Neurocomputing 350 (2019) 70–90

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Towards a more efficient and cost-sensitive extreme learning machine: A state-of-the-art review of recent trend

Peter Adeniyi Alaba[a,*], Segun Isaiah Popoola[b], Lanre Olatomiwa[c,d], Mathew Boladele Akanle[b], Olayinka S. Ohunakin[e,h], Emmanuel Adetiba[b,f], Opeoluwa David Alex[g], Aderemi A.A. Atayero[b], Wan Mohd Ashri Wan Daud[a]

[a] Department of Chemical Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia
[b] Department of Electrical and Information Engineering, Covenant University, Ota, Ogun-State, Nigeria
[c] Department of Electrical and Electronics Engineering, Federal University of Technology, PMB 65, Minna, Nigeria
[d] Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough, United Kingdom
[e] The Energy and Environment Research Group (TEERG), Department of Mechanical Engineering, Covenant University, Ogun State, Nigeria
[f] HRA, Institute for Systems Science, Durban University of Technology, P.O. Box 1334, Durban, South Africa
[g] Department of Computer Science, University of the People, 225 S Lake Ave Suite 300, Pasadena, CA 91101, USA
[h] Senior Research Associate, Faculty of Engineering & the Built Environment, University of Johannesburg, South Africa

ARTICLE INFO

Article history:
Received 26 July 2018
Revised 21 February 2019
Accepted 12 March 2019
Available online 17 April 2019

Communicated by Miche Y

Keywords:
Extreme learning machine
Artificial intelligence
Big data analytics
Sample structure preserving
Imbalance data

ABSTRACT

In spite of the prominence of extreme learning machine model, as well as its excellent features such as insignificant intervention for learning and model tuning, the simplicity of implementation, and high learning speed, which makes it a fascinating alternative method for Artificial Intelligence, including Big Data Analytics, it is still limited in certain aspects. These aspects must be treated to achieve an effective and cost-sensitive model. This review discussed the major drawbacks of ELM, which include difficulty in determination of hidden layer structure, prediction instability and Imbalanced data distributions, the poor capability of sample structure preserving (SSP), and difficulty in accommodating lateral inhibition by direct random feature mapping. Other drawbacks include multi-graph complexity, global memory size, one-by-one or chuck-by-chuck (a block of data), global memory size limitation, and challenges with big data. The recent trend proposed by experts for each drawback is discussed in detail towards achieving an effective and cost-sensitive model.

1. Introduction

Extreme learning machine (ELM) is a kind of neural network (NN) characterized by biologically inspired single-hidden-layer feedforward network (SLFN), using biological learning techniques rather than artificial learning techniques. It is a biological learning technique that involves the use of kernels, random neurons (with or without unknown modeling/shape), and optimization constraint. ELM is more effective in terms of speed, generalization performance, simplicity and efficiency than the traditional NN in practical applications. The word “extreme” implies beyond conventional artificial learning methods, towards brain-like learning [1]. ELM helps to fill the gap between biological learning and machine learning mechanism [1,2]. Rather than using known activation function such as sigmoid, ELM uses unknown nonlinear piecewise continuous functions h(x) being the real activation functions of most living brain neurons [1]. Theoretically, ELM somehow combines brain learning features, matrix theory, control theory, neural network theory, and linear system theory, which were previously regarded to be isolated with big gaps.

Due to the capability for a wide range of activation functions h(x), ELM exhibits universal classification capability and universal approximation capability [1]. ELM can be used in solving problems pertaining to regression, classification, representational learning, feature selection, clustering, and several other learning tasks. Successful applications of ELM have been reported in several domains, such as output power forecasting [3], system identification [2], function approximation [4,5], biomedical engineering [2], biological information processing, data classification [6], computer vision, pattern recognition [7,8], robotics and control [2]. ELM generates the input layer (sensory layer) weights and the hidden nodes biases randomly and determines the output layer weights rationally by solving a generalized inverse matrix. The study of Huang et al. [9,10], substantiated that SLFNs with randomly generated hidden node parameters and with radial or additive basis function hidden

[*] Corresponding author.
E-mail address: adeniyipee@live.com (P.A. Alaba).

https://doi.org/10.1016/j.neucom.2019.03.086
0925-2312/© 2019 Elsevier B.V. All rights reserved.

nodes could function as universal approximators. This is achievable by simply computing the output weights, which link the hidden layer (associator layer) to the output nodes, thereby substantiating its wide application in solving regression and classification problems [11]. Huang, Zhu and Siew [12] and Liang, Huang, Saratchandran and Sundararajan [13] also established that iterative methods are not needed at all for adjustment of SLFNs parameters in ELM, unlike NN, whose network parameters require iterative methods due to the extensive usage of gradient-based learning algorithms [14]. In comparison with other learning techniques, such as back-propagation algorithm (BP), various variants of BP and deep learning algorithms, support vector machine (SVM), the key superiority of ELM is that it does not involve iterative tuning of the parameters [15].

Despite the prominence of ELM model, as well as its excellent features such as insignificant intervention for learning and model tuning, the simplicity of implementation, and high learning speed, which makes it a fascinating alternative method for Artificial Intelligence, including Big Data Analytics, it is still limited in certain aspects. Huang et al. [2] suggested further research on high dimensional data analysis. The network structure of ELM is more complex for large data since the hidden biases and input weights are randomly selected, making the traditional ELM to require more hidden nodes, thereby affecting the network generalization ability. The order of complexity of the algorithm for output weights estimation is O(M^2K), where M is the number of hidden units and K is the number of training points [16]. Furthermore, some ELM could be time-consuming and ineffective due to the predetermination of their network size when trial by error is used before training [13,14]. ELM uses of batch training approach, indicating that the network is trained with all the dataset at once, thereby requiring large processing power and memory. Classical ELM is also plagued with prediction instability. It occurs because of random initialization of the hidden layer biases and the input weights, and imbalanced data distributions. ELM has no mechanism that takes care of imbalanced data distributions that may be encountered in many fields [17] since it is assumed that every single class size is comparatively balanced and the costs of misclassification are equal in the entire datasets [18]. The imbalanced class distribution could make the classification mechanisms learn very complex models, thereby over-fitting [19]. Moreover, despite the recent prominence of ELM due to remarkable learning speed, little or no manual intervention, and good generalization performance, the generalization performance could be worse than that of SVM algorithm for small sample cases or small network size. This is ascribed to the use of Monte Carlo (MC) sampling technique for generation of random input weights. ELM models with small network size exhibit random input weight with poor SSP capability, leading to poor generalization performance [20].

ELM cannot accommodate lateral inhibition by direct use of random feature mapping [21]. This is a serious challenge in learning of large-scale data since ELM was initially developed to run on a machine with a single processing unit. Handling a block of data, or incremental training samples (one-by-one or chunk-by-chunk) is another serious drawback in ELM. Incremental training samples (such as dynamic changes of tidal level) are presented chunk-by-chunk or one by one and they involve time-varying dynamics. Therefore, modeling time-varying process becomes challenging in real-time [22].

This review discussed the major drawbacks of ELM, which affect its efficiency and cost, as well as the current techniques used as the panacea. The drawbacks addressed here include difficulty in determination of hidden layer structure, prediction instability and Imbalanced data distributions, the poor capability of sample structure preserving (SSP), and difficulty in accommodating lateral inhibition by direct random feature mapping. Other drawbacks include multi-graph complexity, global memory size, one-by-one or chuck-by-chuck (a block of data), global memory size limitation, and challenges with big data. The recent trend proposed by experts for each drawback are discussed in detail towards achieving an effective and cost-sensitive model.

2. Extreme learning machine

ELM developed by Huang at el [12,23] utilizes Single Layer Feedforward Neural Network (SLFN) Architecture [24]. ELM engages a random selection of input weights to rationally calculate the output weights of SLFN. The generalization performance of ELM is remarkable with a high learning speed. ELM does not require much human intervention and the learning speed is thousands time faster when compared with the conventional techniques. The rational determination of the network parameters is automatic, thereby needing trivial human intervention and making it efficient for realtime and online applications. ELM has several benefits like high generalization performance, fast learning speed, ease of use, appropriate for several nonlinear kernel and activation functions.

2.1. ELM formulation

The mathematical description of SLFN function with L hidden nodes [13,25] incorporates both RBF and additive hidden nodes in a unified manner as

$$f_L(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), \quad x \in R^n, \quad a_i \in R^n \tag{1}$$

where a_i and b_i are the hidden nodes training parameters and β_i the weight that connects the output node to the ith hidden node. $G(a_i, b_i, x)$ represents the ith hidden node output corresponding to the input x. Considering additive hidden node with the activation function $g(x)$: $R \to R$ (e.g., threshold and sigmoid), $G(a_i, b_i, x)$ is expressed as

$$G(a_i, b_i, x) = g(a_i.x + b_i), \quad b_i \in R \tag{2}$$

where a_i is the weight vector connecting the ith hidden node to the input layer and b_i is the ith hidden node bias. $a_i.x$ represents the inner product of vector a_i and x in R^n. For RBF hidden node with activation function $g(x)$: $R \to R$ (e.g., Gaussian), $G(a_i, b_i, x)$ expressed as

$$G(a_i, b_i, x) = g(b_i \|x - a_i\|), \quad b_i \in R^+ \tag{3}$$

where a_i and b_i represent the center and impact factor of ith the RBF node. R^+ represents the set of positive real values. The RBF network is a special case of SLFN with RBF nodes in its hidden layer. For N, arbitrary dissimilar samples $(x_i, t_i) \in R^n \times R^m$. Thre parameters, x_i represents an $n \times 1$ input vector and t_i represents an $m \times 1$ target vector. If an SLFN, which has L hidden nodes can approximate the N samples with zero error. This indicates that there are β_i, a_i and b_i such that

$$f_L(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), \quad j = 1, \ldots, N. \tag{4}$$

Eq. (4) can be written compactly as

$$X\beta = T \tag{5}$$

where

$$X\left(\tilde{a}, \tilde{b}, \tilde{x}\right) = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ & \cdots & \\ (a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L} \tag{6}$$

with $\tilde{a} = a_1, \ldots, a_L; \quad \tilde{b} = b_1, \ldots, b_L; \quad \tilde{x} = x_1, \ldots, x_L$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \text{ and } T = \begin{bmatrix} t_1^T \\ \vdots \\ t_L^T \end{bmatrix}_{N \times m} \tag{7}$$

*T*he hidden layer output matrix of SLFN is denoted by X. The i^{th} column of X is the i^{th} output of the hidden node corresponding to inputs $x_1, \ldots, x_N$.

ELM formulated using SLFN algorithm with L hidden neurons can train L different samples without error. Even if the number of distinct samples $(N) <$ number of hidden neurons (L) [26]. ELM assigns random parameters to the hidden nodes to compute the output weights using Moore–Penrose generalized inverse matrix expressed as pseudoinverse of H with a small tolerable error $\varepsilon > 0$. The hidden node parameters of ELM a_i and b_i (input weights and biases or centers and impact factors) does not require tuning during training but could be allotted with random values. The below theorems hold:

Theorem 1. *Let an SLFN with L additive or RBF hidden nodes and an activation function* $g(x)$ *that is substantially differentiable at any given R interval. Then, for arbitrary L separate input vectors* $\{x_i | x_i \in R^n, i = 1, \ldots, L\}$ *and* $\{(a_i, b_i)\}_{i=1}^{L}$ *arbitrarily formulated with any continuous probability distribution. The hidden layer output matrix is invertible, the hidden layer output matrix H of the SLFN is invertible and* $\|X\beta - T\| = 0$ *[23].*

Theorem 2. *For any small positive value* $\varepsilon > 0$ *and activation function* $g(x)$: $R \to R$ *that is substantially differentiable in any interval, there exists* $L \le N$ *such that for N arbitrary distinct input vectors* $\{x_i | x_i \in R^n, i = 1, \ldots, L\}$ *for any* $\{(a_i, b_i)\}_{i=1}^{L}$ *arbitrarily obtained according to any continuous probability distribution* $\|X_{N\times L}\beta_{L\times m} - T_{N\times m}\| < \varepsilon$ *with probability one Liang et al. [13].*

Since the hidden node parameters of ELM does not require tuning during the learning process and the fact that they are basically assigned random values, Eq. (5) is expressed as a linear system and the output weights can be expressed as

$$\beta = X^{\dagger} T \tag{8}$$

where $X\dagger$ is the Moore–Penrose generalized inverse [26] of the hidden layer output matrix H and can be computed using techniques like orthogonal projection, orthogonalization, iterative, singular value decomposition (SVD) [15]. The orthogonal projection method can be used only when $X^T T$ is nonsingular and $X^{\dagger} = (X^T T)^{-1} X^T$. Orthogonalization and iterative techniques are limited due to the use of searching and iterations. Implementations of ELM employs SVD for computation the pseudoinverse of X, because it is relevant in all situations. ELM is, therefore, a batch learning technique.

3. Difficulty in the determination of network architecture

Due to the random selection of hidden biases and input weights, the traditional ELM certainly requires more hidden nodes, making the network structure more complex for large data, thereby affecting the network generalization ability. The order of complexity of the algorithm for output weights estimation is O(M^2K), where M is the number of hidden units and K is the number of training points [16]. Furthermore, some ELM could be time-consuming and ineffective due to the predetermination of their network size when trial by error is used before training [13,14]. ELM uses of batch training approach, indicating that the network is trained with all the dataset at once, thereby requiring large processing power and memory. Although online training techniques (rather than batch training) have been recommended as the panacea, the training time is mostly determined by the network size to retain an O(M^2) implementation complexity because of the big number of neurons characteristically employed in the single hidden layer [27,28].

3.1. Incremental ELM methods

The difficulty in the determination of network architecture can be dealt with using three different approaches, which are incremental, pruning, and the combination of incremental and pruning. Incremental approach initializes a small SLFN, and then iteratively add hidden nodes group by group (or one by one) to the SLFN until a specified condition is fulfilled. The criterion for termination of I-ELM is that the training error is below a specified minimum or the number of hidden nodes attains a specified peak. I-ELM is the pioneering effort of incremental approaches recommended by Huang et al. [25]. I-ELM adds hidden to the hidden layer nodes one-by-one and freezes the output weights of the prevailing hidden nodes. Huang et al. proposed two I-ELM variants called EI-ELM (Enhanced I-ELM) [29] and CI-ELM (Convex I-ELM) [30]. In EI-ELM, at every single learning step, some hidden nodes are generated randomly, and the hidden node causes the major reduction in the residual error will be included in the current network [15]. EI-ELM offers a more compact network without an increase in computations when compared with I-ELM. CI-ELM enhanced the rate of I-ELM convergence with the aid of convex optimization technique to recompute the output weights of the prevailing nodes upon random addition of a new hidden node.

3.2. Pruning methods

Pruning techniques commence from a large SLFN and employ a number of heuristics to prune insignificant hidden nodes. The first pruning technique reported for ELM is Pruned-ELM (P-ELM) [31]. Using information theory criterion, P-ELM utilizes the information gain (IG) and Chi-squared (2) to determine the significance between the class labels and the hidden nodes and subsequently eradicates the nodes with low relevance. The foremost shortcoming is that P-ELM exhibits high computational complication because of the discretization for the probability density estimation of continuous variables [15]. Moreover, the discretization will unavoidably bring about loss of information. Miche, Sorjamaa, Bas, Simula, Jutten, and Lendasse [11] proposed a two-stage pruning algorithm called Optimally Pruned ELM (OP-ELM). Multi-response sparse regression algorithm was used to rank the hidden nodes in the first stage, while the leave-one-out (LOO) validation technique was used to select the optimum number of hidden nodes in the second stage. Recently, Alencar et al. [32] used the optimal subset of hidden nodes based on a genetic algorithm to formulate a pruning technique for ELM. This technique suggests a multi-objective fitness function that defines a compromise between the number of pruned neurons and accuracy.

3.3. Combination of pruning and incremental methods

The hidden nodes need a dynamic adjustment when using a combination of incremental and pruning techniques. Dynamically adjusted ELM (DA-ELM) [33] and dynamic ELM (DELM) [34] algorithms demonstrated this. The dynamic adjustment of hidden node parameters was achieved by a recursive expectation-minimization method in DA-ELM, while the hidden nodes are dynamically deleted based on their importance to the performance in the network in D-ELM. The benefit of D-ELM is that it simultaneously adjusts the network parameters and network architectures dynamically.

Table 1
Examining accuracy of ginv function, Courrieu's method and SVD method with the matrix 2-norm in error matrices that correspond to the four characteristic properties of the Moore–Penrose inverse (Error Results for rank-2n matrices, $n = 8, 9, 10, 11, 12$) [38].

	Rank	$\|TT^{\dagger} T - T\|_2$	$\|T^{\dagger} TT^{\dagger} - T^{\dagger}\|_2$	$\|TT^{\dagger} - (TT^{\dagger})*\|_2$	$\|T^{\dagger} T - (T^{\dagger} T)*\|_2$
SVD method (Matlab pinv) Courrieu's method (geninv) Proposed method (ginv)	$_{2}8$	1.765×10^{-12}	1.5573×10^{-12}	7.2466×10^{-13}	6.1911×10^{-13}
		5.4709×10^{-8}	1.8213×10^{-9}	4.3151×10^{-10}	1.7632×10^{-8}
		1.7441×10^{-11}	1.2075×10^{-11}	1.4535×10^{-13}	7.2225×10^{-10}
SVD method (Matlab pinv) Courrieu's method (geninv) Proposed method (ginv)	$_{2}9$	3.4774×10^{-12}	4.7250×10^{-12}	1.7556×10^{-12}	1.2273×10^{-12}
		1.0247×10^{-6}	4.1029×10^{-8}	2.8935×10^{-9}	9.7309×10^{-8}
		9.6675×10^{-11}	3.0203×10^{-8}	3.9218×10^{-13}	6.4164×10^{-9}
SVD method (Matlab pinv) Courrieu's method (geninv) Proposed method (ginv)	$_{2}10$	1.5329×10^{-11}	9.9524×10^{-12}	4.6853×10^{-12}	5.3752×10^{-12}
		5.2165×10^{-6}	3.1255×10^{-7}	1.0565×10^{-8}	9.4435×10^{-7}
		3.3869×10^{-10}	2.3629×10^{-7}	7.2036×10^{-13}	2.7511×10^{-8}
SVD method (Matlab pinv) Courrieu's method (geninv) Proposed method (ginv)	$_{2}11$	3.7990×10^{-10}	6.0937×10^{-9}	1.5855×10^{-10}	1.2520×10^{-12}
		6.8162×10^{-2}	4.4275×10^{-3}	3.9699×10^{-5}	1.7735×10^{-3}
		1.7449×10^{-8}	5.3528×10^{-4}	2.5498×10^{-5}	1.9404×10^{-11}
SVD method (Matlab pinv) Courrieu's method (geninv) Proposed method (ginv)	$_{2}12$	–	–	–	–
		-1.5743×10^{-5}	-5.8952×10^{-6}	-1.8897×10^{-7}	-1.9087×10^{-9}
		3.9876×10^{-7}	1.219×10^{-8}	5.4433×10^{-10}	4.1422×10^{-11}

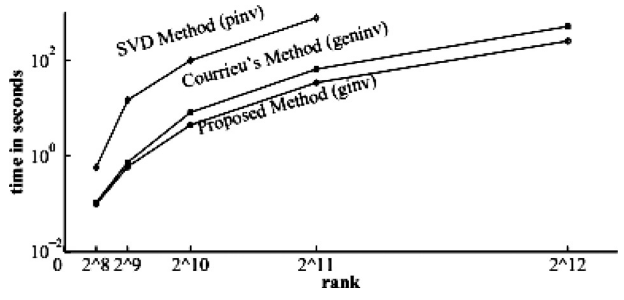

Fig. 1. Time efficiency curves [38].

The common weakness of incremental, pruning and the combination of incremental and pruning is high computation complexity. For pruning techniques, the high computation complexity is because of iterative computations of the heuristics employed in measuring the significance of hidden nodes. In incremental techniques, the high computation complexity is caused by the iterative computation of the Moore–Penrose generalized inverse matrix. Moore–Penrose generalized inverse matrix is estimated using expensive (singular value decomposition) SVD method [35]. Since it is necessary to compute the Moore–Penrose generalized inverse matrix repeatedly for each iteration, whenever a node or a group of nodes is included in the hidden layer [15].

3.4. Generalized inverse method

Several attempts have been made towards increasing the computational speed of Moore–Penrose generalized inverse matrix [36–38]. Katsikis and Pappas [38] constructed a more reliable and fast method called ginv function using Matlab for computation of Moore–Penrose inverse matrix of a rank-n tensor-product matrix. The ginv function first computes the corresponding Gram matrix of the linearly independent vectors $f_1, \ldots, f_n$[3] and subsequently, it resolves the appropriately defined $n \times n$ linear system. Particularly, for each $j = 1, \ldots, n$, the ginv function gives the corresponding $\lambda_i(e_j)$ in the expansion

$$X^{\dagger} e_j = \lambda_i \sum_{i=1}^{n} \lambda_i\left(e_j\right) e_i \tag{9}$$

to obtain the generalized inverse of a certain tensor-product matrix X in the form of

$$X^{\dagger} = \begin{pmatrix} \lambda_1(e_1) & \lambda_1(e_2) & \ldots & \lambda_1(e_k) \\ \lambda_2(e_1) & \lambda_2(e_2) & \ldots & \lambda_2(e_k) \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_n(e_1) & \lambda_n(e_2) & \ldots & \lambda_n(e_k) \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 \end{pmatrix} \tag{10}$$

Assuming X is the corresponding matrix illustration of a rank-n operator, then X is a $k \times k$ matrix and its first n columns did not linearly depend on vectors of R^k, $n < k$ and all the other elements are zeros.

Katsikis and Pappas [38] reported that the generalized inverse based on ginv function needs significantly lesser effort, principally for large matrices, in comparison with the effort required by Courrieu's method and SVD method (Fig. 1). This means that ginv function procedure could achieve a fast-computational solution with a minimal amount of computational resources. The approximations obtained in each one of the tested cases are very accurate and reliable (Table 1). The ginv function also includes a rank test, which helps to streamline the application of the Matlab function for the concerned user, seeing that rank test costs over 50% of the computational time for ginv function response. This makes ginv function a rapid technique for computation of generalized inverses.

3.5. Conjugate Gram–Schmidt process

Other prominent incremental approaches include Gram–Schmidt I-ELM (GSI-ELM) proposed by Zhao, Li, Xi, Liang, Sun and Chen [39] and Error Minimized-ELM (EM-ELM) developed by Feng et al. [40]. GSI-ELM employs the finest-hidden node from a random subset through a specific processing condition at every learning step. The major difference between EM-ELM and I-ELM is that EM-ELM recursively updates the entire output weights upon addition of a new hidden node to the network. Based on testing accuracy and convergence rate, EM-ELM beats I-ELM. EM-ELM was further improved by optimizing the hidden node by addition of PSO to the SLFN one by one as proposed by Han et al. [14].

Toutounian, Soleymani and Ataei [36,37] developed a technique using conjugate Gram–Schmidt process and the Moore–Penrose inverse (CGS-MPi) of partitioned matrices for computation of Moore–Penrose inverse matrices. Conjugate Gram–Schmidt process is

Table 2
Comparison of Gram–Schmidt process and the Moore–Penrose inverse (CGS-MPi) with different iterative methods for dense matrices based on time and error [37].

n	Grevile time/error	SVD time/error	GSO time/error	Geninv time/error	CGS-MPI time/error	SMS time/error	Newton-Raphson time/error	Iterative (of order 8) time/error
32	0.006/1.4e−15	0.000/2.0e−14	0.002/5.8e−15	0.002/3.4e−15	0.002/1.4e−15	0.016/3.2e−13	0.016/3.8e−15	0.031/7.8e−15
64	0.014/1.8e−15	0.016/5.6e−14	0.005/1.1e−15	0.005/4.7e−15	0.006/3.3e−15	0.016/8.5e−13	0.032/8.6e−15	0.062/1.6e−14
128	0.086/3.4e−15	0.081/1.2e−14	0.033/3.2e−15	0.025/7.8e−15	0.031/3.7e−15	0.14/2.07e−12	0.290/7.2e−14	0.405/2.9e−14
256	1.295/4.7e−15	0.581/2.2e−14	0.609/5.3e−15	0.153/1.2e−14	0.210/6.3e−15	0.94/8.6e−12	2.420/3.9e−14	3.132/6.0e−14
512	12.083/6.7e−15	13.034/5.7e−14	5.458/1.1e−15	1.240/2.1e−14	1.648/8.7e−15	7.33/3.25e−11	19.400/7.5e−14	28.164/1.4e−13
1024	105.191/1.2e−14	98.064/2.1e−14	42.711/1.3e−14	11.009/3.0e−14	11.989/1.5e−14	62.97/8.17e−11	158.940/1.5e−13	222.55/2.4e−13

represented by the nested loop below:

$$z_i^{(j)} \leftarrow z_i^{(j-1)} - \frac{\left\langle z_j^{(j-1)},\ z_i^{(j-1)} \right\rangle C}{\left\langle z_j^{(j-1)},\ z_j^{(j-1)} \right\rangle C} z_j^{(j-1)}, \tag{11}$$

In CGS-MPi the matrix X, was calculated by the conjugate Gram–Schmidt method as a unit upper triangular and $Z^{\ T} = L^{-1}$, where L represents the unit lower triangular factor in the root free Cholesky factorization $C = LDL^{\ T}$ and the matrix D is exactly the same here as in $(X^{\ T}X)^{-1} = ZD^{-1}Z^{\ T}$. Moore inverse of X, represented by $X^{\ T}$, is the unique matrix $A \in \mathbb{C}^{n \times m}$, which satisfies the below Penrose equations

(i) $XAX = X$
(ii) $AXA = A$
(iii) $(XA)^* = XA$
(iv) $(AX)^* = AX$.

$\mathbb{C}^{n \times m}$ represents the set of all $m \times n$ matrices in the field of complex numbers, and $*$ signifies conjugate and transpose.

Moreover, Benzi and Tuma, [41] revealed that the conjugate Gram–Schmidt process simultaneiously produces the L factor and Z factor. The entries of L can be determined using

$$l_{ij} = \frac{\left\langle z_j^{(j-1)},\ z_i^{(j-1)} \right\rangle C}{\left\langle z_j^{(j-1)},\ z_j^{(j-1)} \right\rangle C},\ i \geq j. \tag{12}$$

The following is the CGS-MPi algorithm [37]:

Algorithm. CGS-MPi.

```
1. Set $z_i^{(0)} = e_i,\ i = 1:\ n,\ L = I,\ s = 0,\ r = 0$
2. Choose a tolerance $\varepsilon$
3. For $j = 1:\ n$ Do:
4. $d_j = z_j^{(j-1)},\ z_j^{(j-1)}C$
5. If $|d_j| < \varepsilon$ then
6. $s = s + 1,\ \pi_2(s) = j$
7. Else
8. $r = r + 1,\ \pi_1(r) = j$
9. For $i = j + 1:\ n$ Do:
10. $l_{ij} = (z_j(j-1),\ z_i(j-1)iC)/d_j,\ z_i(j) = z_i(j-1) - l_{ij}z_j(j-1)$
11. EndDo
12. EndIf
13. EndDo
14. Perform the permutation set $\pi = [\pi_1\ \pi_2]$ and the permutation matrix $P\pi$
15. Permute the columns (rows) of matrices $Z$, $D$, and $L$ appropriately by using
    permutation matrix $P_\pi (P_\pi{}^T)$, i.e., $Z = P_\pi{}^T\ Z\ P_\pi,\ L = P_\pi{}^T\ L\ P_\pi,\ D = P_\pi{}^T D\ P_\pi$
16. Set $D11 = D(1:\ r,\ 1:\ r)$ and compute $U - 1 = Z(1:\ r,\ 1:\ r)D - 111/2$
17. Compute $L_1 = LD^{1/2}$, and set $V = L_1{}^T\ (1:\ r,\ r+1:\ n)$
18. Compute $B = U^{-1}V$
19. Compute $K = (I_{n-r} + B^T B)^{-1}$
20. Compute $W^\dagger = \begin{bmatrix} l_r - BKB^T \\ KB^T \end{bmatrix} U^{-1}$
21. Compute $A^\dagger = P_\pi\ (W^\dagger)(W^\dagger)^T\ P_\pi{}^T\ A^T$.
```

Toutounian and Ataei [37] reported that their experimental data reveal that for sparse large matrices, the Moore–Penrose inverses computed by this technique is realistically perfect with a fast computation speed, which is more than that of pseudoinverses computed by the other techniques (Tables 2 and 3).

Toutounian and Soleymani [36] also developed a faster iterative method for computation of the roots of an algebraic equation f $(x) = 0$ towards computation of the Moore–Penrose inverse. This technique proved to be a novel fourth-order computational algorithm for computation of an approximate inverse of a square matrix. The approximate inverse is a promising preconditioner for solving linear systems by using the preconditioned Krylov subspace technique (GMRES algorithm) to improve the computational speed [42]. Krylov subspace techniques are usually called parameter-free iterative techniques since they are free from selecting parameters like those needed for successive over-relaxation (SOR) type techniques [43].

The preconditioner is a transformation matrix, which transforms the coefficient matrix into one with a more promising spectrum. The preconditioner is reliable and can effectively lessen the computational time and number of iterations the necessary for convergence (Table 4). The preconditioners are developed according to the equations proposed by Li and Li [44]:

$$V_{i+1} = V_i(3I - 3AV_i + (AV_i)^2,\ i = 0,\ 1,\ 2, \ldots. \tag{13}$$

Where I is the identity matrix. They ascertained that the iterative technique (1) exhibit third-order convergence and satisfies the error inequality $\|e_{i+1}\| \leq \|A\|^2 \|e_i\| 3$, where $e_i = A^{-1}$ - V_i. Additionally, Li and Li [44] presented a collection of formula as follows:

$$V_{i+1} = V_i\left[kI - \frac{k(k-1)}{2}AV_i + \ldots + (AV_i)^{k-1}\right],\ k = 2,\ 3,\ \ldots. \tag{14}$$

They revealed that under the condition $\|I$ - $AV_0\| < 1$, the iterative formula is convergent to A^{-1} with k order of convergence.

To find a solution to $f(x) = 0$, Toutounian and Soleymani [36] assumed that $f(x)$ has a simple root at α, and x_0 is an initial guess satisfactorily close to a. To solve the equation $f(x) = 0$, the following three-step technique was suggested:

$$\begin{cases} y_i = x_i - f'(x_i)^{-1} f(x_i), \\ z_i = x_i - 2^{-1} f(x_i)(f'(x_i)^{-1} + f'(y_i)^{-1}), \\ x_i = z_i - (f'[z_i, x_i]^{-1} f(z_i), \qquad i = 0,\ 1,\ 2,\ \ldots. \end{cases} \tag{15}$$

Where $f,[z_i;\ x_i] = (z_i - x_i) - 1(f(z_i) - f(x_i))$ is the two-point divided difference. The three-step iterative technique is used secant method and the proposed two-step cubically iterative technique by Homeier [45]. To increase the order of convergence of Homeier's technique from three to four, the secant method was used at the third step of

$$V_{i+1} = \frac{1}{2}V_i[9I - AV_i(16I - AV_i(14I - AV_i(6I - AV_i)))],$$
$$i = 0,\ 1,\ 2,\ \ldots. \tag{16}$$

where I is the identity matrix with the same dimension as the matrix A and V_0 is an initial approximation for A^{-1}. It was revealed that the sequence of iterates $\{V_i\}_{i=0}^{i=\infty}$ converges to A^{-1} with fourth-order only if $\|I - AV_0\| < 1$, where $\|.\|$ is any subordinate matrix norm.

Furthermore, to minimize the effect of computational complexity, [46] proposed the construction of a deep ELM network as

Table 3
Comparison of Gram–Schmidt process and the Moore–Penrose inverse (CGS-MPi) with different direct methods for sparse and large matrices [37].

Matrix	Greville Time/Error	SVD method Time/Error	GSO Time/Error	Geninv Time/Error	CGS-MPi Time/Error	SMS Time/Error	Newton-Raphson Time/Error	Iterative (of order 8) Time/Error
WELL1033_*Z*	8.2/2.4e−11	1.8/3.4e−13	3.6/1.3e−11	0.45/1.0e−10	0.43/3.3e−11	7.344/1.1e−08	24.952/1.6e−09	38.201/4.2e−13
WELL1850_*Z*	58.7/2.8e−13	17.8/1.3e−13	24.8/1.3e−12	3.8/2.2e−11	2.6/1.2e−12	48.717/1.8e−08	153.17/1.4e−09	229.850/2.9e−13
ILCC1850_*Z*	58.4/1.6e−08	18.3/2.4e−11	24.8/1.1e−08	3.9/2.0e−08	2.6/2.1e−08	18.859/4.2e−01	194.57/7.1e−10	280.600/6.0e−12
GR−30−30_*Z*	42.5/3.3e−13	21.6/5.3e−14	18.5/1.4e−12	7.6/2.2e−10	3.5/1.7e−12	60.936/7.4e−09	57.108/3.4e−13	51.528/2.8e−13
WATT1_*Z*	374.0/1.6e−04	307.6/6.0e−01	154.3/6.8e+08	63.8/2.0e+15	4.6/1.0e−06	212.060/4.4e−05	164.48/2.2e−05	2454.000/9.9e−06

Table 4
Computation time and the number of iterations of preconditioned GMRES algorithm (IT) required to satisfy $\|\mathrm{rk}\|2 = \|\mathrm{r}_0\|_2 \leq 10^{-8}$. "$\mathrm{V}_1$-(1)-Order-3, V_2-(1)-order-3", "V1-(2)-Order-4 V_2-(2)-order-4", and "V $_1$-(4)-Order-4, V_2-(4)-order-4", denote the preconditioned GMRES algorithm with the preconditioners V_1; V_2 using Eqs. (13), (14), and (16) [36].

Algorithms/system orders	100		400		900		1600	
	IT	Time	IT	Time	IT	Time	IT	Time
V $_1$ -(3)-Order-3	18	0.0138	35	0.0373	51	0.1031	67	0.3997
V $_2$ -(3)-Order-3	10	0.0053	20	0.0289	29	0.0918	38	0.2304
V $_1$ -(4)-Order-4	12	0.0051	22	0.0182	32	0.0536	42	0.1811
V $_2$ -(4)-Order-4	6	0.0124	11	0.0666	16	0.1990	21	1.0840
V $_1$ -(18)-Order-4	11	0.0045	21	0.0181	30	0.0521	40	0.1783
V $_2$ -(18)-Order-4	5	0.0142	10	0.1801	14	0.5930	19	1.4057

a stack of supervised autoencoder ELM modules, and training it module by module. This engenders a considerable improvement of the memory usage and network training time, while concurrently enhancing classification error-rate performance better than what is achievable via single ELM with a similar size of hidden units.

3.6. Cholesky factorisation method

The pseudoinverse of Moore–Penrose could be computed using Tikhonov regularization, QR factorization or LU factorization [42,47,48]. Tikhonov regularization, also called Ridge regression is a limit characterization approach for computation pseudoinverse [49,50]. Considering an arbitrary matrix X

$$X^{\dagger} = \lim_{\delta\to 0}\left(X^T X + \delta I\right)^{-1} X^T = \lim_{\delta\to 0} X^T\left(XX^T + \delta I\right)^{-1}. \tag{17}$$

I represents a size-compatible identity matrix. The limit of the full expression was taken and not only the matrix with the variable $\delta((XX^T + \delta_I)^{-1}$ or $(X^T X + \delta_I))$ [50]. The finite δ is used in an approximation of $X^{\dagger}$ on a vector represented by g. For the approximation to be accurate, δ should be a small multiple of the $\delta_r{}^2$, which is the square of the smallest non-zero singular value of X. For $\delta > 0$, the *a priori* error estimate is given by [50,51].

$$\left\| \left(X^T X + \delta I\right)^{-1} X^T - X^{\dagger} \right\|_2 \leq \delta \left\| X^{\dagger} \right\|_2^3. \tag{18}$$

This regularization can be implemented using Cholesky factorization algorithm to solve the sparse matrix [48]. XX^{T} and $X^{\mathrm{T}}X$ are sparse if X is banded or sparse. For instance, XX^{T} and $X^{\mathrm{T}}X$ are pentadiagonal, if X is tridiagonal. the matrix becomes inverted positive definite when a positive diagonal is added. Cholesky factorization is prominent due to its remarkable computational speed. The main purpose of using Cholesky factorization is to obtain an approximation of the upscaled Hessian matrix (not its exact upscaling) in mesh adaptivity- and nonlinearity-based homogenization. Tikhonov regularization technique can be utilized to enhance the accuracy of the approximation of the pseudoinverse [48]. Upon computation with Cholesky factorization algorithm, enhancement with Tikhonov regularization becomes cost-effective. Presenting the pseudoinverse using by the sum of a series, we have [49,52].

$$X^{\dagger} = \sum_{i=1}^{\infty} X^T\left(XX^T + I\right)^{-i} \tag{19}$$

Note, if K^T is removed from the series as one of the factors, the sum will diverge. Assuming the property $(\alpha X^{\dagger}) = \alpha^{-1}X^{\dagger}$ for $\alpha \neq 0$, and $\delta = \alpha^{-2}$ the following is derived from the above equation:

$$X^{\dagger} = \sum_{i=1}^{\infty} X^T \delta^{i-1}\left(XX^T + \delta I\right)^{-i} \forall \delta > 0. \tag{20}$$

The expression is valid for all $\delta > 0$ since $\alpha \neq 0$ is arbitrary. It rescales the matrix X effectively, making the computation of the inverse of $XX^T + \mathrm{dI}$ not severely-conditioned when X is rank-deficient (and δ is properly chosen). The inverse can be calculated accurately if δ is large, but more factors in the series are required for accurate approximation of $X^{\dagger}$. On the other hand, fewer terms are required to attain a satisfactory accuracy if δ is smaller, but the inverse could not be calculated correctly in finite a precision [48].

QR decomposition/factorization is the orthogonal-triangular decomposition of a matrix X [53], which is expressed as

$$[X]_{M\times N} = [Q]_{M\times M}[R]_{M\times N} \tag{21}$$

Where R is an $M \times N$ upper triangular matrix, and Q is a unitary matrix (orthonormal) of size $M \times M$. The columns of Q, which could be determined using Gram–Schmidt orthogonalization [54] process, become an orthonormal basis for the column space of matrix X. Q and R can be partitioned as

$$Q = [Q_1 \quad Q_2] \tag{22}$$

and

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}. \tag{23}$$

Where $Q_1 = n \times (n-p)$, Q_2 is $n \times p$, $R_{11} = (n-p) \times (n-p)$, and $R_{12} = (n-p) \times p$.

Assuming auxiliary matrices K and S

$$K = \left(S^T S + 1\right)^{-1} S^T R_{11}^{-1} Q_1^T \tag{24}$$

and

$$S = R_{11}^{-1} R_{12}. \tag{25}$$

K is $p \times n$, and S is $(n-p) \times p$. Since R_{11} is a triangular matrix, back-substitutions can be used to compute R_{11}^{-1}. Moreover, for the computation of S and X, the 'inversion' of a $p \times p$ system $(S^T S + I)$ is required [48]. Lastly, with the use of the matrices S and X and the QR factorization, the pseudoinverse can be expressed as [55].

$$X^{\dagger} = \begin{bmatrix} R_{11}^{-1}(Q_1^T - R_{12}K) \\ K \end{bmatrix}. \tag{26}$$

The expression can be utilized in computing $X^{\dagger}f$ for a known vector f. To calculate X and $X^{\dagger}f$, $Q_1^T f$ must be generated. Generally, the original matrix X is sparser than the factors Q and R. Thus, a typical QR decomposition algorithm cannot harness the sparsity of X. Consequently, [56] implemented a sparse QR decomposition algorithm.

To perform sparse QR factorization, the columns of matrix X must be noted prior to the QR factorization to minimize substitution. Therefore, the Q and R factors of $K\ P$ must be generated, where P is a permutation matrix, generally selected automatically [56]. If $XP = QR$, where P is a permutation matrix, which is orthonormal (being orthogonal and a unit vector), then the pseudoinverse can be expressed as $X^{\dagger} = P(QR)^{\dagger}$.

The absolute values of the first row elements of matrix R are bigger than those of the other rows [57], if the columns of X correlate with each other and the first row elements of matrix R comprises the maximum energy of the signal. This makes R matrix so fascinating that several researchers [58] prefer using QR decomposition. Moreover, the computation of QR decomposition complexity when compared with SVD [54].

The LU factorization for an arbitrary matrix X is expressed as $X{=}LU$, where L is the lower triangular matrix and U is the upper triangular matrix. Cholesky decomposition occurs if $L = U^*$, while Doolittle factorization occurs if L has 1′s on its main diagonal, and Crout factorization occurs If U has 1′s on its main diagonal [59]. For all these cases, the following has been established:

$$X^{\dagger} = U^{\dagger}L^{\dagger} = U^*(UU^*)^{-1}(L^*L)^{-1}L^*. \tag{27}$$

Stanimirović & Tasić [59] proposed an algorithm using Cholesky factorization and general representations of symmetric positive matrices. They implemented the algorithm using programming languages DELPHI and MATHEMATICA.

In summary, the difficulty in determining the network architecture can be handled using three incremental technique, pruning technique, and the combination of incremental and pruning technique. The recently used incremental techniques are I-ELM, Enhanced I-ELM, Convex I-ELM, Gram–Schmidt I-ELM, and Error Minimized-ELM. Based on testing accuracy and convergence rate, EM-ELM beats I-ELM. This is because EM-ELM recursively updates the entire output weights upon addition of a new hidden node to the network. Pruned-ELM and optimally pruned ELM are the successfully pruned techniques. Optimally pruned ELM suggests a multi-objective fitness function that determines the relationship between accuracy and number of pruned neurons. Moreover, a combination of incremental and pruning technique was demonstrated by dynamic adjustment of hidden node parameters using the recursive expectation-minimization method in DA-ELM, while the hidden nodes are dynamically deleted according to their significance to network performance in D-ELM.

4. Prediction instability and imbalanced data distributions

Prediction accuracy is certainly a crucial measurement of models in risk analysis [19]. Prediction instability is one of the major limitations of ELM [19,60]. It occurs because of random initialization of the hidden layer biases and the input weights, and imbalanced data distributions. ELM has no mechanism that takes care of imbalanced data distributions that may be encountered in many fields [17] as it is assumed that every single class size is comparatively balanced and the costs of misclassification are equal in the entire datasets [18]. The imbalanced class distribution could make the classification mechanisms learn very complex models, thereby over-fitting [19]. The techniques for solving the imbalanced data problem are classified into algorithmic techniques and re-sampling techniques [61,62]. Re-sampling techniques are preferred for most imbalanced conditions, since only the original training dataset are modify, rather than adjusting the inherent mechanism of algorithmic techniques [19]. Therefore, the re-sampling technique is transportable and external [63], and handles an imbalanced learning condition effectively by using standard classifier [62].

Several studies on imbalance learning only study the influence of the number of the class samples, whereas the dispersion degree of the data is ignored, leading to suboptimal learning results [17,64,65]. Imbalance in data such as image annotation, gene expression, fraud detection, bioinformatics, and oil spill detection data [66,67] has a negative impact on classifier performance, resulting in a biased training, where the minority classes are under-trained and the majority classes are comparatively under-trained. This culminates in an inaccurate prediction of minority classes. This shows that minority classes are of vital importance in prediction accuracy.

Therefore, several researchers have worked towards discovering an accurate, effective and viable prediction mechanism results [17,68]. Liu et al. [68,69] embedded dissimilarity and cost-sensitive factors into classifiers to boost classification stability and minimizes classification costs for classifying high-scale, imbalanced datasets, and redundant. They further improved the work by the introduction of misclassification costs into a classifier (Figs. 2 and 3). This algorithm was named Cost-Sensitive Dissimilar ELM (CS-D-ELM). The CS-D-ELM algorithm was also enhanced by embedding rejection cost to further enhance classification stability. The algorithm was used for classification of Gene Expression Data. Their report shows that embedding rejection into CS-D-ELM algorithm effectually cuts down the average and overall classification cost, while the classification accuracy was not significantly affected.

One of the viable method used for class imbalance learning is Ensemble method, which can efficiently improve classification performance by a combination of a number of weak learners based on a specific rule [18]. Several varieties of ensemble learning techniques such as stacking, boosting and bagging have been developed [70], which combine a number of base learners based on different schemes. Prominent of all the strategies employed for the combination of base learners in ensemble scheme is weighted voting and simple voting. The weighted voting method requires assignment of weights to each base learner and votes by weighting to select the category of a new sample, while simple voting method selects the final category of a new sample based on the predicted classes with the most votes. Apparently, the weights vary amid the different output classes in each base learner. Consequently, the determination of the weights of base learners is essential towards achieving a superior classification performance. Weights selection could be optimized using a genetic algorithm (GA), particle swarm optimization (PSO) [71], as well as differential evolution (DE), which has been effectively employed in several fields like multi-objective optimization, constrained optimization and parallel computing.

Zhai et al., [15] proposed ensemble dropout extreme learning via fuzzy integral (FI-ELM) for data classification. Based on accuracy testing, the developed FI-ELM algorithm beats the traditional ELM (Table 5) with a probability of at least 0.95. The outstanding accuracy could be due to the testing accuracy of every single basic classifier is typically lower than the testing accuracy of ensemble classifiers, or fuzzy integral can satisfactorily represent the interaction between basic classifiers.

Zhang et al. [18] developed an ensemble method based on weighted extreme learning machine (WELM) for class imbalance data problem. This approach optimized the weights of base

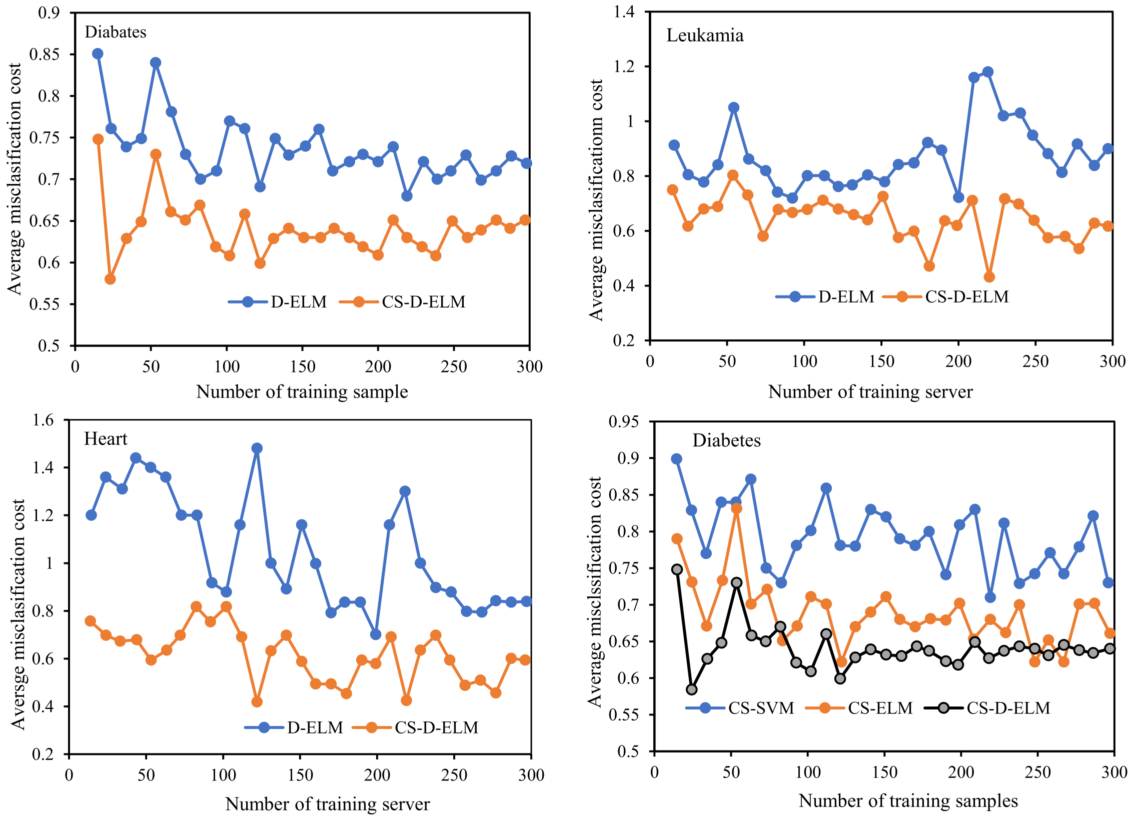

Fig. 2. Average misclassification costs of D-ELM and CS-D-ELM on diabetes, heart, and leukemia datasets [67].

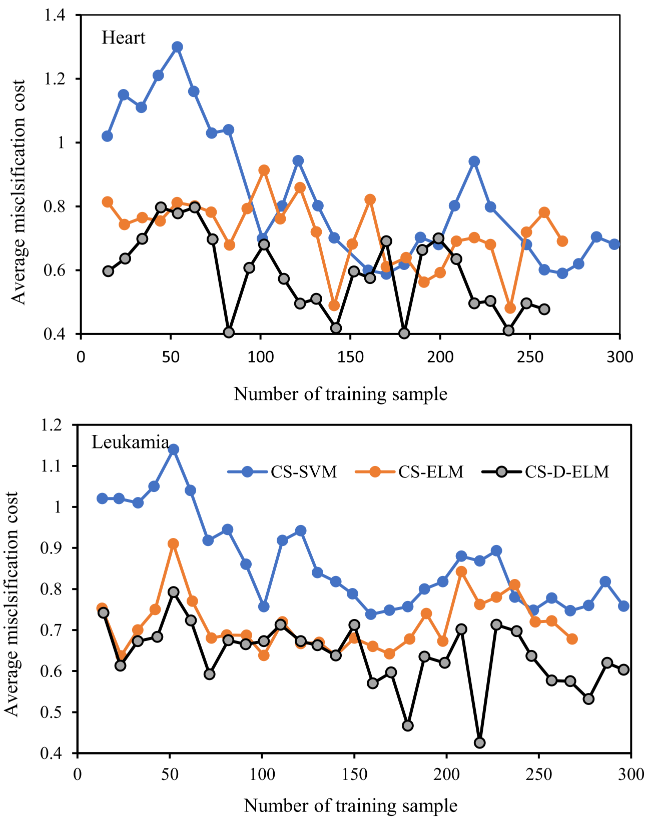

Fig. 3. Average misclassification costs of CS-ELM, CS-SVM, and CS-D-ELM on heart, and leukemia datasets [67].

Table 5
accuracy of FI-EML compared with the classical ELM [15].

Data sets	FI-ELM	ELM
CT	0.9290	0.8683
Fertility	0.8685	0.8267
Forest	0.8764	0.8364
Heart	0.9425	0.8896
Ionosphere	0.9266	0.8806
Iris	0.9658	0.9462
Mammographic	0.8340	0.7808
Optical	0.9619	0.9208
Parkinsons	0.9317	0.8646
Pima	0.8224	0.7676
Cloud	0.8164	0.7962
Shuttle	0.9687	0.9486
Skin	0.9794	0.9546
Linkage	0.9963	0.9801

learners by differential evolution (DE) algorithm, and when applied to several datasets, WELM demonstrated an improved classification performance based on the geometric mean G-*mean* metric (Table 6).

Furthermore, the ensemble method can be optimized by incorporation of reactivated regularization (ER^2-ELM), a discriminatory approach proposed by [60]. The technique generates base classifiers by engaging hybrid seasonings, which consisting of L, γ, p. L represents the number on hidden layer nodes, γ stands for regularization factor, and p is the p^{th} candidate from a set of transfer functions. The hybrid seasonings are used as the control factors towards creating a pool through grid sampling. This was done by conducting probability density estimation to demonstrate how difficult it is to identify an instance, and subsequently adopting a random factor to determine if the ELM base learner is sequentially reactivated. Application of ER^2-ELM considerably decreases

Table 6
G-*mean* comparison results with several algorithms [18].

Dataset	Non-ensemble		Vote-based ensemble		DE-based ensemble	
	(C, L)	G-mean (%)	(C, L)	G-*mean* (%)	(C, L)	G-mean (%)
glass1	(2^{16}, 780)	73.46	(2^{30}, 590)	74.32	(2^{18}, 390)	77.72
haberman	(2^{-6}, 350)	60.37	(2^{12}, 140)	63.1	(2^{28}, 540)	62.68
ecoli1	(2^{-2}, 270)	89.21	(2^{40}, 890)	89.72	(2^{0}, 390)	91.39
new-thyroid2	(2^{16}, 290)	99.47	(2^{10}, 340)	99.47	(2^{32}, 560)	99.24
yeast3	(2^{8}, 810)	92.11	(2^{4}, 270)	94.25	(2^{2}, 490)	94.57
ecoli3	(2^{20}, 40)	88.59	(2^{10}, 10)	88.68	(2^{18}, 40)	89.5
glass2	(2^{12}, 380)	79.02	(2^{8},170)	86.45	(2^{16}, 350)	87.51
yeast1_7	(2^{10}, 180)	76.01	(2^{20}, 370)	78.95	(2^{38}, 20)	78.94
ecoli4	(2^{24}, 190)	96.01	(2^{8}, 750)	96.33	(2^{14}, 310)	96.77
abalone9_18	(2^{24}, 260)	87.89	(2^{4}, 120)	89.24	(2^{16}, 330)	90.13
glass5	(2^{26}, 390)	94.38	(2^{18}, 570)	94.55	(2^{12}, 260)	94.55
yeast5	(2^{16}, 480)	93.27	(2^{12}, 330)	94.51	(2^{28}, 430)	94.59
Average	–	85.82	–	87.46	–	88.13

the complexity in prediction computation when compared with other ensemble methods. This is mainly because other ensemble methods did not discriminatory approach.

Several authors have also used ELM incorporated L_2 regularisation penalty to suppress prediction instability [72–74]. L_2 regularisation penalty, which is in the form of Tikhonov regularization utilizes square of the regression coefficients to solve minimization problem,

$$\min_{\lambda,\hat{w}}\left[\sum_{i=1}^{n}\left(y_i - x_i\hat{w}\right)^2 + \lambda\sum_{j=1}^{p}\hat{w}_j^2\right] \tag{28}$$

culminating in a bias-variance adjustment [73]. Where, x_i, y_i, w_i and n are the inputs, outputs, regression weights, and sample size. λ is the Tikhonov penalty coefficient, which can be determined using the Bayesian information criterion or Nelder–Mead simplex method (fminsearch in MATLAB) [72,75]. For regularized ELM with L_2 penalty, addressing outlier interferences by weighting the sum of squares we have

$$\min_{\lambda,d,\hat{w}}\left[\lambda\sum_{i=1}^{n}\left(d_i\left(y_i - x_i\hat{w}\right)\right)^2 + \sum_{j=1}^{p}\hat{w}_j^2\right], \tag{29}$$

where the di are the weights used in addressing the outliers.

Incorporation of L_2 regularization penalty promotes predictiction accuracy, and minimized and stabilize MSE with an increasing number of neurons [73]. This performance is better than the traditional ELM and variances that employs a Gauss-Markov solution, which uses Ordinary Least Squares (OLS). However, Like OLS, L_2 penalty provides no parsimonious solution because it retains all the variables to enable further interpretability [76].

The OP-ELM proposed for simplification of network structure for large data using an LOO criterion for the choice of the optimum number of neurons in Section 2 could be regularized for enhanced prediction stability. Allen [77] proposed the use of Prediction Sum of Squares [78] to give LOO error a closed matrix form, thereby ensuring MSE fast computation speed. The formula for Allen's PRESS is given by

$$MSE^{PRESS} = \frac{1}{n}\sum_{n=1}^{n}\left(\frac{X_{i\cdot}\left(X^TX\right)^{-1}X^Ty_i - y_i}{X_{i\cdot}\left(X^TX\right)^{-1}X_i^T - 1}\right)^2, \tag{30}$$

meaning that each observation is identified using the other $n-1$ observations and the residuals are finally squared and summed up [72]. Considering Tikhonov regularization Eq. (14) becomes

$$MSE^{PRESS}(\lambda) = \frac{1}{n}\sum_{n=1}^{n}\left(\frac{X_{i\cdot}\left(X^TX+\lambda I\right)^{-1}X^Ty_i - y_i}{X_{i\cdot}\left(X^TX+\lambda I\right)^{-1}X_i^T - 1}\right)^2,$$

being the regularized version. (31)

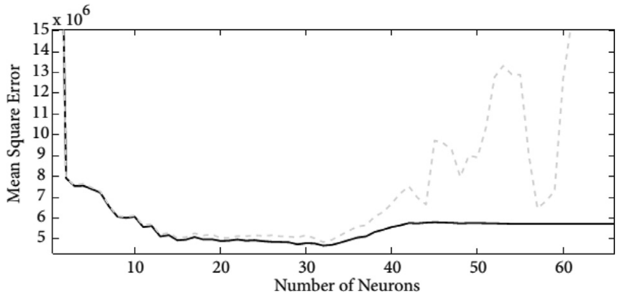

Fig. 4. Comparison of the MSE for the original OP-ELM (grey dashed line) and the proposed TROP-ELM (solid black line) for one data set (Auto Price) for a varying amount of neurons (in the order ranked by the LARS) [73].

For regression problem, OP-ELM network can be regularized using L_2 penalty, that is Tikhonov regularized OP-ELM (TROP-ELM) Lendasse et al. [73] investigated ten datasets and reported that averagely, TROP-ELM performs on ~27% better than the OP-ELM, giving a standard deviation of 52% lower than that of the OP-ELM. TROP-ELM employs both L_1 and L_2 penalties in cascade to reduces over-fitting, the negative influences of random initialization and avoid large computational times problems usually experienced when the two penalties are intertwined [73,79]. Fig. 4 compares the performance of OP-ELM and TROP-ELM on Auto price dataset based on MSE.

L_1 penalty was implemented on the output weights in OP-ELM using Least Angle Regression (LARS) between the hidden and output layer to rank the best neuron. However, the implementation of OP-ELM can be marred (cause numerical instability) if the dataset is not fully ranked. Implementation of TROP-ELM with an ensemble of regularization methods could provide a more robust and scalable model. Mozaffari et al. [72] proposed an ensemble of regularized OP-ELM with negative correlation (OP-ELM-ER-NCL). They reported that the redundant complexity of hidden neurons reduces, and the numerical stability of identifier increases. The inclusion of negative correlation learning approach enables the ensemble to choose the most effective regularization methods and remove the redundant (ineffective) ones, thereby decreasing the complexity of the resulting ensemble.

Xiao et al., [17] used class-specific cost regulation ELM (CCR-ELM) to classify imbalanced data by introduction of class-specific regulation cost for misclassification of each class in the performance index. This is capable of reducing the impacts of a number of class samples, as well as that of the dispersion degree of the data. Their report shows that CCR-ELM is capable of significant improvement of classification, as well as generalization performance in comparison with the classical ELM. CCR-ELM is

Table 7
Classification accuracy of four algorithms on air pollutant data set [65].

	HS-OSELM	OSELM	ELM	MC-OSELM	SM-OSELM
Training time (s)	0.2652	0.2184	0.0611	0.0649	0.3853
Test time (s)	0.0597	0.0591	0.0659	0.0615	0.0619
Minority training accuracy (%)	0.9175	0.1967	0.2377	0.8779	0.853
Majority training accuracy (%)	0.9013	0.9918	0.9903	0.9081	0.8998
Minority test accuracy (%)	0.7353	0.4412	0.3529	0.7157	0.7059
Majority test accuracy (%)	0.8642	0.992	0.9888	0.8562	0.8498
Whole training accuracy (%)	0.9089	0.9253	0.9274	0.8979	0.8804
Whole test accuracy (%)	0.8462	0.9148	0.8997	0.8365	0.8297
G-mean	0.7971	0.6597	0.5905	0.7818	0.7736

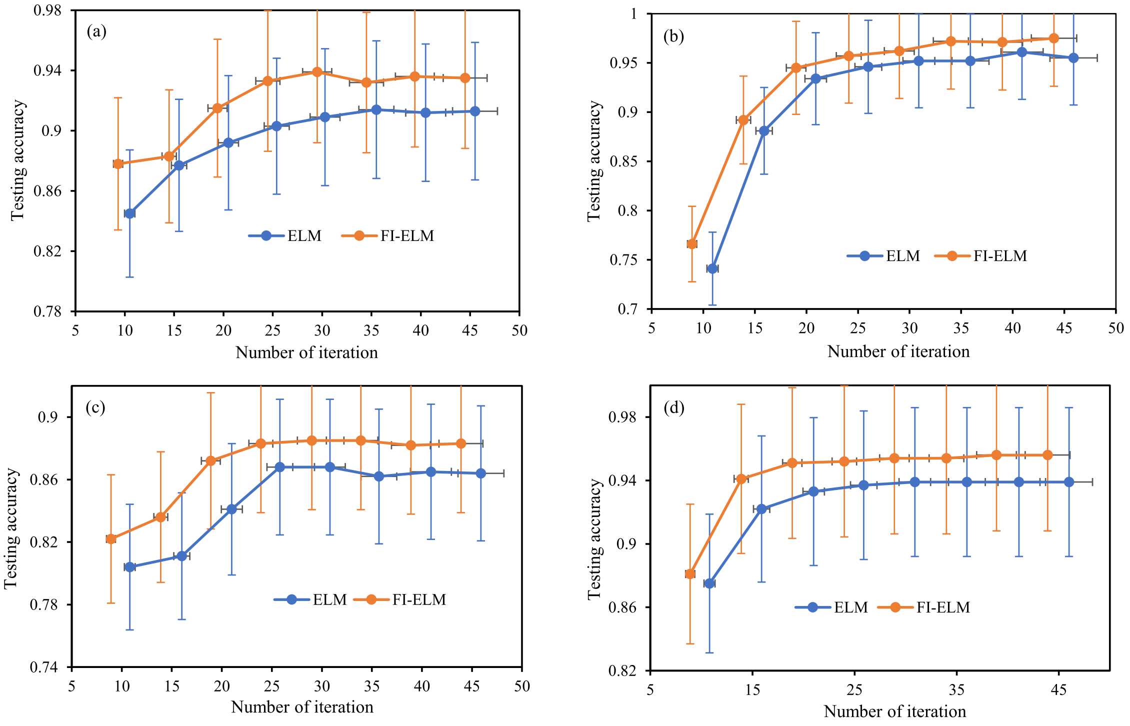

Fig. 5. Prediction stability of FI-ELM compared with the traditional ELM on (a) data set Ionosphere, (b) data set Iris, (c) data set Forest, (d) data set Optical [15].

directly efficient for both binary classification and multiclass classification. Akbulut et al. [64] proposed a novel neutrosophic set theory weighted based ELM system, referred to as neutrosophic weighted extreme learning machine (NWELM). This scheme uses an neutrosophic c-means (NCM) clustering algorithm to approximate ELM output weights. The report reveals that the developed NWELM scheme is efficient in handling the problem of class imbalance. In comparison with two ensemble-based weighted ELM, weighted ELM and unweighted ELM methods, NEWLM proved to be more effective for both real binary and artificial imbalance data sets.

Mao et al., [80] proposed OS-ELM with two-stage hybrid strategy, using Sherman-Morrison matrix inversion in a novel online fast leave-one-out cross-validation (LOO–CV) to solve online imbalance data problem and an add-delete mechanism to update network weights. To model the distribution of the minority class data at the offline stage, principal curve, and database technique was used. This technique was tested using real-world Macau air pollutant forecasting dataset and four UCI datasets. They reported that the proposed OS-ELM with two-stage hybrid approach performs better than the traditional ELM, meta-cognitive OS-ELM, and OS-ELM in terms of numerical stability and generalization performance. In a more recent study, Mao et al. [65] presented a novel hybrid sampling online ELM for sequential class-imbalanced data learning and deduction process. This is aimed to balance the majority and minority classes with the same sequential distribution characteristic of the source data. The technique comprises offline and online stages. The principal curve was introduced to build confidence regions of the majority and minority classes at the offline stage. From the forgoing, under-sampling of majority class and over-sampling of minority class are performed to form new synthetic samples to establish the initial ELM model. The most valuable synthetic samples of majority class were selected based on sample significance at the online stage. Subsequently, a new LOO–CV algorithm using Cholesky decomposition was suggested to decide whether to update the weight of the ELM network during the online stage or not. It was established that the proposed online ELM exhibits upper bound of information loss. They tested the online ELM scheme using one real-world air pollutant forecasting dataset and seven UCI datasets. The online ELM is capable of simultaneous improvement of both majority and minority classes' classification performance based on ROC curve, G-mean value, and accuracy in comparison with ELM, OS-ELM, OSELM with SMOTE scheme and meta-cognitive OS-ELM (Table 7).

Zhai et al. [15] proposed ensemble dropout extreme learning via fuzzy integral (FI-ELM) for data classification. The developed FI-ELM algorithm beats the traditional ELM based on prediction stability (Fig. 5). Therefore, the developed FI-ELM algorithm is capable if improving prediction stability. The outstanding performance of FI-ELM in classification problem is ascribed to the following points:

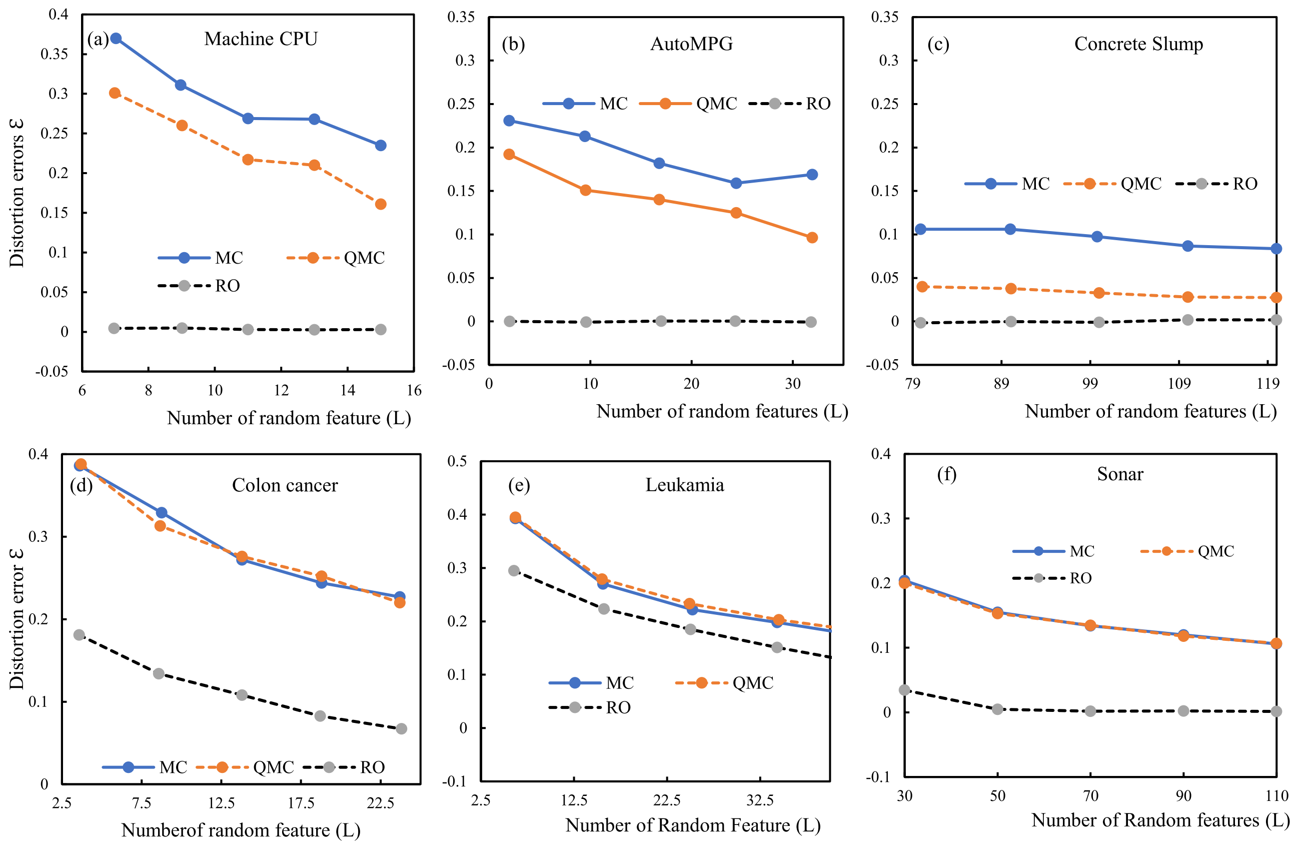

Fig. 6. The mean distortion errors of MC, QMC and RO on 3 regressions (a, b & c) and classification (d, e & f) data sets over 50 runs [20].

(1) Fuzzy integral is capable of modeling the relations between the basic classifiers with complementary information since it is a tool for classifier fusion.
(2) Since the basic classifiers developed using dropout method exhibit different architectures, the basic classifiers exhibit a remarkable diversity.
(3) For each basic classifier employed for fusion, it was trained on the entire training set rather than on a subset of the set. Therefore, all basic classifiers trained on the training set could see every instance of the training set.

There is a need to study the influence of dropout probability on testing accuracy, and the relationship between testing accuracy and dropout probability. The upper bound of the number of dropout nodes is required. Moreover, the effect of random initialization with different probability distributions like exponential distribution, the Gaussian distribution, uniform distribution, etc.) on testing accuracy needs to be studied.

In summary, the use of embedded dissimilarity and cost-sensitive factors is capable of boosting classification stability and minimizing the cost of classification for high-scale, imbalanced, and redundant datasets. Moreover, the problem of data imbalance can be addressed by using an algorithm-based technique, and a leave-one-out cross-validation algorithm using Sherman-Morrison matrix inversion lemma, and using Cholesky decomposition, while network weights update could be done via the add-delete mechanism. Furthermore, ensemble method can be optimized by incorporation of reactivated regularization such as L_1 and L_2 regularisation penalty, as in Tikhonov regularized OP-ELM (TROP-ELM) to promote prediction accuracy and minimized and stabilize MSE with an increasing number of neurons.

5. Poor capability of sample structure preserving (SSP)

Despite the recent prominence of ELM due to remarkable learning speed, little or no manual intervention, and good generalization performance, the generalization performance could be worse than that of SVM algorithm for small sample cases or small network size. This is ascribed to the use of Monte Carlo (MC) sampling technique for the generation of random input weights. ELM models with small network size exhibit random input weight with poor SSP capability, leading to poor generalization performance [20]. Several authors have attempted to seek panacea to this by enhancing ELM using manifold learning techniques such as Riemannian metric [81], graph Laplacian [82] and manifold regularization [83,84]. This has only helped in preserving the local neighbor structure information as the output weights learning proceeds without considering the local structure loss resulting from random input weights, which is more crucial to the performance of the overall learning [20]. Wang and Lu [20] made a concerted effort towards proffering a remarkable solution to poor SSP capability by using Quasi-Monte Carlo (QMC) technique, and unified random orthogonal (RO) projection technique for both regression and classification datasets. They reported that the QMC technique exhibited a deteriorated SSP performance worse than or similar to that of the MC technique for classification (Sonar, Leukemia and Conlon Cancer cases) datasets (Fig. 6). On the other hand, both the QMC and RO projection technique exhibited a remarkable improvement in the SSP capability for regression datasets (AutoMPG, Concrete Slump, and Machine CPU) (Fig. 6). Therefore, the RO projection technique exhibited a remarkable solution to poor SSP capability for both regression and classification datasets. Yang [85] developed a novel approach based on ELM and human visual system (HVS) as an algorithm for multi-focus image

Algorithm 1 Construction of the random orthogonal projection matrix.

```
1. Input:
2. Input matrix $X_{n\times N}$ consists of N n-dimensional input vectors;
3. Objective dimension $L(define\ \tilde{n} = min\{L, n\})$;
4. Output:
5. The random orthogonal projection matrix $W_{L\times n}$.
6. Step 1: Generate a random matrix $A_{L\times\tilde{n}}$ by MC method.
7. Step 2: Orthogonalize A by column based on Gram–Schmidt orthogonalization
   method, and obtain $A^{orth}$
8. Step 3:
9. (1) If $L < n$
10. Compute $W^{pca}$, whose rows consist of the first $L$ loading vectors
    corresponding to the first $L$ principal components on input matrix X, and let
    $W_{L\times n} = A^{orth}W^{pca}$
11. (2) Else
12. $W_{L\times n} = A^{orth}$.
13. Return: random orthogonal projection matrix $W_{L\times n}$.
```

fusion. They reported a high structural similarity, making the algorithm superior to a number of widely used fusion techniques (Algorithm 1). The degree of structural similarity was measured using modified structural similarity metric (MSSIM) given as

$$MSSIM\begin{cases} t(\omega)SSIM(A,F|\omega)+(1-t(\omega))SSIM(B,F|\omega), \\ \quad for\ SSIM(A,B|\omega)\geq 0.75 \\ max\{SSIM(A,F|\omega),\ SSIM(B,F|\omega)\}, \\ \quad for\ SSIM(A,B|\omega)\leq 0.75 \end{cases} \tag{32}$$

Where *SSIM* represents the structural similarity of the image, $t(\omega)$ is the local weights and it is defined as:

$$t(\omega)=\frac{s(A|\omega)}{s(A|\omega)+s(B|\omega)} \tag{33}$$

$s(A|\omega)$ is the variance of the image A in ω.

In summary, the poor capability of sample structure preserving (SSP) can be remarkably handled by RO projection technique for both regression and classification datasets. An approach based on ELM and human visual system (HVS) as an algorithm is also effective and exhibit high structural similarity for multi-focus image fusion.

6. Difficulty in accommodating lateral inhibition by direct random feature mapping

The random feature mapping involves the introduction of optimization constraint (Fig. 7), which is the fundamental features of ELM. Feature mapping in ELM enhances its capacity for universal approximation and efficiency in training. Furthermore, random feature mapping enhances generalization performances and reduces over-fitting [2].

Physiological research revealed that similar layer neurons are laterally inhibited to each other such that the outputs of each layer are sparse [86]. Meanwhile, it is difficult for ELM to accommodate lateral inhibition by direct use of random feature mapping [21]. Yu and Sun [21] developed a sparse coding ELM (ScELM) algorithm by using sparse coding method for mapping of the inputs layer to the hidden layer instead of the random mapping employed by the traditional ELM. The encoding stage was done using gradient projection (GP) based technique with l1 norm optimization [87], while Lagrange multiplier algorithm was used in learning the output weights between hidden and output layers. The sparsity in the proposed ScELM is to make the hidden-layer feature representations more relevant and unique to enhance classification performance.

In summary, the use of sparse coding method rather than random mapping for mapping of the inputs layer to the hidden layer by using gradient projection (GP) based technique with l1 norm optimization for the encoding stage, and Lagrange multiplier algorithm in learning the output weights between hidden and output layers.

7. Challenges with big data

The global interest in Big data, as well as its economic value and adoption, have continued to grow exponentially [88]. The advent of emerging technologies (such as Internet of Things (IoT) [89,90], Machine-to-Machine (M2M) communications [91], and cloud/fog computing [92–94]), and the high proliferation of smart mobile devices [95] has contributed immensely to the evolution of the Big data era. ELM has been widely and successfully applied to gain useful insights from data obtained in different domains including health [96–98], finance [99], commerce [100], aerospace [101,102], agriculture [103,104], and energy [105,106]. The adoption of conventional ELM for large-scale data learning has become a serious challenge for both academic researchers and industry experts. This is partly because classical ELM was primarily designed to run on a machine that has only one processing unit. In this case, all the instances in the training dataset are loaded on a single processor and the same output weight vector is used. In contrast, Big data are usually large in volume; they are often produced at high velocity, and they come from heterogeneous sources [107–109]. Consequently, information is usually stored in a distributed manner to ensure data confidentiality in most Big data applications. So, the ability of classical ELM to process large-scale data is restricted by limited available memory space.

Much work has been done to improve the efficiency of classical ELM such that data streams from different sources can be processed in parallel and distributed manner. The combined advantages of parallel processing and MapReduce framework [110–116] may be exploited to achieve big data learning in ELM. He et al. [117] introduced and applied Parallel Extreme Learning Machine (PELM) to solve the regression problem in Big datasets using MapReduce framework. In a similar work, Zhao et al. [118] successfully solved a large-scale recommendation problem in location-based social network applications using the combined technologies of parallel processing and MapReduce model. Even though the speed of large-scale data learning was much faster in PELM than in classical ELM, the storage of intermediate results on disks and multiple copies of each task increased the training overhead and reduced the training speed and accuracy of PELM. Duan et al. [119] achieved a higher learning speed and reduce the training cost in big data classification by implementing three sub-algorithms in parallel using Spark framework. The computations of the hidden layer output matrix and matrix decompositions were done locally. Meanwhile, intermediate results were retained in distributed memory and cache. In addition, the diagonal matrix was used as broadcast variables instead of making several copies for each task. The details of the comparative analysis of the developed algorithm with other ELM variants are provided in Table 8 and Fig. 8. Freund [120] applied AdaBoost [121] technique to MapReduce-based parallel ELM in training massive dataset. Adaboost method improved the learning accuracy and reduced the training cost. Luo et al. [122] suggested the use of a distributed model for the implementation of ELM algorithm in parallel machines. They designated an output weight vector to a particular processor and computed a common output weight vector using Alternating Direction Method of Multipliers (ADMM) technique [123].

Moreover, a bulk of the training time is used up on the computation of Moore–Penrose generalized inverse of the hidden layer matrix In classical ELM [124]. When dealing with non-singular matrices, Singular Value Decomposition (SVD) technique of solving this matrix tends to perform better than iterative, orthogonalization, and orthogonal techniques [38,50,125]. However, the *i*

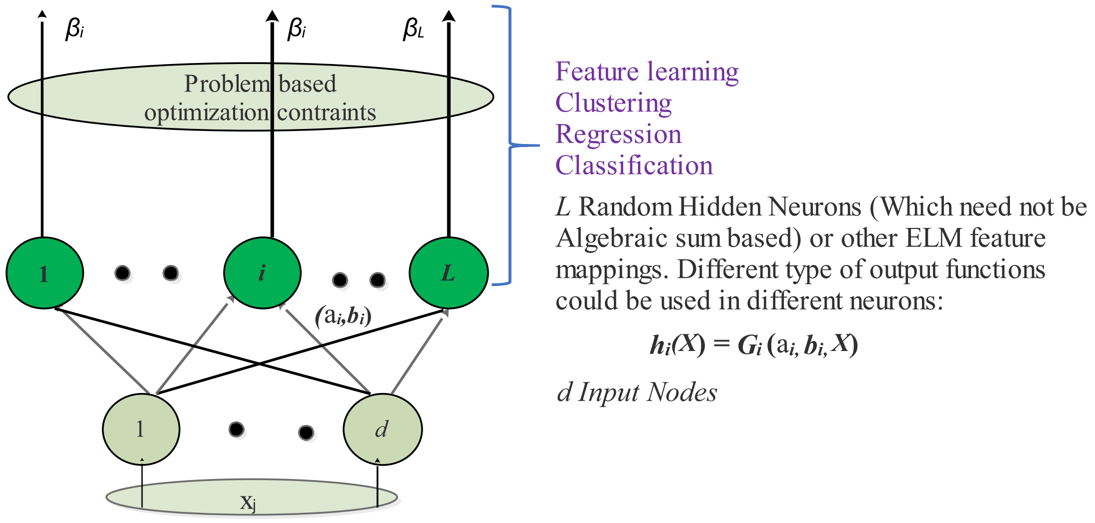
β_i
β_i
β_L
Problem based optimization contraints
Feature learning
Clustering
Regression
Classification
L Random Hidden Neurons (Which need not be Algebraic sum based) or other ELM feature mappings. Different type of output functions could be used in different neurons:
$h_i(X) = G_i(a_i, b_i, X)$
d Input Nodes
1
i
L
(a_i, b_i)
1
d
x_j

Fig. 7. ELM random feature mapping [1].

Table 8
Performance evaluation of ELM, PELM, ELM*, ELM*-Improved, and SELM on different large-scale datasets [119].

Dataset	ELM		PELM		ELM*		ELM*-Improved		SELM	
	Training accuracy	*Testing accuracy*	*Training accuracy*	*Testing accuracy*	*Training accuracy*	*Testing accuracy*	*Training accuracy*	*Testing accuracy*	*Training accuracy*	*Testing accuracy*
Patient	0.7842	0.7708	0.7916	0.7689	0.7759	0.7659	0.7903	0.7619	0.7981	0.7689
Outpatient	0.7794	0.7602	0.7654	0.7684	0.7812	0.7626	0.7871	0.7542	0.7801	0.7589
Medicine	0.7642	0.7512	0.7589	0.7601	0.7512	0.7488	0.7546	0.7498	0.7588	0.7568
Breast Cancer	0.7215	0.7204	0.7406	0.7389	0.7386	0.7402	0.7462	0.7418	0.7506	0.7584
Heart Disease	0.6974	0.7012	0.7422	0.7428	0.7399	0.7482	0.7368	0.7435	0.7501	0.7424
Chronic Kidney Disease	0.7934	0.8012	0.8022	0.8101	0.7873	0.7917	0.7901	0.8044	0.7894	0.8103
Hepatitis	0.8019	0.8125	0.7982	0.8013	0.7948	0.8093	0.7899	0.8201	0.7934	0.8094
Gastritis	0.7943	0.8002	0.8014	0.7945	0.8011	0.7917	0.7984	0.7911	0.8045	0.7967
Hypertension	0.7612	0.7741	0.7904	0.8017	0.7933	0.8049	0.8023	0.8113	0.7991	0.8117

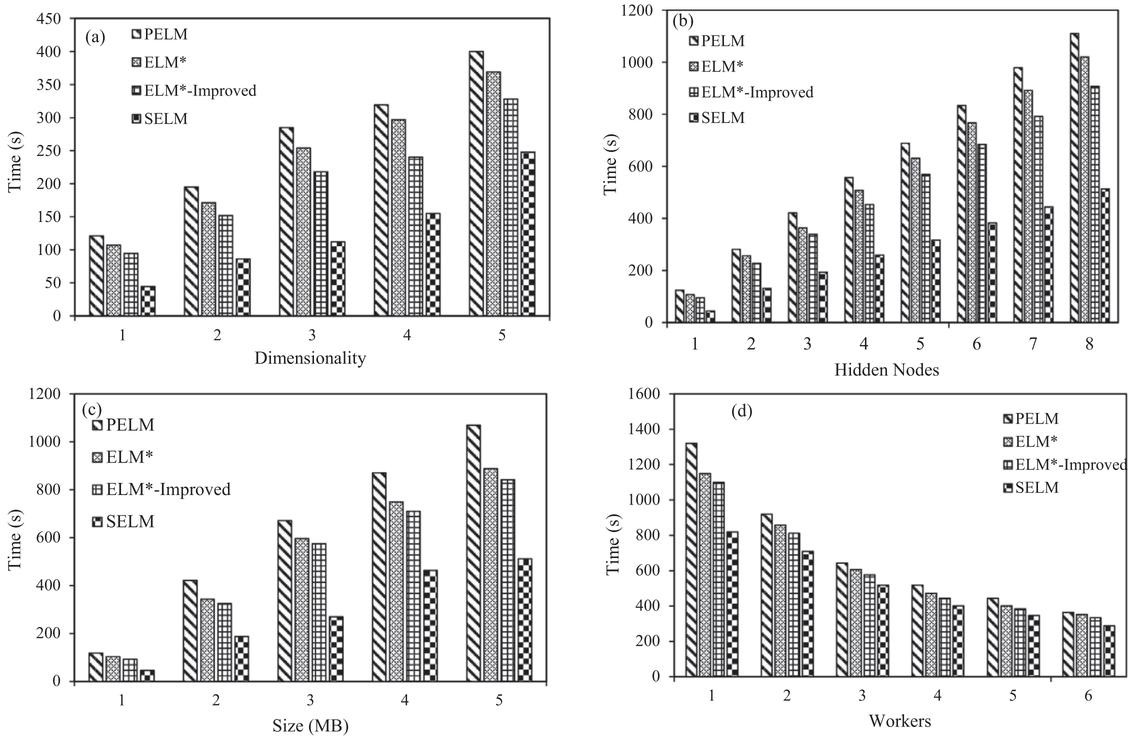

Fig. 8. Runtime under different conditions: (a) dimensionality (b) hidden nodes (c) size of samples (d) workers [119].

processing time in SVD method increases as the volume of the training data becomes huge [126]. The capability of SVD to handle large-scale data was extended in [127] and the CANDECOMP and TUCKER decomposition methods were realized. Massive data can, therefore, be trained in a cost-effective manner using scalable tensor decomposition based ELM [128]. Tucker decomposition and PARAFAC decomposition outperformed the classical ELM when dealing with scalability issues. Application of other forms of TUCKER and CANDECOMP/PARAFAC decomposition methods may further enhance the scalability and reduce the dimension of attributes required to the desired minimum [128].

In big data classification applications, there are usually occasions when new data instances are to be added. At other times, the removal of outdated data instances, and correction of errors in existing training data may become necessary. In each of these cases, traditional ELM requires retraining with the updated training dataset; and this consumes more time since the training dataset is considered huge. Therefore, it becomes very difficult to manage the rapid updating of large-scale training data. However, the retraining process can be restricted to the overlapped information between the old training dataset and the updated training dataset to reduce the training time. Xin et al. [129] proposed another variant of ELM to handle the challenging task of rapid updating in big data learning. The matrix multiplication was found to be most time-consuming and its computation can be achieved through incremental, decremental, and correctional learning methods. Hence, traditional ELM can be improved to accommodate these learning methods.

Learning algorithms developed based on kernels are known for their good generalization performance [130]. In Kernel Extreme Learning Machine (KELM), a kernel-based learning method is introduced into classical ELM to improve its generalization performance. This variant of ELM has been widely employed for both regression and classification applications in different fields [131–139]. Unlike other kernel-based learning methods, KELM does not consider bias factor as part of its optimization constraints. The elimination of the bias term is responsible for the improved generalization performance and the reduced computational requirements in KELM. However, the generalization ability of KELM is much limited to small-scale datasets since its performance on big datasets was reported to be relatively low [5,140]. In addition, large-scale data learning in KELM consumes more time because all the instances in the large-scale training dataset are used as support vectors [141]. This will consequently increase the size of the kernel matrix and it will make the process less computationally efficient. Deng et al., [140] proposed that the choice of the support vectors should be a portion of the Big dataset. By so doing, the kernel matrix will be reduced and the output weight matrix can be computed using Karush–Kuhn–Tucker (KKT) theorem [142]. In most practical Big data applications, the number of support vectors is usually far less than the number of training samples. In order to realize good estimation of any continuous target function in kernel-based ELM, it was proven in [140] that only the Strict Positive Definite (SPD) kernels [143] can be used because they possess the required universal learning properties as explained in [25]. The resulting kernel-based supervised algorithm is called Reduced Kernel Extreme Learning Machine (RKELM). The generalization performance and the computational efficiency of RKELM were tested on six different large-scale datasets. When compared to KELM, the new variant of ELM performed better with faster training speed and the large-scale learning process requires less memory space. The higher efficiency exhibited by RKELM was attributed to the sparsity of the datasets. In addition, a considerable amount of training time was saved because the support vectors were produced by only a few non-zero elements. On the other hand, traditional ELM wasted much time in the learning process because support vectors were produced for all elements of the sparse datasets. A considerably large number of support vectors will introduce the challenge of over-fitting in RKELM. This problem can be handled by properly fine-tuning the control parameter during the minimization process of the output weight function. However, further work must be done to address the sparseness of RKELM. In another work [144], the iterative learning operation was eliminated to achieve an efficient and fast multi-label classification algorithm.

In summary, the adoption of parallel computing based on MapReduce model has demonstrated better efficiency in terms of training speed, accuracy, and generalization performance than the conventional serial processing methods. To properly handle the problems of additional overhead and relatively slow training speed caused by intermediate operations in parallel ELM, the large-scale training data can be partitioned to enable local computations of parallel algorithms on Spark framework. By so doing, SELM produced the best speedup relative to the performance of PELM, ELM*, and ELM*-Improved algorithms.

8. More on big data: acceleration of ELM for practical big data tasks

To extend the capability of ELM for handling large scale or big data tasks (such as image classification, voice recognition, and object detection & tracking), authors in [145,146] proposed multilayer or hierarchical ELM (H-ELM) frameworks that are based on the universal approximation capability of the original ELM. Thus, H-ELM has extended the original ELM from shallow architecture to deep architecture, which has potentially provided a boost to the capability of ELM. Generally, deep architectures involve two major operations, namely, unsupervised multilayer features extraction and supervised features classification [145]. These operations require large amounts of computing resources because of the intensive matrix operations that are involved and cost plenty of computational time on normal Central Processing Unit (CPU). Thus, researchers have explored several methods to enhance the utilization of ELM for processing tasks in the big data domain. Examples of such methods include Hadoop's MapReduce framework, Graphical Processing Unit (GPU) as well as a hybrid of in-memory and GPU computing.

Hadoop's MapReduce framework is an open source model created by Google for processing large data sets through automatic parallelization and execution on a large cluster of distributed computers. The runtime machine in the framework handles machine failures and also schedules inter-machine communication for efficient use of network resources and the Hadoop Distributed File System (HDFS) based disk operations [110]. There are two phases of computation in any MapReduce job, which are the map function and reduce function and they are fully parallelized in the MapReduce framework. He et al. [117] developed an efficient Parallel ELM (PELM) based on MapReduce framework for regression tasks with very large-scale datasets. The authors designed two MapReduce jobs to implement the parallel ELM algorithm. The first involves the calculation of hidden node output matrix $\mathbf{H}$ and the second is the calculation of $\mathbf{H}^{\mathbf{T}}$ **x** $\mathbf{H}$ and $\mathbf{H}^{\mathbf{T}}$ **x** $\mathbf{T}$ matrices [117]. The presented experimental results by the authors showed that the algorithm did not only process large-scale datasets but also achieved good speedup, scaleup, and size up performances.

Xin et al. [147] developed the ELM*, which is a distributed ELM based on MapReduce to further improve on the computational efficiency of PELM [117]. The authors implemented the Moore Penrose inverse matrix using the parallel attributes of MapReduce framework while the output weight vector in ELM was computed with centralized computing thereby utilizing just one MapReduce job unlike the two that was used in PELM.

Table 9
Comparison of averaged computation time (in second) using CPU and GPU for the different oELM Classifiers [154].

	CPU				**GPU**			
Dataset	oELM	$oELM_{under}$	$oELM_{over}$	$oELM_{cost}$	oELM	$oELM_{under}$	$oELM_{over}$	$oELM_{cost}$
HYP	13.04	12.02	26.56	13.04	1.32	1.19	1.19	1.32
LED	13.38	12.93	19.56	13.38	1.40	1.21	1.21	1.40
Poker	13.13	12.15	20.89	13.13	1.35	1.21	1.21	1.35
RBF_B	13.32	12.36	19.41	13.32	1.39	1.20	1.20	1.39
SEA	12.21	11.16	17.58	12.21	1.20	1.10	1.10	1.20
Average	13.02	12.12	20.80	13.02	1.33	1.18	1.18	1.33

Furthermore, Wang et al. [148] developed the Parallel Online Sequential Extreme Learning Machine (POS-ELM) by adapting the Online Sequential Extreme Learning Machine (OS-ELM) to the MapReduce framework in order to train incremental data samples in parallel. Several other scholars [149–152] have recently focused on the study of parallel ELM algorithm based on MapReduce framework. Despite these efforts, MapReduce is a disk-based operation and every stage must interact with other stages via the HDFS. Therefore, the overheads and the huge amount of disk I/O operations that are involved in MapReduce jobs are limiting factors in the MapReduce framework approach to ELM enhancement.

In recent years, Graphics Processing Units (GPUs) have arisen in the High-Performance Computing (HPC) domain as parallel processors due to their competitive prices and high computational power. With the introduction of the NVIDIA Compute Unified Device Architecture (CUDA) library, deep learning library called cuDNN as well as the generic OpenCL and OpenACC libraries, it has become easier to leverage on GPU for general-purpose (GPGPU) computation without having to rewrite algorithms as video card operations. Speedups of up to 300 times are achievable through execution of codes on a single GPU instead of a CPU, and with multiple GPUs, higher speedups can be obtained [153]. Thus, scholars have leveraged on GPUs to enhance the computational efficiency of ELM for solving class imbalance problems and for big data applications. Krawczyk [154] proposed the implementation of online ELM (oELM) for class imbalance problem and utilized GPU for rapid updating of the classifier. According to the author, the approach produced a highly efficient mining of high-speed, drifting and imbalanced data streams with GPU providing significant acceleration. The methods investigated in the oELM to mitigate the class imbalance problem are undersampling ($oELM_{under}$), oversampling ($oELM_{over}$) and cost-sensitive adaptation ($oELM_{cost}$) and the benchmark datasets used for the experiments are HYP, LED, Poker and RBF_B. Table 9 clearly illustrates the acceleration achieved in the study when GPU was employed as against when the computation was done on CPU. As presented in the Table, the average improvement across the methods are oELM (89.78%), $oELM_{under}$ (90.26%), $oELM_{over}$ (94.33%) and $oELM_{cost}$ (91.30%).

Alia-Martinez et al. [155] developed a library in R language for GPU acceleration of ELM for big datasets. The authors performed a sensitivity analysis, which identified matrix multiplication as the most computationally demanding operations that consume 99% of execution time in ELM. Thus, this operation was executed on the GPU, which consequently achieved a speedup rate of about 15 times in most computation scenarios. In a codicil, a speedup rate of 6 times was achieved in the study when GPU was employed for the selection and training of thousands of models using a genetic algorithm to fine-tune ELM.

Moreover, GPU-acceleration of ELM has been employed by different authors in specific big data application areas such as remote sensing [156], hyperspectral image classification [157] and multimodal sentiment analysis [158]. In addition to a hybrid of in-memory computing with GPU as implemented in [159], CPU and GPU were also engaged cooperatively in [160] to further produce powerful high computational platforms for ELM implementation.

9. Difficulties in handling a block of data

Handling a block of data, or incremental training samples (one-by-one or chunk-by-chunk) is another serious drawback in ELM. Incremental training samples (such as dynamic changes of tidal level) are presented chunk-by-chunk or one by one and they involve time-varying dynamics. However, it is difficult to develop a model that is suitable for time-varying process, making identification and prediction challenging in real-time [22]. To obtain accurate predictions for time-varying systems in real time, development of an adaptive model, which exhibits adaptive structure and parameters, is practically required.

Several authors have employed sampling and learning technique called sequential learning. Sequential learning is an adaptive learning approach that processes data in a sequential manner and tunes network accordingly [161]. Sequential learning scheme was invented as a resource allocation network (RAN) that learns samples one by one [162]. The use of RAN makes the network more complex by sequential addition of new samples to the network. To proffer solution to this drawback, several variances of RAN have been developed [163].

Liang et al. and Sun et al. [13,164] investigated another kind of sequential learning technique called online sequential extreme learning machine (OS-ELM), which has the capacity to handle incremental training samples. This is because OS-ELM uses incremental data for adjustment of training data set towards implementing online learning, meaning that the system does not require all the training data to be available before computation [165] OS-ELM was obtained using the theory of ELM [1, 2], due to its exceptionally fast learning speed and generalization performance, and its performance has been assessed using several standard problems [166]. Migration from ELM to OS-ELM (as one of ELM variance) is a migration from batch learning to sequential learning, which is capable of handling a block of data with various chunk size chunk by chunk. Although OS-ELM is proficient in handling a block of data chunk-by-chunk with a faster training speed and good generalization performance, training with conventional OS-ELM engenders a continuous increase in dimension, resulting in “dimensionality curse”. Consequently, conventional OS-ELM is not appropriate for data that arrives ceaselessly or large-scale datasets, as well as online identification and prediction of time-varying systems whose dynamics are affected by environmental disturbances and inner condition.

For easy programming, the Moore–Penrose generalized inverse of the independent variable matrix was calculated SVD technique. As the number of samples becomes large, the solution to the multiple regression problems may not be feasible in “memory-resident” mode because the memory of a single processing unit will no longer accommodate the high-dimensional matrix. Therefore, a “disk-resident” parallel ELM algorithm was recommended for multiple regression systems. Wang et al. [148] developed a

Table 10
Comparison of accuracy of ELM algorithms with different block sizes [151].

Dataset	Chunk size (KB)	PEOS-ELM-B		PEOS-ELM-S		PEOS-ELM-C		PEOS-ELM-BSC	
		Training accuracy	*Testing accuracy*	*Training accuracy*	*Testing accuracy*	*Training accuracy*	*Testing accuracy*	*Training accuracy*	*Testing accuracy*
MNIST	1885	0.883	0.893	0.882	0.890	0.882	0.890	0.882	0.890
	3770	0.881	0.888	0.881	0.890	0.883	0.892	0.884	0.891
	7540	0.882	0.892	0.881	0.888	0.882	0.892	0.881	0.888
	15,080	0.881	0.890	0.880	0.890	0.883	0.891	0.882	0.892
DNA	9	0.955	0.927	0.956	0.922	0.956	0.922	0.959	0.922
	18	0.954	0.926	0.956	0.926	0.956	0.924	0.951	0.916
	36	0.951	0.925	0.958	0.924	0.951	0.917	0.953	0.922
	72	0.954	0.920	0.955	0.918	0.950	0.926	0.950	0.916
KDDCup99	8354	0.993	0.857	0.994	0.858	0.994	0.857	0.994	0.858
	16,708	0.994	0.858	0.994	0.857	0.994	0.858	0.994	0.858
	33,416	0.993	0.857	0.994	0.858	0.994	0.858	0.994	0.858
	66,832	0.994	0.858	0.994	0.858	0.993	0.858	0.994	0.858

new variant of ELM algorithm that calculates the hidden layer output matrix of big data fragments in parallel based on traditional ELM and OS-ELM algorithms to achieve efficient big data learning using MapReduce programming model. Unlike OS-ELM where reception of training data takes place at different times, training data are received at the same time in POS-ELM. The output weight matrix is computed in the same way in both OS-ELM and POS-ELM but the times spent in computing the hidden layer output matrix differ. The developed algorithm, Parallel Online Sequential Extreme Learning Machine (POS-ELM), can handle learning in big data while maintaining good accuracy as the volume of training data and the number of attribute increase. Specifically, POS-ELM achieves higher speedup with a larger volume of learning data. In addition to the suitability of POS-ELM for large-scale online sequential learning, the proposed algorithm can equally be adopted for learning on the data received at once. In recent time, some researchers [117,147,148,167,168] have implemented OS-ELM algorithm in parallel machines to reduce the time spent in training large-scale datasets; but the problems introduced by noisy dataset were not investigated. Huang et al. [151] developed a Parallel Ensemble of Online Sequential ELM (PEOS-ELM) algorithm to parallelize the computations of the hidden layer output matrix in Map phase and the output weights matrix in Reduced phase for different ensemble models. The results of the performance evaluation of PEOS-ELM relative to OS-ELM variants and its variants are shown in Table 10 and Fig. 9.

Recently, the use of WOS-ELMK [169] for Class Imbalance Learning (CIL) of big data streams has also been exploited but led to an increase in computational complexity. Ding et al. [169] proposed that a fixed-size window method can be used to cut the computational cost in WOS-ELMK. At each time step, a fixed amount of the oldest training samples is removed from the window. The training process in this method involves initiation, decremental learning, and sequential learning. The performance of this new ELM variant is comparable to that of VWOS-ELM proposed by Mirza and Lin [170]. Therefore, window-based ELMK can be used to save computational load in big data learning (Table 11).

Furthermore, OS-ELM could be improved by the addition of a pruning technique to the hidden units in addition to the original unit, which entails sequential addition of samples, making the network adaptive to real-time alteration of time-varying dynamics [22]. Yin [22] developed an online identification and prediction scheme called variable structure OS-ELM (VS-OSELM) for a time-varying system by adding a pruning strategy in OS-ELM to delete obsolete hidden units. In VS-OSELM, the representation ability of the hidden unit to the system dynamic is obtained by an index called normalized error reduction ratio (*nerr*). The hidden units with a small value of *nerr* are considered as outdated and removed from the network. Yin [22] implemented VS-OSELM for online prediction of tidal level Old Port Tampa in Florida, United States, is a complex time-varying process to assess its representation ability in time-varying schemes. During the identification process, the addition and pruning of hidden units are carried out concurrently. Upon completion of the network structure adjustment, the connection parameters are computed. The simulation results show that VS-OSELM is efficient for identifying and predicting complex time-varying systems with fast learning speed and high prediction accuracy (Table 10).

Luo et al. [165] seek to improve OS-ELM via incorporation of timeliness management structure into ELM for learning data. The developed algorithm is referred to as timeliness online-regularized extreme learning machine (TORELM). TORELM accepts training data that arrives the system one-by-one with varied or fixed chunk size under the same framework (Algorithm 2). Algorithm 2 presents the step-by-step list of instructions on how to train the neural network using TORELM learning method. First, weight w_j, bias b_j, and a small value σ are assigned in a random manner for the value of $j = 1, \dots, L$, where L represents the number of hidden neurons. Then, the output matrix of the hidden layer, $\mathbf{H_o}$, is computed. The initial output weight, β_o, is updated and the first group of data, $\mathbf{N_s}$, is added in order to obtain the corresponding values of $\mathbf{H}_1$ and β_1 when $k = 1$. The initial model is formulated at $j = 0$ based on Eq. (34):

$$\beta_{k+1} = \beta_k + w.\mathrm{K}_{k+1}^{\mathrm{T}}\mathrm{H}_{k+1}^{\mathrm{T}}(\mathrm{T}_{k+1} - \mathrm{H}_{k+1}\beta_k) \tag{34}$$

However, the training process is brought to a halt whenever $|\beta_{k(j+1)} - \beta_{k(j)}|$ turns to be less than the value of σ. In this case, the output weight is obtained at the instance of $k(j+1)$ and the value of k increases by one. Otherwise, the model is computed based on Eq. (35) and the value of j is incremented by one.

$$\beta_{k(j+1)} = \beta_{k(j)} + w.\mathrm{K}_{k+1}^{\mathrm{T}}\mathrm{H}_{k+1}^{\mathrm{T}}\left(\mathrm{T}_{k+1} - \mathrm{H}_{k+1}\beta_{k(j)}\right) \tag{35}$$

In a case where new incremental data are introduced, the initial model calculation is performed again. The absolute difference between the two consecutive output weights is tested to know whether the result obtained is less than the value of σ or not. The training process stops when none of the conditions mentioned earlier is satisfied.

TORELM algorithm strengthens the recent data and weakens the previous one in order to minimize both empirical risk and structural risk. Regularization technique combined with the timeliness structure of TORELM by using a weight factor to maintain balance towards achieving superior generalization performance. This

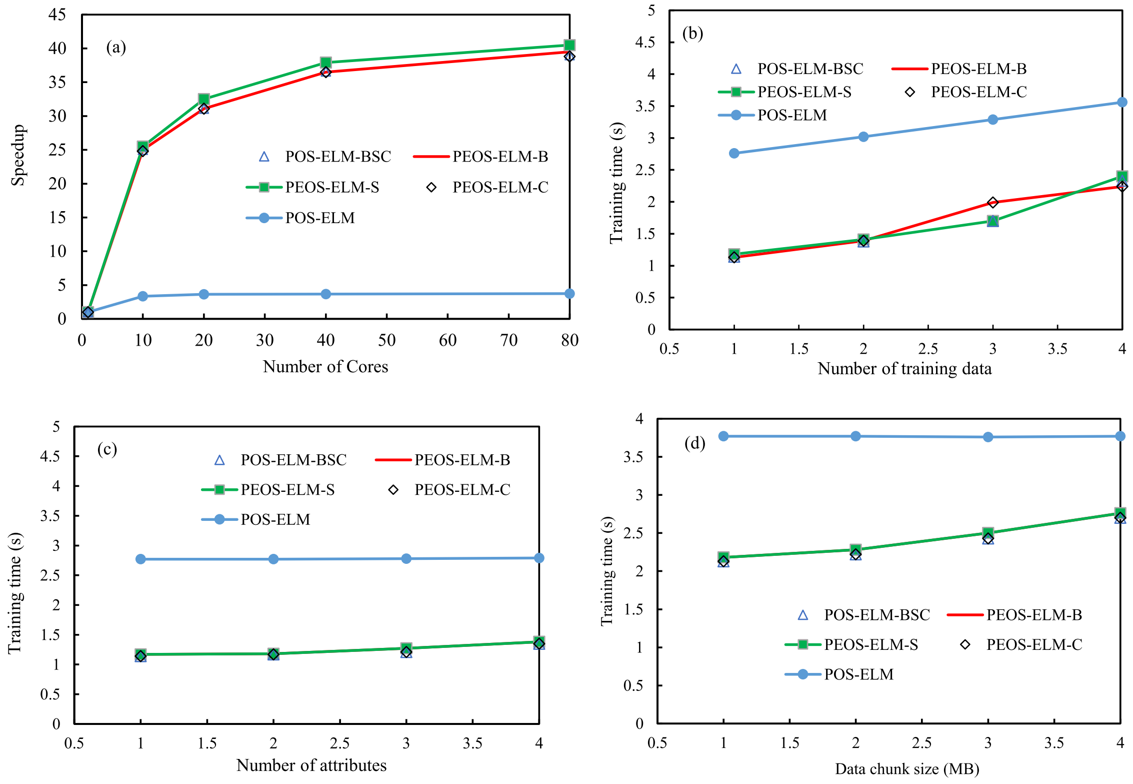

Fig. 9. Scalability of ELM algorithms with respect to (a) number of cores, (b) number of large-scale training data, (c) number of attributes (d) data chunk size [151].

Table 11
Comparison of tidal level prediction simulation results [116].

Prediction methods	VS-OSELM			OS-ELM			MRAN		
Simulation results	*RM SEP*	CC	Time	*RM SEP*	CC	Time	*RM SEP*	CC	Time
1-hour-ahead	0.0365	0.995	17.8981	0.0503	0.9791	1826.8976	0.0694	0.9698	66.1048
2-hours-ahead	0.0438	0.9844	18.6718	0.0706	0.9587	1718.9116	0.0856	0.9591	67.2609
3-hours-ahead	0.0694	0.9614	18.8557	0.0958	0.9233	1701.0416	0.0906	0.9484	67.9703
6-hours-ahead	0.1159	0.8811	19.1606	0.1082	0.9008	1742.0923	0.1403	0.8753	69.9908
12-hours-ahead	0.1271	0.871	19.7699	0.127	0.8595	1716.1559	0.1661	0.8649	74.3415
24-hours-ahead	0.129	0.8674	19.6549	0.127	0.8598	1735.41	0.1953	0.8421	75.9229

Algorithm 2 Framework of TORELM.

```
1. Start
2. Input $N_o$ initial training samples $\mathbf{N_s}$, the number of hidden neurons $L$, and
   activation function $\mathbf{g}(\cdot)$
3. Randomly assign weight $w_j$, bias $b_j$, and a small value $\sigma$ ($j = 1, \ldots, L$)
4. Calculate hidden layer output matrix $\mathbf{H_o}$ and $\alpha = -\gamma(\mathbf{H_o}\beta_o - \mathbf{T_o})^{\mathrm{T}}$, $\varepsilon_i = \alpha_i/\gamma$
5. Update the initial output weight $\beta_o = (\frac{\mathbf{I}}{\gamma} + \mathbf{H_o^T D^2 H_o})^{\dagger}\mathbf{H_o^T D^2 T_o}$
6. Add a group of data $\mathbf{N_s}$, calculate corresponding $\mathbf{H}_1$ and $\beta_1$, and let $k = 1$
7. Let $j = 0$ and calculate model: $\beta_{k+1} = \beta_k + \mathbf{w}.\mathbf{K}_{k+1}^{\mathrm{T}}\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\beta_k)$
8. **if** $|\beta_{k(j+1)} - \beta_{k(j)}| < \sigma$ **then**
9. Stops and obtains $\beta_{k(j+1)}$
10. $k = k + 1$
11. **else**
12. $\beta_{k(j+1)} = \beta_{k(j)} + \mathbf{w}.\mathbf{K}_{k+1}^{\mathrm{T}}\mathbf{H}_{k+1}^{\mathrm{T}}(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\beta_{k(j)})$
13. $j = j + 1$
14. **if** new incremental data **then**
15. **goto** #7
16. **else**
17. Stop
```

help to address the imbalance between structural risk and empirical risk in a number of traditional learning techniques. The simulation results reveal that TORELM can achieve better stability and higher learning accuracy than other variance of ELM such as TMELM-based, WOSELM-based, and OSELM-based techniques (Fig. 10).

In summary, the difficulty in handling a block of data could be tackled by variable structure OS-ELM (VS-OSELM) for the time-varying system by adding a pruning strategy in OS-ELM to delete obsolete hidden units using normalized error reduction ratio (nerr) for measuring representation ability. Another recent method is timeliness online-regularized ELM (TORELM), which strengthens the recent data and weakens the previous one to minimize both structural risk and empirical risk to prevent "dimensionality curse".

Conclusion

This work gives a state-of-the-art review of the recent trend towards achieving an efficient and cost-effective ELM model. Various drawbacks of ELM as a machine learning technique such as difficulty in determination of hidden layer structure, prediction instability and Imbalanced data distributions, the poor capability of sample structure preserving (SSP), and difficulty in accommodating lateral inhibition by direct random feature mapping, were addressed. Other drawbacks addressed include multi-graph complexity, global memory size, one-by-one or chuck-by-chuck (a block of data), global memory size limitation, and challenges with big data.

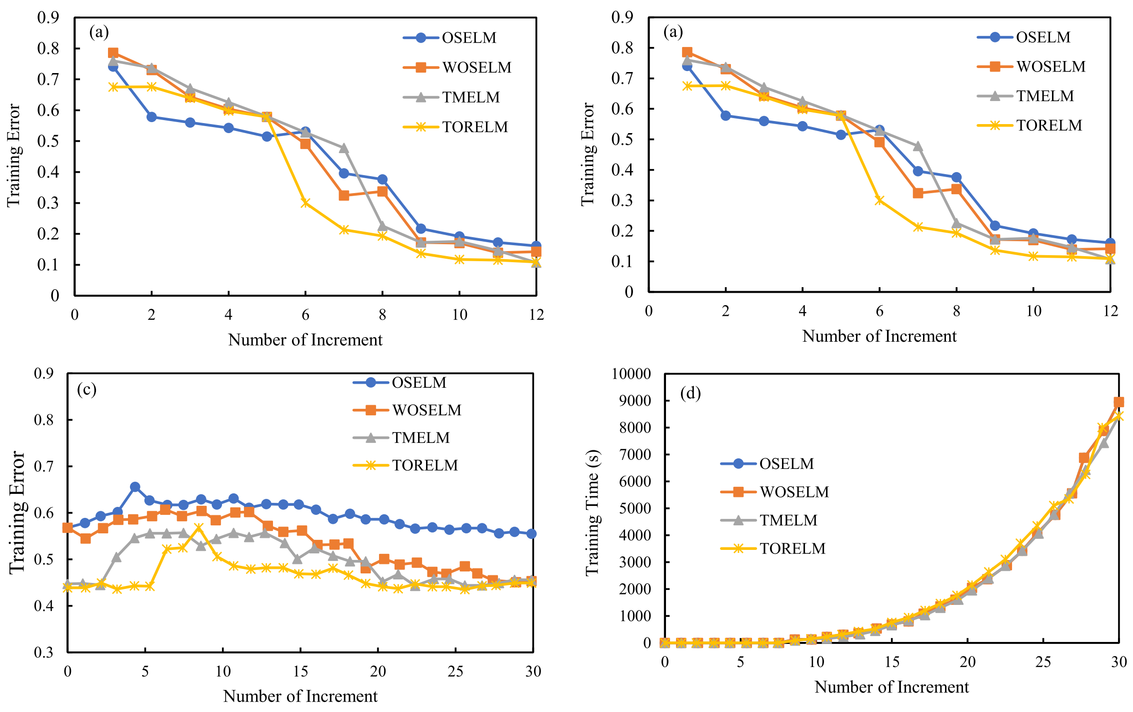

Fig. 10. Performance comparison of TORELM and other ELM-based learning algorithms based on (a & b) high-dimensional data set (Libras Movement with 15 classes of 24 instances each, and 91 attributes); (c & d) large scale data set (EEG Eye State with 14,980 instances and 15 attributes) [165].

The difficulty in determination of hidden layer structure has been addressed based on recent models proposed by various scholars working in the field, by reducing the computational complexity of the Moore–Penrose generalized inverse matrix. This could be achieved using ginv function, Conjugate Gram–Schmidt process, preconditioned Krylov subspace technique, Tikhonov regularization, QR factorization or LU factorization, which are faster iterative methods for computation when compared with the traditional SVD method. By using these alternative techniques, the problems of data imbalance and prediction instability will be alleviated. Moreover, the problem of data imbalance can be addressed by using an algorithm-based technique, and a leave-one-out cross-validation algorithm using Sherman-Morrison matrix inversion lemma, and using Cholesky decomposition, while network weights update could be done via the add-delete mechanism.

The poor capability of sample structure preserving (SSP) has been remarkably handled by RO projection technique for both regression and classification datasets. An approach based on ELM and human visual system (HVS) as an algorithm is also effective and exhibit high structural similarity for multi-focus image fusion.

To enable ELM to accommodate lateral inhibition, the use of sparse coding method rather than random mapping for mapping of the inputs layer to the hidden layer by using gradient projection (GP) based technique with L1 norm optimization for the encoding stage, and Lagrange multiplier algorithm in learning the output weights between hidden and output layers.

On the use of ELM for Big data learning, the adoption of parallel computing based on Hadoop's MapReduce framework and GPU acceleration have demonstrated better efficiency in terms of training speed, accuracy, and generalization performance than the conventional serial processing methods. To properly handle the problems of additional overhead and relatively slow training speed caused by intermediate operations in parallel ELM, the large-scale training data can be partitioned to enable local computations of parallel algorithms on Spark framework. By so doing, SELM produced the best speedup relative to the performance of PELM, ELM*, and ELM*-Improved algorithms.

The difficulty in handling a block of data could be tackled by variable structure OS-ELM (VS-OSELM) for the time-varying system by adding a pruning strategy in OS-ELM to delete obsolete hidden units using normalized error reduction ratio (*nerr*) for measuring representation ability. Another recent method is timeliness online-regularized ELM (TORELM), which strengthens the recent data and weakens the previous one in order to minimize both structural risk and empirical risk to prevent "dimensionality curse".

Acknowledgments

The authors acknowledge the Fundamental Research Grant Scheme (FRGS) from the University of Malaya for funding this work through Project no. "FP046-2017A".

References

[1] G.-B. Huang, An insight into extreme learning machines: random neurons, random features and kernels, Cogn. Comput. 6 (2014) 376–390.
[2] G. Huang, G.-B. Huang, S. Song, K. You, Trends in extreme learning machines: a review, Neural Netw. 61 (2015) 32–48.
[3] M. Hossain, S. Mekhilef, M. Danesh, L. Olatomiwa, S. Shamshirband, Application of extreme learning machine for short term output power forecasting of three grid-connected PV systems, J. Clean. Prod. 167 (2017) 395–405.
[4] S.Y. Wong, K.S. Yap, H.J. Yap, A constrained optimization based extreme learning machine for noisy data regression, Neurocomputing 171 (2016) 1431–1443.
[5] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 42 (2012) 513–529.
[6] H. Zhu, E.C. Tsang, X.-Z. Wang, R.A.R. Ashfaq, Monotonic classification extreme learning machine, Neurocomputing 225 (2017) 205–213.
[7] A. Ekbal, S. Saha, Simultaneous feature and parameter selection using multi-objective optimization: application to named entity recognition, Int. J. Mach. Learn. Cybern. 7 (2016) 597–611.
[8] S. Yu, H. Chen, Q. Wang, L. Shen, Y. Huang, Invariant feature extraction for gait recognition using only one uniform model, Neurocomputing 239 (2017) 81–93.
[9] M.-B. Li, G.-B. Huang, P. Saratchandran, N. Sundararajan, Fully complex extreme learning machine, Neurocomputing 68 (2005) 306–314.
[10] G.-B. Huang, Q.-Y. Zhu, C.K. Siew, Real-time learning capability of neural networks, IEEE Trans. Neural Netw. 17 (2006) 863–878.

[11] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally pruned extreme learning machine, IEEE Trans. Neural Netw. 21 (2010) 158–162.
[12] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the IEEE International Joint Conference on Neural Networks, IEEE, 2004, pp. 985–990.
[13] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (2006) 1411–1423.
[14] F. Han, M.-R. Zhao, J.-M. Zhang, Q.-H. Ling, An improved incremental constructive single-hidden-layer feedforward networks for extreme learning machine based on particle swarm optimization, Neurocomputing 228 (2017) 133–142.
[15] J. Zhai, L. Zang, Z. Zhou, Ensemble dropout extreme learning machine via fuzzy integral for data classification, Neurocomputing 275 (2018) 1043–1052.
[16] M.D. McDonnell, M.D. Tissera, A. van Schaik, J. Tapson, Fast, simple and accurate handwritten digit classification using extreme learning machines with shaped input-weights. arXiv preprint arXiv:arXiv:1412.8307 (2014).
[17] W. Xiao, J. Zhang, Y. Li, S. Zhang, W. Yang, Class-specific cost regulation extreme learning machine for imbalanced classification, Neurocomputing 261 (2017) 70–82.
[18] Y. Zhang, B. Liu, J. Cai, S. Zhang, Ensemble weighted extreme learning machine for imbalanced data classification based on differential evolution, Neural Comput. Appl. 28 (2017) 259–267.
[19] S.-J. Lin, C. Chang, M.-F. Hsu, Multiple extreme learning machines for a two-class imbalance corporate life cycle prediction, Knowl. Based Syst. 39 (2013) 214–223.
[20] W. Wang, X. Liu, The selection of input weights of extreme learning machine: a sample structure preserving point of view, Neurocomputing 261 (2017) 28–36.
[21] Y. Yu, Z. Sun, Sparse coding extreme learning machine for classification, Neurocomputing 261 (2017) 50–56.
[22] J.-C. Yin, A variable-structure online sequential extreme learning machine for time-varying system prediction, Neurocomputing 261 (2017) 115–125.
[23] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[24] A.J. Annema, K. Hoen, H. Wallinga, Precision requirements for single-layer feedforward neural networks, in: Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems, IEEE, 1994, pp. 145–151.
[25] G.-B. Huang, L. Chen, C.K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (2006) 879–892.
[26] R. Singh, S. Balasundaram, Application of extreme learning machine method for time series analysis, Int. J. Intell. Technol. 2 (2007) 256–262.
[27] B. Widrow, A. Greenblatt, Y. Kim, D. Park, The No-Prop algorithm: a new learning algorithm for multilayer neural networks, Neural Netw. 37 (2013) 182–188.
[28] J. Tapson, A. van Schaik, Learning the pseudoinverse solution to network weights, Neural Netw. 45 (2013) 94–100.
[29] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (2008) 3460–3468.
[30] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.
[31] H.-J. Rong, Y.-S. Ong, A.-H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, Neurocomputing 72 (2008) 359–366.
[32] A.S. Alencar, A.R.R. Neto, J.P.P. Gomes, A new pruning method for extreme learning machines via genetic algorithms, Appl. Soft Comput. 44 (2016) 101–107.
[33] G. Feng, Y. Lan, X. Zhang, Z. Qian, Dynamic adjustment of hidden node parameters for extreme learning machine, IEEE Trans. Cybern. 45 (2015) 279–288.
[34] R. Zhang, Y. Lan, G.-B. Huang, Z.-B. Xu, Y.C. Soh, Dynamic extreme learning machine and its approximation capability, IEEE Trans. Cybern. 43 (2013) 2054–2065.
[35] W. Guo, T. Xu, Z. Lu, An integrated chaotic time series prediction model based on efficient extreme learning machine and differential evolution, Neural Comput. Appl. 27 (2016) 883–898.
[36] F. Toutounian, F. Soleymani, An iterative method for computing the approximate inverse of a square matrix and the Moore–Penrose inverse of a non-square matrix, Appl. Math. Comput. 224 (2013) 671–680.
[37] F. Toutounian, A. Ataei, A new method for computing Moore–Penrose inverse matrices, J. Comput. Appl. Math. 228 (2009) 412–417.
[38] V.N. Katsikis, D. Pappas, Fast computing of the Moore–Penrose inverse matrix, Electron. J. Linear Algebra 17 (2008) 637–650.
[39] Y.-P. Zhao, Z.-Q. Li, P.-P. Xi, D. Liang, L. Sun, T.-H. Chen, Gram–Schmidt process based incremental extreme learning machine, Neurocomputing 241 (2017) 1–17.
[40] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Trans. Neural Netw. 20 (2009) 1352–1357.
[41] M. Benzi, M. Tuma, A robust preconditioner with low memory requirements for large sparse least squares problems, SIAM J. Sci. Comput. 25 (2003) 499–512.
[42] S. Wang, E.d. Sturler, G.H. Paulino, Large-scale topology optimization using preconditioned Krylov subspace methods with recycling, Int. J. Numer. Methods Eng. 69 (2007) 2441–2468.
[43] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.
[44] W. Li, Z. Li, A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix, Appl. Math. Comput. 215 (2010) 3433–3442.
[45] H. Homeier, On Newton-type methods with cubic convergence, J. Comput. Appl. Math. 176 (2005) 425–432.
[46] M.D. Tissera, M.D. McDonnell, Deep extreme learning machines: supervised autoencoding architecture for classification, Neurocomputing 174 (2016) 42–49.
[47] Y. Chen, J. Feng, Efficient method for Moore–Penrose inverse problems involving symmetric structures based on group theory, J. Comput. Civil Eng. 28 (2012) 182–190.
[48] C. Jhurani, L. Demkowicz, Multiscale modeling using goal-oriented adaptivity and numerical homogenization. Part II: algorithms for the Moore–Penrose pseudoinverse, Comput. Methods Appl. Mech. Eng. 213 (2012) 418–426.
[49] A.N. Tikhonov, Regularization of incorrectly posed problems, Sov. Math. Dokl. 4 (1963) 1624–1627.
[50] A. Ben-Israel, T.N. Greville, Generalized Inverses: Theory and Applications, Springer Science & Business Media, 2003.
[51] I.U.E. Boiarintsev, Methods of Solving Singular Systems of Ordinary Differential Equations, John Wiley & Sons Incorporated, 1992.
[52] A.E. Hoerl, R.W. Kennard, Ridge regression: applications to nonorthogonal problems, Technometrics 12 (1970) 69–82.
[53] R. Mehta, N. Rajpal, V.P. Vishwakarma, LWT-QR decomposition based robust and efficient image watermarking scheme using Lagrangian SVR, Multimed. Tools Appl. 75 (2016) 4129–4150.
[54] W. Song, J.-j. Hou, Z.-h. Li, L. Huang, Chaotic system and QR factorization based robust digital image watermarking algorithm, J. Cent. South Univ. Technol. 18 (2011) 116–124.
[55] A. Bjorck, Numerical methods for least squares problems, Siam, 1996.
[56] T. Davis, Multifrontral multithreaded rank-revealing sparse QR factorization, in: Dagstuhl Seminar Proceedings, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.
[57] Y. Naderahmadian, S. Hosseini-Khayat, Fast watermarking based on QR decomposition in wavelet domain, in: Proceedings of the Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), IEEE, 2010, pp. 127–130.
[58] H.-y. Chen, Y.-s. Zhu, A robust watermarking algorithm based on QR factorization and DCT using quantization index modulation technique, J. Zhejiang Univ. Sci. C 13 (2012) 573–584.
[59] P.S. Stanimirović, M.B. Tasić, Computing generalized inverses using LU factorization of matrix product, Int. J. Comput. Math. 85 (2008) 1865–1878.
[60] B. Zhang, Z. Ma, Y. Liu, H. Yuan, L. Sun, Ensemble based reactivated regularization extreme learning machine for classification, Neurocomputing 275 (2018) 255–266.
[61] S. Garcı, I. Triguero, C.J. Carmona, F. Herrera, Evolutionary-based selection of generalized instances for imbalanced classification, Knowl. Based Syst. 25 (2012) 3–12.
[62] M. Gao, X. Hong, S. Chen, C.J. Harris, A combined SMOTE and PSO based RBF classifier for two-class imbalanced problems, Neurocomputing 74 (2011) 3456–3466.
[63] A. Estabrooks, T. Jo, N. Japkowicz, A multiple resampling method for learning from imbalanced data sets, Comput. Intell. 20 (2004) 18–36.
[64] Y. Akbulut, A. Şengür, Y. Guo, F. Smarandache, A novel neutrosophic weighted extreme learning machine for imbalanced data set, Symmetry 9 (2017) 142.
[65] W. Mao, M. Jiang, J. Wang, Y. Li, Online extreme learning machine with hybrid sampling strategy for sequential imbalanced data, Cogn. Comput. 9 (2017) 780–800.
[66] N. García-Pedrajas, J. Pérez-Rodríguez, M. García-Pedrajas, D. Ortiz-Boyer, C. Fyfe, Class imbalance methods for translation initiation site recognition in DNA sequences, Knowl. Based Syst. 25 (2012) 22–34.
[67] D. Zhang, M.M. Islam, G. Lu, A review on automatic image annotation techniques, Pattern Recognit. 45 (2012) 346–362.
[68] Y. Liu, H. Lu, K. Yan, H. Xia, C. An, Applying cost-sensitive extreme learning machine and dissimilarity integration to gene expression data classification, Comput. Intell. Neurosci. 2016 (2016) 1–9.
[69] H.-J. Lu, E.-H. Zheng, Y. Lu, X.-P. Ma, J.-Y. Liu, ELM-based gene expression classification with misclassification cost, Neural Comput. Appl. 25 (2014) 525–531.
[70] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) 42 (2012) 463–484.
[71] X. Xue, M. Yao, Z. Wu, J. Yang, Genetic ensemble of extreme learning machine, Neurocomputing 129 (2014) 175–184.
[72] A. Mozaffari, N.L. Azad, Optimally pruned extreme learning machine with ensemble of regularization techniques and negative correlation penalty applied to automotive engine coldstart hydrocarbon emission identification, Neurocomputing 131 (2014) 143–156.
[73] A. Lendasse, A. Akusok, O. Simula, F. Corona, M. van Heeswijk, E. Eirola, Y. Miche, Extreme learning machine: a robust modeling technique? Yes!„ in: Proceedings of the International Work-Conference on Artificial Neural Networks, Springer, 2013, pp. 17–35.

[74] Y. Miche, M. Van Heeswijk, P. Bas, O. Simula, A. Lendasse, TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization, Neurocomputing 74 (2011) 2413–2421.
[75] G. Schwarz, Estimating the dimension of a model, Ann. Stat. 6 (1978) 461–464.
[76] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc. Ser. B (Stat. Methodol.) 67 (2005) 301–320.
[77] D.M. Allen, The relationship between variable selection and data agumentation and a method for prediction, Technometrics 16 (1974) 125–127.
[78] W.H. Press, Numerical Recipes: The Art of Scientific Computing, 3rd edition, Cambridge University Press, 2007.
[79] A. Akusok, K.-M. Björk, Y. Miche, A. Lendasse, High-performance extreme learning machines: a complete toolbox for big data applications, IEEE Access 3 (2015) 1011–1025.
[80] W. Mao, J. Wang, L. He, Y. Tian, Online sequential prediction of imbalance data with two-stage hybrid strategy by extreme learning machine, Neurocomputing 261 (2017) 94–105.
[81] W. Mao, Y. Zheng, X. Mu, J. Zhao, Uncertainty evaluation and model selection of extreme learning machine based on Riemannian metric, Neural Comput. Appl. 24 (7-8) (2014) 1613–1625.
[82] G. Huang, S. Song, J.N. Gupta, C. Wu, Semi-supervised and unsupervised extreme learning machines, IEEE Trans. Cybern. 44 (2014) 2405–2417.
[83] Y. Zhou, B. Liu, S. Xia, B. Liu, Semi-supervised extreme learning machine with manifold and pairwise constraints regularization, Neurocomputing 149 (2015) 180–186.
[84] B. Liu, S.-X. Xia, F.-R. Meng, Y. Zhou, Manifold regularized extreme learning machine, Neural Comput. Appl. 27 (2016) 255–269.
[85] Y. Yang, M. Yang, S. Huang, Y. Que, M. Ding, J. Sun, Multifocus image fusion based on extreme learning machine and human visual system, IEEE Access 5 (2017) 6989–7000.
[86] B.A. Olshausen, D.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by V1? Vis. Res. 37 (1997) 3311–3325.
[87] M.A. Figueiredo, R.D. Nowak, S.J. Wright, Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems, IEEE J. Sel. Top. Signal Process. 1 (2007) 586–597.
[88] A. Nadkarni, D. Vesset, Worldwide Big Data Technology and Services Forecast, 2016–2020, International Data Corporation, IDC, 2016.
[89] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (2010) 2787–2805.
[90] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): a vision, architectural elements, and future directions, Futur. Gener. Comput. Syst. 29 (2013) 1645–1660.
[91] P.K. Verma, R. Verma, A. Prakash, A. Agrawal, K. Naik, R. Tripathi, M. Alsabaan, T. Khalifa, T. Abdelkader, A. Abogharaf, Machine-to-Machine (M2M) communications: a survey, J. Netw. Comput. Appl. 66 (2016) 83–105.
[92] T.H. Noor, S. Zeadally, A. Alfazi, Q.Z. Sheng, Mobile cloud computing: challenges and future research directions, J. Netw. Comput. Appl. 115 (2018) 70–85.
[93] B. Varghese, R. Buyya, Next generation cloud computing: new trends and research directions, Futur. Gener. Comput. Syst. 79 (2018) 849–861.
[94] P. Hu, S. Dhelim, H. Ning, T. Qiu, Survey on fog computing: architecture, key technologies, applications and open issues, J. Netw. Comput. Appl. 98 (2017) 27–42.
[95] C.V.N. Index, Cisco Visual Networking Index: Forecast and Methodology 2015-2020, CISCO, 2015 White paper.
[96] Y. Ding, Y. Wang, D. Zhou, Mortality prediction for ICU patients combining just-in-time learning and extreme learning machine, Neurocomputing 281 (2018) 12–19.
[97] S.H. Ling, P.P. San, H.T. Nguyen, Non-invasive hypoglycemia monitoring system using extreme learning machine for Type 1 diabetes, ISA Trans. 64 (2016) 440–446.
[98] D. Li, Q. Xie, Q. Jin, K. Hirasawa, A sequential method using multiplicative extreme learning machine for epileptic seizure detection, Neurocomputing 214 (2016) 692–707.
[99] Q. Yu, Y. Miche, E. Séverin, A. Lendasse, Bankruptcy prediction using extreme learning machine and financial expertise, Neurocomputing 128 (2014) 296–302.
[100] F.L. Chen, T.Y. Ou, Sales forecasting system based on Gray extreme learning machine with Taguchi method in retail industry, Expert Syst. Appl. 38 (2011) 1336–1345.
[101] F. Lu, J. Jiang, J. Huang, X. Qiu, Dual reduced kernel extreme learning machine for aero-engine fault diagnosis, Aerosp. Sci. Technol. 71 (2017) 742–750.
[102] Y.-P. Zhao, F.-Q. Song, Y.-T. Pan, B. Li, Retargeting extreme learning machines for classification and their applications to fault diagnosis of aircraft engine, Aerosp. Sci. Technol. 71 (2017) 603–618.
[103] N.L. da Costa, L.A.G. Llobodanin, M.D. de Lima, I.A. Castro, R. Barbosa, Geographical recognition of Syrah wines by combining feature selection with extreme learning machine, Measurement 120 (2018) 92–99.
[104] X. Dou, Y. Yang, Evapotranspiration estimation using four different machine learning approaches in different terrestrial ecosystems, Comput. Electron. Agric. 148 (2018) 95–106.
[105] C. Zhang, Q. Liu, Q. Wu, Y. Zheng, J. Zhou, Z. Tu, S.H. Chan, Modelling of solid oxide electrolyser cell using extreme learning machine, Electrochim. Acta 251 (2017) 137–144.
[106] L. Wang, X. Li, Y. Bai, Short-term wind speed prediction using an extreme learning machine model with error correction, Energy Convers. Manag. 162 (2018) 239–250.
[107] D. Laney, 3D Data Management: Controlling Data Volume, Velocity and Variety, 6, META Group Research Note, 2001.
[108] H. Chen, R.H. Chiang, V.C. Storey, Business Intelligence and Analytics: From Big Data to Big Impact, MIS quarterly, 2012, pp. 1165–1188.
[109] D.B. Data, A Practical Guide to Transforming the Business of Government, TechAmerica Foundation" s Federal Big Data Commission, 2012.
[110] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51 (2008) 107–113.
[111] R. Lämmel, Google's MapReduce programming model—Revisited, Sci. Comput. Program., 70 (2008) 1–30.
[112] K. Shim, MapReduce algorithms for big data analysis, Proc. VLDB Endow. 5 (2012) 2016–2017.
[113] T. White, Hadoop: The Definitive Guide, O'Reilly Media, Inc, 2012.
[114] D. Borthakur, The hadoop distributed file system: aArchitecture and design, Hadoop Project Website, 11 (2007) 21.
[115] S. Owen, R. Anil, T. Dunning, E. Friedman, Mahout in Action, Manning Publications Co, Greenwich, CT, USA, 2011.
[116] C.-T. Chu, S.K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, K. Olukotun, A.Y. Ng, Map-reduce for machine learning on multicore, Adv. Neural Inf. Process. Syst. (2007) 281–288.
[117] Q. He, T. Shang, F. Zhuang, Z. Shi, Parallel extreme learning machine for regression based on MapReduce, Neurocomputing 102 (2013) 52–58.
[118] X. Zhao, Z. Ma, Z. Zhang, A Novel Recommendation System in Location-Based Social Networks Using Distributed ELM, Memetic Computing, 2017.
[119] M. Duan, K. Li, X. Liao, K. Li, A parallel multiclassification algorithm for big data using an extreme learning machine, IEEE Trans. Neural Netw. Learn. Syst. 29 (6) (2018) 1–15.
[120] F.Ö. Çatak, Classification with boosting of extreme learning machine over arbitrarily partitioned data, Soft Comput. 21 (2017) 2269–2281.
[121] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[122] M. Luo, L. Zhang, J. Liu, J. Guo, Q. Zheng, Distributed extreme learning machine with alternating direction method of multiplier, Neurocomputing 261 (2017) 164–170.
[123] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends® Mach. Learn. 3 (2011) 1–122.
[124] T. Dou, X. Zhou, The fast computation methods for extreme learning machine, in: Proceedings of the Second International Conference on Intelligent Computing and Cognitive Informatics, 2015.
[125] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins Univ Press, 1996.
[126] V.N. Katsikis, D. Pappas, A. Petralias, An improved method for the computation of the Moore–Penrose inverse matrix, Appl. Math. Comput. 217 (2011) 9828–9834.
[127] I. Jeon, E.E. Papalexakis, C. Faloutsos, L. Sael, U. Kang, Mining billion-scale tensors: algorithms and discoveries, VLDB J. 25 (2016) 519–544.
[128] N.K. Nair, S. Asharaf, Tensor decomposition based approach for training extreme learning machines, Big Data Res. 10 (2017) 8–20.
[129] J. Xin, Z. Wang, L. Qu, G. Wang, Elastic extreme learning machine for big data classification, Neurocomputing 149 (2015) 464–471.
[130] P. Neuvial, Asymptotic results on adaptive false discovery rate controlling procedures based on kernel estimators, J. Mach. Learn. Res. 14 (2013) 1423–1459.
[131] X. Liu, L. Wang, G.-B. Huang, J. Zhang, J. Yin, Multiple kernel extreme learning machine, Neurocomputing 149 (2015) 253–264.
[132] C. Chen, W. Li, H. Su, K. Liu, Spectral-spatial classification of hyperspectral image based on kernel extreme learning machine, Remote Sens. 6 (2014) 5795–5814.
[133] W.-Y. Deng, Q.-H. Zheng, Z.-M. Wang, Cross-person activity recognition using reduced kernel extreme learning machine, Neural Netw. 53 (2014) 1–7.
[134] A. Iosifidis, A. Tefas, I. Pitas, On the kernel extreme learning machine classifier, Pattern Recognit. Lett. 54 (2015) 11–17.
[135] W. Li, C. Chen, H. Su, Q. Du, Local binary patterns and extreme learning machine for hyperspectral imagery classification, IEEE Trans. Geosci. Remote Sens. 53 (2015) 3681–3693.
[136] H. Fu, C.-M. Vong, P.-K. Wong, Z. Yang, Fast detection of impact location using kernel extreme learning machine, Neural Comput. Appl. 27 (2016) 121–130.
[137] W. Deng, Q. Zheng, K. Zhang, Reduced kernel extreme learning machine, in: Proceedings of the Eighth International Conference on Computer Recognition Systems CORES, Springer, 2013, pp. 63–69.
[138] H.-L. Chen, G. Wang, C. Ma, Z.-N. Cai, W.-B. Liu, S.-J. Wang, An efficient hybrid kernel extreme learning machine approach for early diagnosis of Parkinson' s disease, Neurocomputing 184 (2016) 131–144.
[139] D. Zhao, C. Huang, Y. Wei, F. Yu, M. Wang, H. Chen, An effective computational model for bankruptcy prediction using kernel extreme learning machine approach, Comput. Econom. 49 (2017) 325–341.
[140] W.-Y. Deng, Y.-S. Ong, Q.-H. Zheng, A fast reduced kernel extreme learning machine, Neural Netw. 76 (2016) 29–38.
[141] Y. Zhai, Y.-S. Ong, I.W. Tsang, The emerging" Big Dimensionality", IEEE Comput. Intell. Mag. 9 (2014) 14–26.
[142] R. Fletcher, Practical Methods of Optimization: vol. 2: constrained Optimization, John Wiley & Sons, Inc., 1981 ONE WILEY DR., SOMERSET, N. J. 08873, 1981, 224.

[143] V. Vapnik, The Nature of Statistical Learning Theory, Springer Science & Business Media, 2013.
[144] F. Luo, W. Guo, Y. Yu, G. Chen, A multi-label classification algorithm based on kernel extreme learning machine, Neurocomputing 260 (2017) 313–320.
[145] J. Tang, C. Deng, G.-B. Huang, Extreme learning machine for multilayer perceptron, IEEE Trans. Neural Netw. Learn. Syst. 27 (2016) 809–821.
[146] L.L.C. Kasun, H. Zhou, G.-B. Huang, C.M. Vong, Representational learning with extreme learning machine for big data, IEEE Intell. Syst. 28 (2013) 31–34.
[147] J. Xin, Z. Wang, C. Chen, L. Ding, G. Wang, Y. Zhao, ELM∗: distributed extreme learning machine with MapReduce, World Wide Web 17 (2014) 1189–1204.
[148] B. Wang, S. Huang, J. Qiu, Y. Liu, G. Wang, Parallel online sequential extreme learning machine based on MapReduce, Neurocomputing 149 (2015) 224–232.
[149] J. Pang, Y. Gu, J. Xu, X. Kong, G. Yu, Parallel multi-graph classification using extreme learning machine and MapReduce, Neurocomputing 261 (2017) 171–183.
[150] Y. Ming, E. Zhu, M. Wang, Y. Ye, X. Liu, J. Yin, DMP-ELMs: data and model parallel extreme learning machines for large-scale learning tasks, Neurocomputing 320 (2018) 85–97.
[151] S. Huang, B. Wang, J. Qiu, J. Yao, G. Wang, G. Yu, Parallel ensemble of online sequential extreme learning machine based on MapReduce, Neurocomputing 174 (2016) 352–367.
[152] Z. Wang, Y. Zhao, Y. Yuan, G. Wang, L. Chen, Extreme learning machine for large-scale graph classification based on MapReduce, Neurocomputing 261 (2017) 106–114.
[153] J. Peddie, The new visualization engine–the heterogeneous processor unit, Expanding the Frontiers of Visual Analytics and Visualization, Springer, 2012, pp. 377–395.
[154] B. Krawczyk, GPU-Accelerated Extreme Learning Machines for Imbalanced Data Streams with Concept Drift, ICCS, 2016, pp. 1692–1701.
[155] M. Alia-Martinez, J. Antoñanzas, F. Antonanzas-Torres, A. Pernía-Espinoza, R. Urraca, A straightforward implementation of a GPU-accelerated ELM in R with NVIDIA graphic cards, in: Proceedings of the International Conference on Hybrid Artificial Intelligence Systems, Springer, 2015, pp. 656–667.
[156] A.S. Garea, D.B. Heras, F. Argüello, GPU classification of remote-sensing images using kernel ELM and extended morphological profiles, Int. J. Remote Sens. 37 (2016) 5918–5935.
[157] J. López-Fandiño, P. Quesada-Barriuso, D.B. Heras, F. Argüello, Efficient ELM-based techniques for the classification of hyperspectral remote sensing images on commodity GPUs, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (2015) 2884–2893.
[158] H.-N. Tran, E. Cambria, Ensemble application of ELM and GPU for real-time multimodal sentiment analysis, Memet. Comput. 10 (2018) 3–13.
[159] C. Chen, K. Li, A. Ouyang, Z. Tang, K. Li, GPU-accelerated parallel hierarchical extreme learning machine on flink for big data, IEEE Trans. Syst. Man Cybern. Syst. 47 (2017) 2740–2753.
[160] M. Van Heeswijk, Y. Miche, E. Oja, A. Lendasse, GPU-accelerated and parallelized ELM ensembles for large-scale regression, Neurocomputing 74 (2011) 2430–2437.
[161] N. Sundararajan, Y. Lu, P. Saratchandran, A sequential learning scheme for function approximation by using minimal radial basis function networks, Neural Comput. 9 (1997) 461–478.
[162] J. Platt, A resource-allocating network for function interpolation, Neural Comput. 3 (1991) 213–225.
[163] S. Suresh, K. Dong, H. Kim, A sequential learning algorithm for self-adaptive resource allocation network classifier, Neurocomputing 73 (2010) 3012–3019.
[164] Y. Sun, Y. Yuan, G. Wang, An OS-ELM based distributed ensemble classification framework in P2P networks, Neurocomputing 74 (2011) 2438–2443.
[165] X. Luo, X. Yang, C. Jiang, X. Ban, Timeliness online regularized extreme learning machine, Int. J. Mach. Learn. Cybern. 9 (2018) 465–476.
[166] G.-B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2011) 107–122.
[167] J. Xiang, M. Westerlund, D. Sovilj, G. Pulkkis, Using extreme learning machine for intrusion detection in a big data environment, in: Proceedings of the Workshop on Artificial Intelligent and Security Workshop, ACM, 2014, pp. 73–82.
[168] J. Chen, H. Chen, X. Wan, G. Zheng, MR-ELM: a MapReduce-based framework for large-scale ELM training in big data era, Neural Comput. Appl. 27 (2016) 101–110.
[169] S. Ding, B. Mirza, Z. Lin, J. Cao, X. Lai, T.V. Nguyen, J. Sepulveda, Kernel based online learning for imbalance multiclass classification, Neurocomputing 277 (2018) 139–148.
[170] B. Mirza, Z. Lin, Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification, Neural Netw. 80 (2016) 79–94.

Peter Adeniyi Alaba does research in Artificial Intelligence (AI), Renewable Energy and Advanced Environmental engineering with a focus on AI in chemical engineering, renewable and sustainable energy, wastewater treatment, solid waste treatment. He is a graduate of Chemical Engineering from the Federal University of Technology, Minna, building up with an M.Eng.Sc. in Catalysis and Reaction Engineering from University of Malaya, Malaysia. He is a Phd candidate of University of Malaya. Peter is an author and has worked in collaboration with several scientists from Australia, Malaysia, Mexico, Egypt, Iraq, Pakistan, Canada, United Kingdom and Nigeria on projects such as AI in chemical engineering, Catalyst Development for Efficient Biodiesel Production, and Carbon Dioxide Reduction, Electrocatalytic reactions, Animal Nutrition, Enzymes, Nanotechnology, Pipeline Corrosion and Wastewater Treatment. He also renders professional services to academic journals such as American Chemical Society (ACS) journals, Chemical Papers, Separation and Purification Technology, Industrial and Engineering Chemistry, Applied Water Science, Journal of Cleaner Production and African Journal of Biotechnology. He is currently a grant peer-review international expert for National Center of Science and Technology Evaluation, JSC, Almaty, Kazakhstan. He is a corporate member of COREN (Council for the Regulation of Engineering in Nigeria).