The resulting latent representation y is then mapped back to a reconstructed vector $z \in [0,1]^d$ in the input space $z = h_{\theta'}(y) = g(W'y + b)$ with $\theta' = \{W', b'\}$.

The autoencoder performs as feature extractor in a multilayer learning framework. It uses the encoded outputs to approximate the original input by minimizing the reconstruction errors. Unlike the original autoencoder, the ELM-based autoencoder [26] randomly maps the input data though nonlinear transformation and then searchs the path back to the original data with optimization method.

3.1.1 *Solution of nonorthogonal problem: ridge regression*

The nonorthogonal model introduced in Section 2 could be easily adopted in ELM-AE proposed in [26]. Using randomly mapped outputs as latent representation, we could directly use the ridge regression as

optimization method to obtain the new basis through random mapping.

The output weights β of ELM-AE are responsible for learning the transformation from the feature space to the input data. When the hidden nodes are less than the input dimension, the ELM-AE tends to achieve compressed representations, and the output weights β are calculated by the following equation [26]:

$$\beta = \left(\frac{1}{\lambda} + H^{\mathrm{T}}H\right)^{-1} H^{\mathrm{T}}X, \tag{20}$$

where $H = [h(1), \ldots, h(N)]^{\mathrm{T}}$ is the hidden layer output vector and $X = [x(1), \ldots, x(N)]^{\mathrm{T}}$ is the original input data.

For the equal-dimension ELM-AE representations, the output β turns to be the solution of orthogonal projection:

$$\beta = H^{-1}X, \tag{21}$$

$$\beta^{\mathrm{T}}\beta = I. \tag{22}$$

However, due to the use of ℓ_2 penalty in the original ELM, the extracted features by the ELM-AE in [26] tend to be dense and may have redundancy. In this case, a more sparse solution is preferred.

3.1.2 *Sparse representations: fast iterative shrinkage-thresholding algorithm (FISTA)*

To generate more sparse and compact features of the inputs, an ℓ_1 optimization is performed for the establishment of ELM-AE in [31], termed as recovery-based ELM (R-ELM). R-ELM uses the random mapped data as the corrupted version of the original ones and adopts ℓ_1 optimization to recover it. Then, the recovered data are finally utilized for the layer-wise unsupervised training.

The optimization model of R-ELM sparse autoencoder can be denoted as the following equation.

$$O_\beta = \underset{\beta}{\operatorname{argmin}} \left\{ \|H\beta - X\|^2 + \|\beta\|_{\ell_1} \right\}, \tag{23}$$

where X represents the input data, H denotes the random mapping output, and β are the hidden layer weights to be obtained. Note that in the existing deep learning (DL) algorithms, X are usually the encoding outputs of the bases β, which need to be adjusted during the iterations of optimization. However, in the R-ELM, as we are to use random mapping for hidden layer feature representation, X are the original data and H is the random initialized output which need not be optimized [13]. Furthermore, the experiments in the next section will show that it would not only help to improve the training time but also the learning accuracy.

Hereinafter, we will describe the optimization algorithm for the ℓ_1 optimization problem. For clear representation, we rewrite the object function in (23) as

$$O_\beta = f(\beta) + g(\beta), \tag{24}$$

where $f(\beta) = \|H\beta - X\|^2$, and $g(\beta) = \|\beta\|_{\ell_1}$ is the ℓ_1 penalty term of the training model.

A fast iterative shrinkage-thresholding algorithm (FISTA) [32] is adopted to solve the problem in (24). FISTA minimizes a smooth convex function with complexity of $O(1/k^2)$, where k denotes the iteration times. The implementation details of FISTA are as follows [31]:

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

(2) Begin the iteration by taking $y_1 = \beta_0 \in R^n$, $t_1 = 1$ as the initial points. Then, for k ($k \geqslant 1$):

(a) $\beta_k = p_L(y_k)$, where p_L is given by

$$p_L = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{L}{2}\|\beta - (\beta_{k-1} - \frac{1}{L}\nabla f(\beta_{k-1})\|^2 + g(\beta) \right\}. \tag{25}$$

(b) $t_{k+1} = \frac{1+\sqrt{1+4{t_k}^2}}{2}$.

(c) $y_{k+1} = \beta_k + (\frac{t_k - 1}{t_{k+1}})(\beta_k - \beta_{k-1})$.

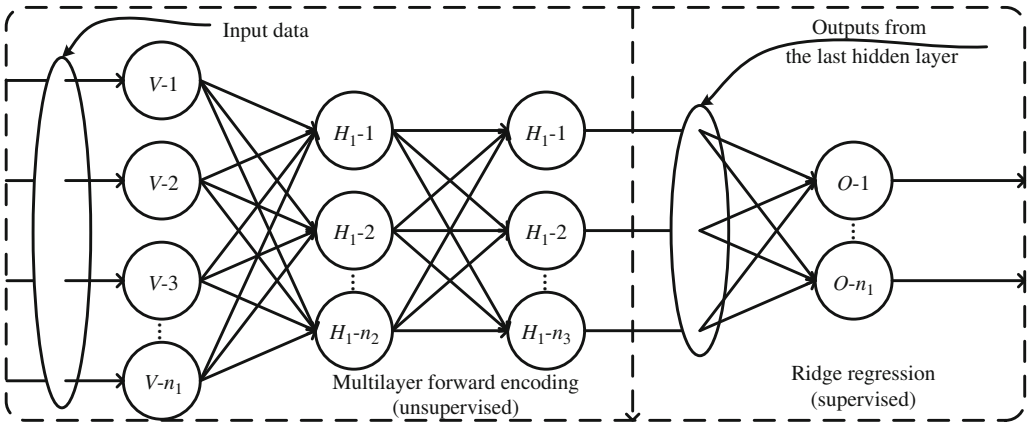

Figure 1 Architecture of ML-ELM framework.

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

And as the autoencoder is adopted as the building block of the R-ELM, higher level feature representations can be generated by layer-wise comparison. Also, compared with the ELM-AE which uses the inputs as outputs with traditional ridge regression method, the ℓ_1 optimization has been proved [32,33] to be a better solution for data recovery and other applications. It will help to reduce the number of neural nodes and thus further improve the testing time.

3.2 Multilayer ELM (ML-ELM)

Based on the principles of ELM-AE, the original ELM can be extended to a multilayer framework [26], as shown in Figure 1. Unlike the commonly used greedy layer-wise training in traditional DL methods [34–39], the ML-ELM training architecture is structurally divided into two separate phases: unsupervised hierarchical feature representation and supervised regression. It could also be seen that after unsupervised hierarchical training in the ML-ELM, the resultant outputs of the Nth layer are viewed as the high-level features extracted from the input data.

Before unsupervised feature learning, the input raw data will be transformed into random feature space, which can help to exploit hidden information among training samples. Then, a N-layer unsupervised learning is performed to obtain the high-level sparse features. Mathematically, the output of each hidden layer can be represented as

$$H_i = g(H_{i-1} \times \beta), \tag{26}$$

where H_i is the output of the ith layer ($i \in [1, N]$), H_{i-1} is the output of $(i-1)$th layer, $g(\cdot)$ denotes the activation function of the hidden layers, and β represents the weighting parameter. Note that here each hidden layer of ML-ELM is an independent module and functions as a separated feature extractor. As the layers increasing, the resulting feature becomes more compact. Once the feature of the previous hidden layer is extracted, the weights or parameters of the current hidden layer will be fixed and need not to be fine-tuned. This is totally different from the existing DL frameworks [34,35], where all the hidden layers are put together as a whole system, with unsupervised initialization. The whole system needs to be retrained iteratively using BP-based NNs. Thus, the training of ML-ELM would be much faster than that of the DL.

The comparisons of different multilayer learning algorithms are presented in Table 4, and one can see that ELM-based ones achieve better learning accuracies with much faster learning speed. Furthermore, it is worth noting that the ML-ELM frameworks (R-ELM and ELM-AE) outperform the original single layer ELM [11] in both learning accuracy and training time. And the reason may be that hierarchical architecture captures more sparse and meaningful feature representations and less nodes are needed.

Table 4 Performance comparison of different methods

Method	Accuracy (%)	Training time (s)
R-ELM [31]	99.12	281.37
ELM-AE [26]	99.01	444.655
DBN [34]	98.89	20580
DBM [39]	99.05	68246
ELM [11]	97.39	790.96

Table 5 Detection performance of different methods

Method	MF-SVM [40]	SA-SVM [41]	SDA-SVM	SDA-ELM
Accuracy (%)	94.57	91.63	96.45	**97.58**
Missing ratio (%)	5.43	8.37	3.55	2.42
False ratio (%)	3.32	5.64	2.03	1.44
Error ratio (%)	8.75	14.01	5.58	3.86

4 Applications

In practical applications, the training and implementation time of machine learning algorithms is usually under certain limitations. In this case, traditional time-consuming learning frameworks (e.g., NNs, SVM, and DL) face severe bottleneck, especially for real-time and high-speed tasks. To address this issue, ELM could provid an accurate prediction of various input data with relatively high computational efficiency. In this section, we will introduce different applications using ELM, and it should be noted that they share some common characteristics: accurate and stable solution with extremely fast learning speed.

4.1 Ship detection

Ship detection with remote sensing images is of vital importance for maritime security and other applications, for example, traffic surveillance, protection against illegal fisheries, and sea pollution monitoring. Vessel monitoring from satellite images provides a wide visual field and covers large sea area, and thus achieves a continuous monitoring of vessels' locations and movements. However, ship detection in optical spaceborne images usually suffer from two main issues [40]: (1) weather conditions like clouds, mists, ocean waves result in more pseudo-targets for ship detection. (2) optical spaceborne images with higher resolution naturally lead to larger data quantity than other remote sensing images, and thus optical spaceborne images are more difficult to be tackled for real-time applications.

To tackle these issues, a compressed domain ship detection framework using ELM for optical spaceborne images is proposed in [17]. Compared with the previous works, this approach achieves faster and better classification with its ELM-based multilayer feature representation model in compressed domain. The overall architecture could be seen in Figure 2. Once ship candidates are extracted with traditional threshold-based coarse segmentation, the feature extraction will be conducted. The singularities of the low frequency component of wavelet decomposition are detected to train the first autoencoder SDA1 [37]. Then, the combination of high frequency components is used to train the second autoencoder SDA2. The two autoencoders are viewed as feature extractors to obtain high-level features both in time and space domains, and the resultant features are fused by ELM to make the final decision. As introduced in above sections, the ELM learns extremely fast and could achieve better generalization than other traditional learning algorithms. The results in Table 5 demonstrate that ELM-based scheme outperforms the state-of-the-art methods in terms of both the detection accuracy.

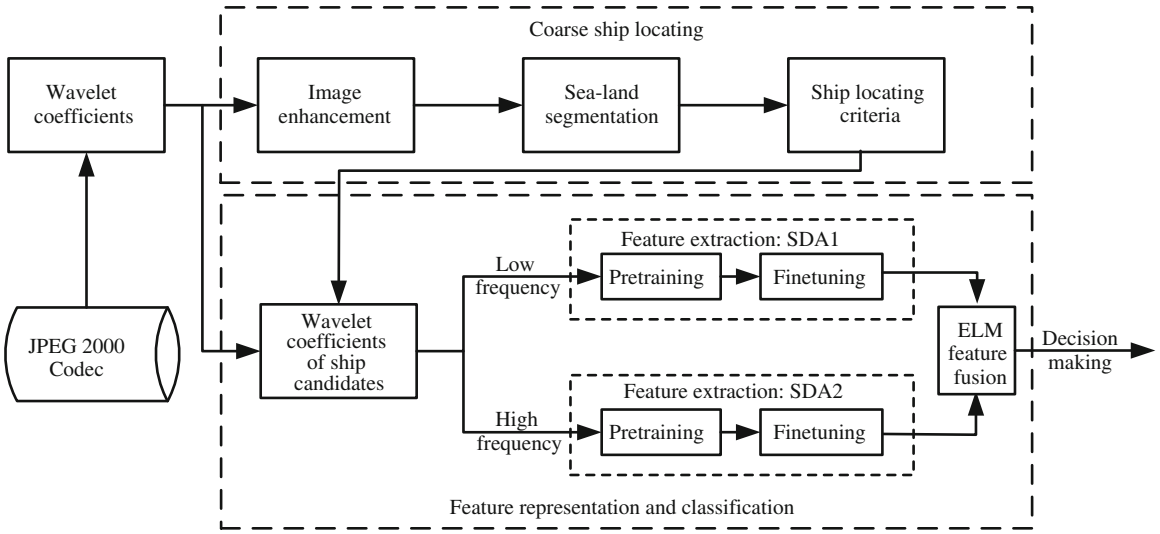

Figure 2 ELM-based ship detection framework.

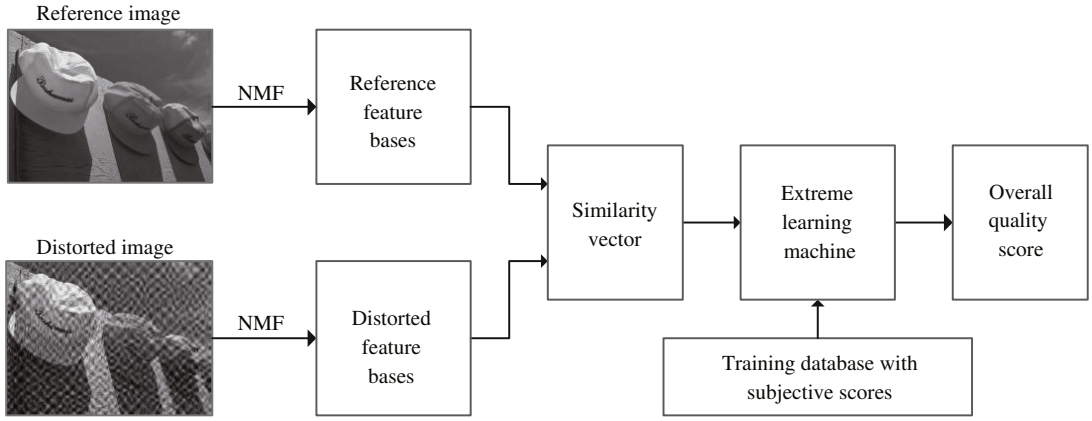

Figure 3 ELM-based IQA framework.

4.2 Image quality assessment

Numerous state-of-the-art perceptual image quality assessment (IQA) [42] algorithms share a common two-stage process: distortion description/feature extraction followed by distortion effects pooling/feature mapping. As for feature extraction, the widely used descriptors include pixel-based ones (e.g., luminance, contrast, gradient) and transformed ones (e.g., singular value decomposition (SVD), wavelet transform). However, most of them seem to have some limitations and not constantly consistent with human perception.

Nonnegative matrix factorization (NMF) [43,44] is capable of generating parts-based representations of objects, which creates possibilities to reflect high-level parts-based visual processing in human brain [45, 46]. The well-known NNs and SVM have been applied in a number of IQA models [47–51]. However, they both suffer from high computational complexity, overfitting, which influences the evaluation performance. In [52], with NMF for extracting image features and ELM for feature mapping, a perceptual full-reference IQA metric is proposed in Figure 3. The input reference and distorted images are first decomposed by NMF, which finally results in similarity vectors. ELM then pools the similarity vectors into quality scores. Table 6 shows the performance comparison of proposed metric and state-of-the-art existing full-reference methods, including CD-MMF [51], SVDR [50], VGS [53], IGM [54], FSIM [55], ADM [56], GSIM [57], and IW-SSIM [58]. The best performances of each criterion are highlighted in boldface.

Table 6 Performance comparison of IQA metrics on TID database

DB	Criteria	NMF-ELM	CD-MMF	SVDR	VGS	IGM	FSIM	ADM	GSIM	IW-SSIM
	SRCC	**0.9465**	**0.9422**	0.7771	0.9022	0.8902	0.8805	0.8617	0.8554	0.8559
TID	KRCC	**0.8913**	**0.8864**	0.5841	0.8467	0.7104	0.6946	0.6842	0.6651	0.6636
(1700)	PLCC	**0.9511**	**0.9476**	0.7889	0.9095	0.8858	0.8738	0.8690	0.8462	0.8579
	RMSE	**0.4188**	**0.4289**	0.8246	0.5577	0.6228	0.6525	0.6620	0.7151	0.6895

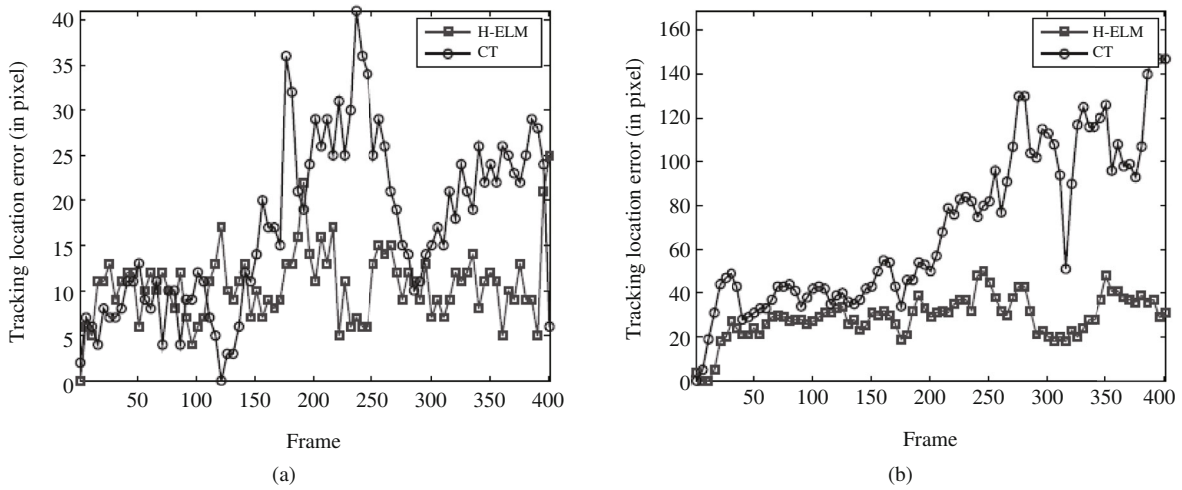

Figure 4 Comparison of tracking location errors on different data sets: (a) David Indoor and (b) Trellis.

4.3 Online visual tracking

Online incremental tracking refers to the tracking method which aims to learn and adapt a representation to reflect the appearance changes of the target. The appearance changes of the specified target include pose variation and shape deformation. Apart from these, extrinsic illumination change, camera motion, viewpoint, and occlusions inevitably cause large appearance variation. Modeling such appearance variation is always one of the nature problems of tracking [59].

Based on the ML-ELM, a novel incremental tracking method is built in [60]. Unlike the conventional online tracking method, more robust features are extracted using ML-ELM, and the classification performance is far better compared with the probability models used in the conventional methods. Also, as demonstrated in the previous section, the training time of ML-ELM is much less than other DL architectures, and this ensures that ML-ELM has obvious advantages in the case of real-time tracking.

In the ML-ELM based tracking framework, as the training is incremental and needs to be online updated, OS-ELM is applied for feature classification and updating. Note that the same sampling scheme is adopted as that in compressive tracking (CT) [61], which is a popular real-time online tracking approach. For ML-ELM tracking, the same initial position is labeled in the first frame, and the tracking results are obtained from the maximum values of the classification responses. In the simulations, we compared the feature extraction, feature updating, and classification performances of the ML-ELM and CT under the same conditions.

The well-used David and Trellis video sequences are tested. The tracking location errors of ML-ELM and CT are presented in Figure 4, and one can see that the error of ML-ELM (denoted as H-ELM in Figure 4) is lower than that of the CT, and the tracking results of ML-ELM are with less fluctuations among different frames. Besides, ML-ELM achieves real-time tracking (over 15 frames with used sequences), by the advantages of fast training of ML-ELM framework.

5 Conclusion

In this paper, an overview and comparison of a new learning technique, i.e., ELM, has been provided. Specifically, we have highlighted the new trends of multilayer learning with ELM; those related to newly emerged ELM-based methods have also been discussed in a more detailed way. Also, several state-of-the-art ELM-based applications have been introduced.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 61301090).

References

1 Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors. Nature, 1986, 323: 533–536
2 Hagan M T, Menhaj M B. Training feedforward networks with the marquardt algorithm. IEEE Trans Neural Netw, 1994, 5: 989–993
3 Wilamowski B M, Yu H. Neural network learning without backpropagation. IEEE Trans Neural Netw, 2010, 21: 1793–1803
4 Chen S, Cowan C, Grant P. Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans Neural Netw, 1991, 2: 302–309
5 Li K, Peng J X, Irwin G W. A fast nonlinear model identification method. IEEE Trans Automat Contr, 2005, 50: 1211–1216
6 Hornik K. Approximation capabilities of multilayer feedforward networks. Neural netw, 1991, 4: 251–257
7 Hassoun M H. Fundamentals of Artificial Neural Networks. MIT Press, 1995. 35–55
8 Cybenko G. Approximation by superpositions of a sigmoidal function. Math Control Signal Syst, 1989, 2: 303–314
9 White H. Artificial Neural Networks: Approximation and Learning Theory. Blackwell Publishers, Inc., 1992. 30–100
10 Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of IEEE International Joint Conference on Neural Networks, Budapest, 2004. 985–990
11 Huang G B, Zhou H, Ding X. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B, 2012, 42: 513–529
12 Huang G B. An insight into extreme learning machines: random neurons, random features and kernels. Cognitive Comput, 2014, 6: 376–390
13 Suykens J A, Vandewalle J. Least squares support vector machine classifiers. Neural Process Lett, 1999, 9: 293–300
14 Huang G B, Chen L, Siew C K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw, 2006, 17: 879–892
15 Huang G B, Chen L. Convex incremental extreme learning machine. Neurocomputing, 2007, 70: 3056–3062
16 Huang G B, Chen L. Enhanced random search based incremental extreme learning machine. Neurocomputing, 2008, 71: 3460–3468
17 Tang J, Deng C, Huang G B, et al. Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine. IEEE Trans Geosci Remot Sen, 2014, in press
18 An L, Bhanu B. Image super-resolution by extreme learning machine. In: Proceedings of IEEE International Conference on Image Processing (ICIP), Orlando, 2012. 2209–2212
19 Suresh S, Venkatesh Babu R, Kim H J. No-reference image quality assessment using modified extreme learning machine classifier. Appl Soft Comput, 2009, 9: 541–552
20 Decherchi S, Gastaldo P, Zunino R. Circular-ELM for the reduced-reference assessment of perceived image quality. Neurocomputing, 2013, 102: 78–89
21 Minhas R, Baradarani A, Seifzadeh S. Human action recognition using extreme learning machine based on visual vocabularies. Neurocomputing, 2010, 73: 1906–1917
22 Huang G B, Chen L. Convex incremental extreme learning machine. Neurocomputing, 2007, 70: 3056–3062
23 Liang N Y, Huang G B, Saratchandran P. A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw, 2006, 17: 1411–1423
24 Rong H J, Huang G B, Sundararajan N. Online sequential fuzzy extreme learning machine for function approximation and classification problems. IEEE Trans Syst Man Cybern B, 2009, 39: 1067–1072
25 Huang G, Song S, Gupta J N. Semi-supervised and unsupervised extreme learning machines. IEEE Trans Syst Man Cybern B, 2014, in press

26 Kasun L L, Zhou H, Huang G B, et al. Representational learning with extreme learning machine for big data. IEEE Intell Syst, 2013, 28: 31–34

27 Tanabe K. Projection method for solving a singular system of linear equations and its applications. Num Math, 1971, 17: 203–214

28 Hoerl A E, Kennard R W. Ridge regression: biased estimation for nonorthogonal problems. Technometrics, 1970, 12: 55–67

29 van Gestel T, de Brabanter J, de Moor B. Least Squares Support Vector Machines. Singapore: World Scientific, 2002

30 Anderson W N, Morley T D. Eigenvalues of the Laplacian of a graph. Linear Multilinear Algebra, 1985, 18: 141–145

31 Tang J, Deng C, Huang G B. A fast learning algorithm for multi-layer extreme learning machine. In: Proceedings of IEEE International Conference on Image Processing, 2014, (in press)

32 Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J Imag Sci, 2009, 2: 183–202

33 Beck A, Teboulle M. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. IEEE Trans Image Process, 2009, 18: 2419–2434

34 Hinton G, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets. Neural comput, 2006, 18: 1527–1554

35 Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks. Science, 2006, 313: 504–507

36 Bengio Y. Learning deep architectures for AI. Found Trends Mach Learn, 2009, 2: 1–127

37 Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE Trans Patt Anal Mach Intell, 2013, 35: 1798–1828

38 Vincent P, Larochelle H, Bengio Y. Extracting and composing robust features with denoising autoencoders. In: Proceedings of ACM International Conference on Machine Learning, Helsinki, 2008. 1096–1103

39 Salakhutdinov R, Hinton G E. Deep Boltzmann machines. In: Proceedings of IEEE International Conference on Artificial Intelligence and Statistics, Clearwater Beach, 2009. 448–455

40 Zhu C, Zhou H, Wang R, et al. A novel hierarchical method of ship detection from spaceborne optical image based on shape and texture features. IEEE Trans Geosci Remot Sen, 2010, 48: 3446–3456

41 Bi F, Liu F, Gao L. A hierarchical salient-region based algorithm for ship detection in remote sensing images. In: Zeng Z G, Wang J, eds. Advances in Neural Network Research and Applications. Berlin/Heidelberg: Springer, 2010. 729–738

42 Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process, 2004, 13: 600–612

43 Lee D D, Seung H S. Algorithms for non-negative matrix factorization. In: Leen T K, Dietterich T G, Tresp V, eds. Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2001. 556–562

44 Lee D D, Seung H S. Learning the parts of objects by non-negative matrix factorization. Nature, 1999, 401: 788–791

45 Buffalo E A, Fries P, Landman R, et al. A backward progression of attentional effects in the ventral stream. Proc Nat Acad Sci, 2010, 107: 361–365

46 Motter B C. Focal attention produces spatially selective processing in visual cortical areas V1, V2, and V4 in the presence of competing stimuli. J Neurophysiol, 1993, 70: 909–909

47 Bouzerdoum A, Havstad A, Beghdadi A. Image quality assessment using a neural network approach. In: Proceedings of IEEE International Symposium on Signal Processing and Information Technology, Rome, 2004. 330–333

48 Carrai P, Heynderickz I, Gastaldo P, et al. Image quality assessment by using neural networks. In: Proceedings of IEEE International Symposium on Circuits and Systems, Scottsdale, 2002. V-253–V-256

49 Narwaria M, Lin W, Cetin A E. Scalable image quality assessment with 2D mel-cepstrum and machine learning approach. Patt Recog, 2012, 45: 299–313

50 Narwaria M, Lin W. SVD-based quality metric for image and video using machine learning. IEEE Trans Syst Man Cybern B, 2012, 42: 347–364

51 Liu T J, Lin W, Kuo C C J. Image quality assessment using multi-method fusion. IEEE Trans Image Process, 2013, 22: 1793–1807

52 Wang S, Deng C, Lin W, et al. A novel NMF-based image quality assessment metric using extreme learning machine. In: Proceedings of IEEE China Summit & International Conference on Signal and Information Processing, Beijing, 2013. 255–258

53 Zhu J, Wang N. Image quality assessment by visual gradient similarity. IEEE Trans Image Process, 2012, 21: 919–933

54 Wu J, Lin W, Shi G, et al. Perceptual quality metric with internal generative mechanism. IEEE Trans Image Process, 2013, 22: 43–54

55 Zhang L, Zhang D, Mou X. FSIM: a feature similarity index for image quality assessment. IEEE Trans Image Process, 2011, 20: 2378–2386

56 Li S, Zhang F, Ma L, et al. Image quality assessment by separately evaluating detail losses and additive impairments.

IEEE Trans Multimedia, 2011, 13: 935–949

57 Liu A, Lin W, Narwaria M. Image quality assessment based on gradient similarity. IEEE Trans Image Process, 2012, 21: 1500–1512

58 Wang Z, Li Q. Information content weighting for perceptual image quality assessment. IEEE Trans Image Process, 2011, 20: 1185–1198

59 Ross D A, Lim J, Lin R S, et al. Incremental learning for robust visual tracking. Int J Comput Vis, 2008, 77: 125–141

60 Tang J, Deng C, Huang G B. Extreme learning machine for multilayer perceptron. IEEE Trans Neural Netw Learn Syst, 2014, in press

61 Zhang K, Zhang L, Yang M H. Real-time compressive tracking. In: Proceedings of European Conference on Computer Vision, Florence, 2012. 864–877