Received October 19, 2020, accepted October 25, 2020, date of publication November 3, 2020, date of current version November 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035398

A Survey of Distributed and Parallel Extreme Learning Machine for Big Data

ZHIQIONG WANG[1,4,5], (Member, IEEE), LING SUI[2], JUNCHANG XIN[2], LUXUAN QU[1], AND YUDONG YAO[3], (Fellow, IEEE)

[1]College of Medicine and Biological Information Engineering, Northeastern University, Shenyang 110819, China
[2]Key Laboratory of Big Data Management and Analytics (Liaoning), School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
[3]Department of Eletrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA
[4]Neusoft Research of Intelligent Healthcare Technology, Company Ltd., Shenyang 110179, China
[5]Acousitics Science and Technology Laboratory, Harbin Engineering University, Harbin 0086, China

Corresponding author: Junchang Xin (xinjunchang@mail.neu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62072089, in part by the China Postdoctoral Science Foundation under Grant 2019T120216 and Grant 2018M641705, and in part by the Fundamental Research Funds for the Central Universities under Grant N2019007, Grant N180101028, Grant N180408019, and Grant N2024005-2.

ABSTRACT Extreme learning machine (ELM) is characterized by good generalization performance, fast training speed and less human intervention. With the explosion of large amount of data generated on the Internet, the learning algorithm in the single-machine environment cannot meet the huge memory consumption of matrix computing, so the implement of distributed ELM algorithm has gradually become one of the research focuses. In view of the research significance and implementation value of distributed ELM, this paper first introduced the research background of ELM and improved ELM. Secondly, this paper elaborated the implementation method of distributed ELM from the two directions: ensemble and matrix operation. Finally, we summarized the development status of distributed ELM and discussed the future research direction.

INDEX TERMS Extreme learning machine, distributed processing, ensemble, matrix operation.

I. INTRODUCTION

With the rapid development of Internet and Internet of things, the amount of data has been increasing rapidly in recent years. We have entered the big data era. The volume, velocity, variety and value are the main features of big data [1]. With the continuous development of big data, the complexity of data is getting higher and higher. In 2017, IBM has added veracity to the 4V big data feature, to emphasize that meaningful data must be true and accurate. After that, the features of big data have been gradually added vitality, emphasizing the vitality of the whole data system; visiualization, emphasizing the explicit display of data; validity, emphasizing the validity of data collection and application. The knowledge hidden in big data is valuable for decision-making in all fields. Machine learning has become one of the hot methods of knowledge discovery and a research hotspot in the field of big data. How to conduct data mining and machine learning on large amounts of data has become an important issue in the era of big data [2]–[4]. With the increase of the number of hidden layers, the traditional training method of neural networks will have many problems such as slow convergence, time consumption and so on. In order to avoid the above problems, neural networks with random weights are proposed in which the weights between the hidden layer and input layer are randomly selected and the weights between the output layer and the hidden layer are obtained analytically [5]–[7]. Extreme learning machine (ELM) is one kind of neural networks with random weights to train single-hidden layer feedforward networks (SLFNs). Compared with traditional SLFNs, ELM has the following remarkable characteristics: First, ELM is characterized by its fast learning speed. Second, traditional SLFNs may face problems such as local minima, inappropriate learning rate and overfitting, it needs to use some optimization methods to avoid these problems. ELM is simpler than traditional SLFNs, which does not involve these problems. For example, Radial Basis Function (RBF) neural network investigated the implicit assumptions made

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

when employing a feed-forward layered network model to analyze complex data [8], and ELM can be extended to RBF network case, which allows the centers and impact widths of RBF kernels to be randomly generated and the output weights to be simply analytically calculated instead of iteratively tuned. Finally, compared with the back propagation algorithm in SLFNs, ELM has better generalization performance [9]. In the original ELM, the nodes in the hidden layer are generated randomly, which is not related to the training data. However, recent research have shown that setting the number of hidden layer nodes arbitrarily may lead to underfitting or overfitting of ELM. The statistical characteristics of data sets and the way of generating random parameters have a significant impact on the performance of ELM [10]–[12]. Recently, there have been many research of ELM based on parameter optimization, such as ELM based on Principal Component Analysis ELM(PCA-ELM) [13], ELM based on Particle Swarm Optimization (PSO-ELM) [14] and ELM based on Genetic Algorithm (GA-ELM) [15]. Compared with the traditional neural network learning methods, ELM has significant learning efficiency. Therefore, ELM provides good generalization performance [16] with its extremely fast learning speed and has been widely used in the fields of text classification [17], image recognition [18], sensory recognition(visual, taste, and smell) [19]–[21], Industrial [22], [23] and bioinformatics [24], [25].

Although in recent years, many variants of ELM have been developed, such as Multilayer Probability ELM (MP-ELM) [26], ELM combined with sparse representation classification (ELM-SRC) [27], Residual compensation ELM(RC-ELM) [28], to improve the efficiency of ELM. But the ELM algorithm is a memory-resident algorithm. That is to say, all pre-processed data must be loaded into computer memory in advance. With the continuous expansion of the training data scale, the memory limitation of traditional serial or single computer environment, and the huge computation amount of the ELM matrix, the traditional ELM cannot play its efficient role. Therefore, in the face of large-scale data sets, the application of parallel algorithm and the the distributed procession of ELM will become one of the key points in the future research.

ELM has been developed for nearly 15 years, a lot of research have been carried out on it domestic and overseas, and remarkable achievements have been made. Huang *et al.* [16] mainly introduced the variants of ELM, such as batch learning mode of ELM, fully complex ELM, online sequential ELM and incremental ELM. And they proposed that ELM should be studied in the future such as digging hidden nodes and evaluating the generalization performance of ELM. Ding *et al.* [29] introduced the research status, such as algorithm, theory and application, including the model and the concrete application of ELM. And they pointed out the future research direction, such as, further improve the generalization performance of ELM model structure and algorithm, combined the online learning and genetic algorithm with ELM, how to make better use of ELM to deal with all kinds of round and multiple classification problems, etc. Huang *et al.* [30] summarized the ELM from interpolation theory, the universal approximation ability and generalization ability, introduced the ELM in biomedical engineering, computer vision, system identification and so on, finally outlook the ELM in the future. Cao *et al.* [31] summarized the research progress of ELM in recent years and its applications in big data processing such as graphics processing, video processing and medical signal processing. They concluded that due to the randomness of the network parameters and the untuned learning strategy, the computational complexity is greatly reduced, which benefits the application of ELM and its variants in intelligent high-dimensional big data processing. Three ELM problems in high-dimensional big data processing are proposed. 1) How to balance the performance and processing time? 2) How to select the optimal number of hidden neurons for a specific application? 3) Most of the research results are realized by computer simulation in the laboratory, real-world devices for different applications are always facing various challenges. In recent years, there have been many studies on ELM, such as discriminative ELM [32], cross-domain ELM [33], evolutionary cost-sensitive ELM [34], and adaptive ELM [35] to solve different problems.

The above research introduced ELM from three aspects of the ELM's variants, optimization methods and its applications in different fields. But there isn't a detailed elaboration and comparison on the distributed method of ELM. In view of the theoretical research and practical significance of distributed ELM, this paper summarized the algorithm on the ELM distributed platform for big data, and divided distributed ELM algorithms into ensemble method and matrix decomposition method according to its implementation modes. Section 2 summarizes the research status of ELM, OS-ELM and ELM with kernel. Section 3 describes two distributed ELM implementation methods: ensemble and matrix operation. Section 4 analyzes the future research trend of distributed ELM and makes a summary.

II. BACKGROUND

In this section, we will describe our background, including ELM, OS-ELM and ELM with kernel.

A. ELM

The distributed ELM researcher mainly focuses on ELM and the online sequential ELM (OS-ELM). Therefore this section will mainly introduces the research status of classical ELM and OS-ELM.

1) CLASSIC ELM

ELM [9], [16], [36] was initially proposed for the single hidden layer feedforward neural networks (SLFNs) [5], [6], and then extended to the generalized SLFNs, where the hidden layer does not need the same neurons. ELM first randomly assigns input weights and hidden layer bias, and then determines the output weights of SLFNs. ELM has great advantage in efficiency and generalization performance compared with the classical neural network algorithm which applied to a wide rage of problems in different fields.

For any N different samples (x_j, t_j), where $x_j = [x_{j1}, x_{j2}, \cdots, x_{jn}]^T \in R^n$ and $t_j = [t_{j1}, t_{j2}, \cdots, t_{jm}]^T \in R^m$. The standard SLFNs mathematical model with L hidden nodes and activation function $g(x)$ is modeled as

$$\sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = o_j \tag{1}$$

where $w_i = [w_{i1}, w_{i2}, \cdots, w_{in}]^T$ is the weight vector connecting the *ith* hidden node with the input node; $\beta_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^T$ is the weight vector connecting the *ith* hidden node with the output node; b_i is the threshold of *ith* hidden node; $o_j = [o_{j1}, o_{j2}, \cdots, o_{jm}]^T$ is the *jth* output vector of SLFNs.

Standard SLFNs with L hidden nodes and activation function $g(x)$ can approximate N samples with zero error. It means that $\sum_{j=1}^{L} ||o_j - t_j|| = 0$ and according to Equation 1, the Equation 2 as follows:

$$\sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = o_j = t_j \quad (j = 1, 2, \cdots, N) \tag{2}$$

The above equation can be succinctly expressed as:

$$H\beta = T \tag{3}$$

where H is:

$$H(w_1, w_2, \cdots, w_L, b_1, b_2, \cdots, b_L, x_1, x_2, \cdots, x_L) = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_L \cdot x_1 + b_L) \\ g(w_1 \cdot x_2 + b_1) & \cdots & g(w_L \cdot x_2 + b_L) \\ \vdots & \vdots & \vdots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_L \cdot x_N + b_L) \end{bmatrix}_{N \times L} \tag{4}$$

$$\beta = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{L1} & \beta_{L2} & \cdots & \beta_{Lm} \end{bmatrix}_{L \times m}$$

$$T = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ t_{N1} & t_{N2} & \cdots & t_{Nm} \end{bmatrix}_{N \times m} \tag{5}$$

H is called the hidden layer output matrix of the neural network and the *ith* column of H is the *ith* hidden node output with respect to inputs $x_1, x_2, \cdots, x_N$. The smallest norm least-squares solution of the above linear system is:

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

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H. The the output function of ELM can be modeled as follows:

$$f(x) = h(x)\widehat{\beta} = h(x)H^{\dagger}T \tag{7}$$

where $h(x)$ is feature mapping function.

2) OS-ELM

Liang *et al.* [37] developed an online sequential learning algorithm for single hidden layer feedforward networks (SLFNs) with additive or radial basis function (RBF) hidden nodes in a unified framework. This algorithm is called online sequential ELM (OS-ELM). OS-ELM can fix or change the block size of the data block one-by-one or chunk-by-chunk learning.

OS-ELM consists of two phases, namely the initialization phase and the sequential learning phase. In the initialization phase, input the activation function $g(x)$, and the appropriate matrix H_0 is filled up for use in the learning phase. The number of data required to fill up should be at least equal to the number of hidden nodes. Following the initialization phase, the learning phase commences either on a one-by-one or chunk-by-chunk (with fixed or varying size) basis as desired. Once the data is used, it is discarded and not used any more. Finally, we get the output weight β.

Initialization Phase: Initialize the learning using a small chunk of initial training data $\mathbb{N}_0 = \{(x_i, t_i)\}_{i=1}^{N_0}$ from the given training set.

a. Assign random input weights a_i and bias b_i (for additive hidden nodes)or center a_i and impact factor b_i (for RBF hidden nodes), $i = 1, \ldots, L$.

b. Calculate the initial hidden layer output matrix H_0

$$H_0 = \begin{bmatrix} g(a_1, b_1, x_1) & \cdots & g(a_L, b_L, x_1) \\ \vdots & \cdots & \vdots \\ g(a_1, b_1, x_{N_0}) & \cdots & g(a_L, b_L, x_{N_0}) \end{bmatrix}_{N_0 \times L} \tag{8}$$

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

d. Set $k = 0$.

Sequential Learning Phase: Present the $k + 1$th chunk of new observations

$$\mathbb{N}_{k+1} = \{(x_i, t_i)\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j} \tag{9}$$

where N_{k+1} denotes the number of observations in the $(k + 1)$th chunk.

a. Calculate the partial hidden layer output matrix H_{k+1} for the $k + 1$th chunk of data N_{k+1} [1]:

$$H_{k+1} = \begin{bmatrix} g(a_1, b_1, x_{\varphi(k)+1}) & \cdots & g(a_L, b_L, x_{\varphi(k)+1}) \\ \vdots & \cdots & \vdots \\ g(a_1, b_1, x_{\varphi(k+1)}) & \cdots & g(a_L, b_L, x_{\varphi(k+1)}) \end{bmatrix}_{N_{k+1} \times L} \tag{10}$$

b. Set $T_{k+1} = [t_{(\sum_{j=0}^{k} N_j)+1}, \ldots, t_{\sum_{j=0}^{k+1} N_j}]^T$.

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

$$P_{k+1} = P_k - P_k H_{k+1}^T (I + H_{k+1} P_k H_{k+1}^T)^{-1} H_{k+1} P_k \tag{11}$$

$$\beta^{(k+1)} = \beta^{(k)} + P_{k+1} H_{k+1}^T (T_{k+1} - H_{k+1} \beta^{(k)}) \tag{12}$$

d. Set $k = k + 1$. Go to Sequential Learning Phase.

[1]To simplify the matrix, we set $\varphi(k) = \sum_{j=0}^{k} N_j$

B. KERNEL ELM

Huang *et al.* [38] combined the learning principle of support vector machine, introduced the kernel function into ELM, and proposed the kernel ELM. Compared with the Extreme SVMs proposed by Liu *et al.* [39] and Benoit *et al.* [40], the ELM constructed with this method has fewer constraints and better learning ability. Kernel ELM does not consider the feature mapping function $h(x)$, the input weight vector w, the bias b and the number of hidden nodes L in ELM. When the mapping is unknown, it will construct a kernel function to represent HH^T:

$$HH^T = \Omega_{ELM} = \begin{bmatrix} K(x_1,x_1) & \cdots & K(x_1,x_N) \\ \vdots & \ddots & \vdots \\ K(x_N,x_1) & \cdots & K(x_N,x_N) \end{bmatrix}$$
$$h(x)H^T = \begin{bmatrix} K(x,x_1) \\ \vdots \\ K(x,x_N) \end{bmatrix} \tag{13}$$

So the connection weight matrix β between hidden layer and output layer can be expressed as:

$$\beta = (\frac{I}{C} + \Omega_{ELM})^{-1} T \tag{14}$$

where C is regularized parameter. And where

$$T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m}) \\ \vdots & \vdots & \vdots \\ t_{N1} & \cdots & t_{Nm}) \end{bmatrix} \tag{15}$$

where the expected output vector of the m output nodes is $t_i = [0, \cdots, 0, 1, 0, \cdots, 0]^T$

The classification formula of kernel ELM expressed as

$$f(x)=g(h(x)H^T\beta)=g(\begin{bmatrix} K(x,x_1) \\ \vdots \\ K(x,x_N) \end{bmatrix}(\frac{I}{C}+\Omega_{ELM})^{-1}T) \tag{16}$$

III. DISTRIBUTED ELM

The ensemble method consists of a group of separately trained classifiers, and an ensemble is usually more accurate than any classifier in the ensemble [41]. Ensemble multiple ELM into one model can achieve the parallelization of ELM and speed up the computation. Although ensemble ELM solves the computational consumption problem of multiple learning on the same data block, the ensemble learning cannot learn all the data. In the ELM computation, the most expensive computation part is the matrix multiplication operator of the Moore-Penrose generalized inverse matrix [42]. The matrix multiplication operator is decomposable. Therefore, matrix operation is proposed to solve the problem that ensemble learning cannot learn all the data. Existing researchers mainly apply ensemble and ELM matrix operation optimization to conduct distributed processing on ELM.

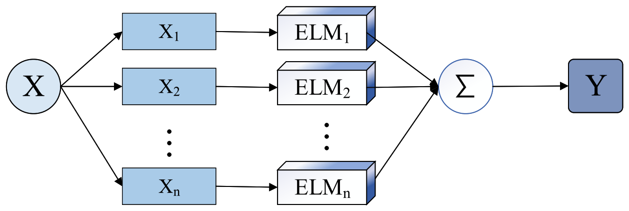

FIGURE 1. Implementation process of results ensemble ELM.

A. ELM ENSEMBLE

There are two main ways to parallelize ELM using the ensemble methods: (1) Results ensemble. Decompose the problem (data set) into sub-problem (sub-data set), train an ELM for each sub-problem (sub-data set), and finally gather the trained results. (2) Parameters ensemble. Divide ELM into multiple sub-ELMs through different partitioning methods, train the ELM sub-models in parallel, and finally combine all the trained sub-models through some algorithms. Results ensemble gather the trained results of multiple models, the trained results have higher accuracy. Parameters ensemble gather the parameters on a single model to calculate, which is more complicated than the calculation on a single model of results ensemble. However, the results ensemble are calculated on multiple models,the calculation complexity is similar. At the same time, parameters ensemble has better performance in memory optimization.

1) RESULTS ENSEMBLE

As shown in Fig. 1, the results ensemble is the aggregation of the results after training multiple ELM. Results ensemble are mainly used to train models in different environments such as GPU [43], network [44], [45], and MapReduce [46]–[48] framework. And the results ensemble method is applied to kernel ELM [49].

In 2011, Heeswijk *et al.* [43] proposed to combine multiple ELM into the inheritance model. They paralleized model training and model structure process among multiple GPU and CPU cores, realized the simultaneous construction of ELM multiple models, paralleized and improved the learning speed of ELM. To address the problem of classification in P2P networks, Sun *et al.* [44] applied OS-ELM to the P2P network, trained each peer node and generated an ensemble classifier. This method not only improved the computation speed, but also solved the problem of low classification accuracy in traditional P2P ensemble classifier, where the local classifier only learns part data. Wang *et al.* [45] applied M^3-network [50] into the ELM ensemble, and proposed a parallel ensemble ELM (M^3-ELM) based on M^3-network. M^3-ELM first decomposes the classification problem into smaller sub-problems, then trains one ELM for each sub-problem, and finally ensemble these ELM with M^3-network. Compared with common ELM, M^3-ELM increased the training speed by 1.6-4.6 times and reduces the training error by 0.37-19.51%.

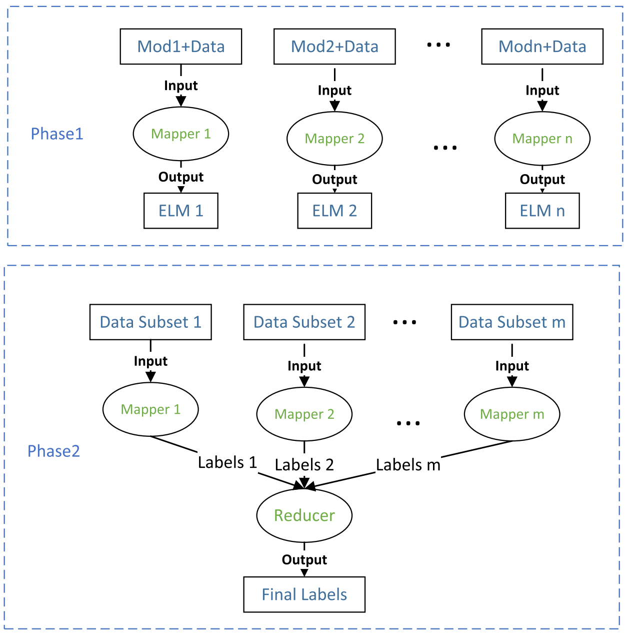

FIGURE 2. **Framework of ELM-MapReduce.**

MapReduce is a computing framework for big data parallel processing, and the ELM ensemble based on the MapReduce framework is widely used. ELM-MapReduce [46] adopts the ELM learning method to build an ELM ensemble classifier on the MapReduce, as shown in figure Fig. 2. ELM is widely used for gesture recognition.

However, when the data set includes multiple objects, the classical ELM may produce large errors. Liang *et al.* [47] built a separate ELM network for each gesture and combined the results after training, instead of building one ELM for all gestures. Noise is frequent in large-scale data, and in order to eliminate the impact of noise, Huang *et al.* [48] first proposed an ensemble OS-ELM framework, which integrates three ensemble methods: bagging, subspace partitioning, and cross validation. Then, they designed a parallel ensemble algorithm of OS-ELM based on the MapReduce to analyze large scale data effectively.

In order to build an ensemble classifier of kernel ELM, Li *et al.* [49] proposed a parallel one-class ELM algorithm (P-ELM) based on bayesian method. P-ELM divides the training data set into k components according to class attributes, and then uses the divided training data set to train k kernel-based one-class ELM classifiers, finally, uses the bayesian method to compare the output function values of one-class P-ELM classifier based on kernel.

2) PARAMETERS ENSEMBLE

The research of parameters ensemble is mainly to train a ELM sub-model on each node through the MapReduce framework, and then collect all the sub-models to form the final model through some algorithms. It is used to solve problems such as integrating classification and regression [51], inefficiency and lack of memory [52], [53], and improving the scalability of data processing [54].

Chen *et al.* [51] proposed MR-ELM to solve both classification and regression problems. MR-ELM trains an ELM sub-model in each Hadoop node, uses local sample blocks, collects trained hidden nodes, and forms the final ELM model. For the regression problem, they used least squares method to calculate the weight of each group of hidden nodes. For classification problems, simply merged the hidden node groups. Different from MR-ELM, the model proposed by Wu *et al.* [52] adopts the generalized inverse method to calculate the weight of each Hadoop node, and the combined weight is obtained by the Moore-Penrose generalized inverse operator to combine all ELM sub-models. Catak [53] constructed AdaBoosted-ELM classifier based on the combination of ELM and AdaBoosting to improve the classifiers prediction ability. Adaboosted-ELM creates data blocks from the training data set by using MapReduce paradigm, and uses each subset of the training data set as a single global classifier function to find the ELM ensemble. Budiman *et al.* [54] integrated CNN architecture with ELM, CNN serves as unsupervised convolution feature learner and ELM as supervised classifier to improve the scalability of big data processing. In the calculation process of MapReduce, the Map process acts as the classifier of CNN-ELM and conducts independent learning on different training data partitions. Reduce process integrates the all weights (kernel weight on CNN and output weight on ELM) on CNN-ELM on an average basis.

B. ELM MATRIX OPERATION OPTIMIZATION

Because the matrix multiplication operator of the Moore-Penrose generalized inverse matrix is the most expensive part, it is very difficult to calculate on a single machine, and the matrix operator is decomposed. Therefore, most research focused on the decomposition of matrix multiplication operation [42], [55]–[57], and proposed the double classifier algorithm that combines the most basic matrix decomposition method with other methods [58], [59]. Subsequent studies focused on some defects of the basic ELM matrix decomposition, and solved the problems of the deficiency of MapReduce framework [60]–[62], only considering supervised ELM [63], [64], and unbalanced data [65], [66]. At the same time, the distributed processing of improved ELM (OS-ELM [67]–[70] and kernel ELM [71]–[73]) is considered.

1) CLASSIC ELM

He *et al.* [55] first proposed the parallel ELM algorithm (PELM) based on the MapReduce framework and designed the key value pairs suitable for ELM algorithm. In PELM, after the first MapReduce job obtained the output matrix H, H is written to HDFS; The second MapReduce job reads the output matrix H from the HDFS to calculate the final ELM result. Therefore, in the two MapReduce jobs, a large number of intermediate results are converted,

Algorithm 1 ELM* Training Algorithm

```
1:  function MAP(sid $id$, sample $s$)
2:      $h$ = new ASSOCIATIVEARRAY
3:      $(x, t)$ = Parse($s$)
4:      for $i = 1 \to L$ do
5:          $h[i] = g(w_i \cdot x + b_i)$
6:      end for
7:      for $i = 1 \to L$ do
8:          for $j = 1 \to L$ do
9:              context.write(triple($'U'$, $i, j$), $h[i]h[j]$)
10:             for $j = 1 \to m$ do
11:                 context.write(triple($'V'$, $i, j$), $h[i]t[j]$)
12:             end for
13:         end for
14:     end for
15: end function
16: function REDUCE(triple $p$, sum $[s_1, s_2, \cdots]$)
17:     temporary variable $\omega = 0$
18:     for all sum $s \in [s_1, s_2, \cdots]$ do
19:         $\omega = \omega + s$
20:         context.write(triple $p$, sum $\omega$)
21:     end for
22: end function
```

increasing the processing time for ELM. Xin *et al.* [42] found that the matrix U and V can share the calculation of h_{ij} of the matrix H, and the calculation of partial sum of u_{ij} and v_{ij} is independent. Therefore, they proposed ELM* to calculate the matrix U and V in one MapReduce process as shown in Algorithm 1. ELM* combines two MapReduce jobs of PELM, and only uses one MapReduce to obtain the final ELM result. ELM* not only reduces the transmission cost of a large number of intermediate results, but also improves the processing efficiency. However, ELM* has a weak learning ability, in order to make up for the deficiency of ELM* in updating the number of hidden nodes, Xin *et al.* [56] proposed an adaptive distributed ELM (A-ELM*). A-ELM* first computes the intermediate matrix multiplication of the updated hidden subset of nodes, then updates the matrix multiplication by modifying old matrix multiplication and intermediate matrix multiplication, and finally uses update matrix multiplication to obtain a new output weight vector. In order to make up for ELM*'s lack of updated large-scale data sets, Xin *et al.* [57] proposed an elastic ELM based on the MapReduce, named E^2LM. E^2LM first calculates the intermediate matrix multiplication of the updated training subset, and then uses the same method as A-ELM* to update the matrix multiplication and obtain the new output weight vector. Since the number of update hidden nodes is much smaller than the whole update part and the updated training data set is smaller than the whole training data set, the calculation time of A-ELM* and E^2LM is much smaller than ELM*. The matrix decomposition method of ELM* shows good performance in improving the performance of centralized recommendation algorithm in large-scale recommendation [74] and in WiFi-based fingerprint indoor positioning system [75].

A large number of data sets not only have a large number of records, but also bring the problem of the feature space dimension, so it is always necessary to reduce the dimension of feature space. Nonlinear principal component analysis (NLPCA) is used as a dimension reduction method, which takes into account the nonlinear relationship between features. Tejasviram *et al.* [58] proposed that Auto Associative ELM (AAELM) perform NLPCA, extract the output of AAELM hidden node, and treat it as NLPCs after the training. And implement AAELM by matrix decomposition on the MapReduce. Decision Trees(DT) [76] is a promising parallel classification algorithm with the advantages of simple implementation, fewer parameters and less computation. However, many parallel DT algorithms ignore the over-segmentation problem, which may lead to redundancy and over-fitting. To solve this problem, Wang *et al.* [59] proposed a hybrid DT induction method – ELM-Tree. When all available segmentation gain ratios are less than the threshold, ELM is embedded as a leaf node. Since the calculation of information gain and gain ratio of different cutting points are independent, it can be completed in parallel. Considering the parallel calculation of ELM output matrix, the parallel calculation is applied to ELM-Tree.

Although distributed processing based on the MapReduce has been widely used, many Map and Reduce tasks are generated. Intermediate results generated in the Map phase will be written to disk;In the Reduce phase these intermediate results will be read from the disk to the Hadoop distributed file system (HDFS). This process greatly increases the communication cost and reduces the learning speed and efficiency. In contrast to Hadoop, Spark operations are based on Resilient Distributed Datasets (RDD), which can be cached in memory across nodes and reused in multiple parallel operations similar to MapReduce. Therefore, multiple occurrences of variables and intermediate variables can be cached in memory rather than on disk, reducing communication costs and I/O overhead. Oneto *et al.* [60], [61] realized emotion recognition and polarity detection on ELM with Spark memory technology, and solved the problem of selecting the super-parameter of ELM with the best generalization performance. Duan *et al.* [62] proposed ELM (SELM) based on Spark framework. By partitioning the corresponding data reasonably to maintain balance among node workload, the hidden layer output matrix calculation algorithm, matrix $\hat{U}$ decomposition algorithm, and matrix V decomposition algorithm perform most of the computations locally, SELM realizes localization of most calculations, while keeping the intermediate results in distributed memory and caching diagonal matrix as broadcast variables, thus reducing a large amount of costs. Compared with PELM [55], ELM* [42], and improved ELM* [56], [57], SELM achieves the highest acceleration speed on the premise that the accuracy is the same as that of traditional ELM.

2) IMPROVED ELM

Currently, distributed ELM only supports supervised learning on labeled training data sets, and does not support the processing of partially labeled or unlabeled training data. Considering parallelization of semi-supervised ELM (SS-ELM), Chen [63] proposed parallel approximation SS-ELM (PASS-ELM). PASS-ELM is based on the approximate adjacent similarity matrix (AASM) algorithm, uses the Locality-Sensitive Hashing (LSH) algorithm to calculate the approximate neighborhood similarity matrix, and adopts the Laplace acceleration method for distributed processing of ELM. Different from the emphasis on PASS-ELM, U-ELM proposed by Wang *et al.* [64] adopts matrix decomposition method to parallelize ELM and the Laplace acceleration method adopted by PASS-ELM to form a complementary, and not only extends distributed ELM to semi-supervised learning, but also to unsupervised learning.

ELM and its variants have been widely used in many big data learning applications, where it is easy to find the raw data with imbalanced hierarchical distribution [77], [78]. Zong *et al.* [79] proposed a weighted ELM (W-ELM) to deal with the imbalance problem. Different from the traditional ELM which treats all training data equally, W-ELM adds different penalty coefficients to weight the training errors of different inputs. Wang *et al.* [65] proposed the distributed processing of W-ELM (DW-ELM), which improved the efficiency of learning a large number of unbalanced data. DW-ELM first uses two MapReduce jobs to effectively calculate matrix multiplication in parallel, and then obtains the corresponding output weight vector through centralized calculation. The experiment shows that, no matter how the experimental parameters change, DW-ELM can always process large-scale data effectively and quickly. After that, Wang *et al.* [66] proposed an improved DW-ELM (IDW-ELM). DW-ELM uses two MapReduce jobs to complete the calculation of U and V matrix, while IDW-ELM only uses one MapReduce job to finish the same calculation, so the transmission time of IDW-ELM is far less than that of DW-ELM.

A traditional ELM assumes that all training data is prepared prior to the training process, however, in some tasks, the training data is sequential. In order to extend ELM to online sequential data, Liang *et al.* [37] proposed the online sequential ELM (OS-ELM), which can learn data one by one or block by block, and can change block size to process block-to-block data. Ai *et al.* [67] proposed a distributed collaborative ELM based on message exchange between adjacent nodes, named DC-ELM. DC-ELM restates the centralized ELM training problem into a separable form between nodes with uniform constraints, and then uses distributed optimization tools to solve the equivalence problem. Although DC-ELM does not parallelize OS-ELM, it uses the online sequential method to conduct distributed processing on ELM. Wang *et al.* [68] proposed a parallel OS-ELM (POS-ELM) based on the MapReduce by analyzing the dependency of OS-ELM matrix calculation. The effective of POS-ELM is equal to that of OS-ELM and ELM, and in large-scale learning, the efficiency of POS-ELM is better. POS-ELM supports training of a single OS-ELM model in parallel, but does not support training multiple OS-ELM models effectively. Therefore, in order to train multiple models accurately and effectively, Huang *et al.* [69], [70] proposed batch parallel OS-ELM (BPOS-ELM), estimated Map and Reduce execution time with historical statistical data, and generated execution plan. BPOS-ELM started a MapReduce to train multiple OS-ELM models according to the generated execution plan.

ELM provides a unified learning program and a widely used type of functional mapping. In these uniform algorithms, kernel ELM uses a kernel rather than a random feature map. However, with the exponential growth of training data in large-scale learning applications, centralized kernel ELM has a large matrix computing memory consumption problem, so it is very important to conduct distributed processing on kernel ELM. Bi *et al.* [71] realized parallelization of kernel ELM on the MapReduce, and realized matrix decomposition on the MapReduce by using orthogonal projection method. Karthick *et al.* [72] adopted Spark ITFS technology to select features through dimensionality reduction, and then classify each node to parallelize the kernel ELM. Pandeeswari *et al.* [73] first proposed the kernel OS-ELM based on the MapReduce, and proposed the online sequential ELM method with kernel (OS-ELM-Ker) based on sparse criterion, and simultaneously considered the parallelization of OS-ELM and kernel ELM.

There are also distributed approaches that take into account both ensemble and matrix operations. For example, Han *et al.* [80] proposed a weighted ensemble ELM (WE-DELM) based on matrix operation, combining matrix decomposition with ensemble operation; Wang *et al.* [81] proposed two models, data parallel regularization ELM (DPR-ELM) and model parallel regularization model (MPR-ELM), respectively using matrix operation and ensemble.

C. OTHERS

In addition to ensemble and matrix operations, there are other ways to implement distributed ELM. For example, iteration acceleration [82], using acceleration package on MATLAB [83], GPU acceleration [84], [85], using online sequential to realize distributed [86], and so on.

Different from He *et al.* [55], Xin *et al.* [42]. Kokkinos and Margaritis [82] *et al.* conducted the incremental version of ELM. Incremental ELM does not use direct matrix-matrix multiplicators, instead of adding neurons one by one, using each neuron to transmit one data, for direct parallelization. Compared with the classical training method of calculating the generalized inverse of regression matrix to solve the output weight, incremental ELM has a lower computational cost. Rizk *et al.* [83] used MATLAB's parallel tool to distribute feature space transition to multiple works, and applied the clustering algorithm to a single worker to achieve parallelization. Graphics processing units (GPUs) has become parallel processing tools due to its high computing power and low

TABLE 1. Distributed ELM Comparison.

Name	Ensemble	Matrix Decomposition	Kernel	OS-ELM	Platform	MPI	GPU	Double Classifier	Scalability	Stability	Efficiency
OS-ELM-based P2P	✓[1]			✓			✓				⋆[2]
Multiple ELM	✓				✓						⋆
M3-ELM	✓				✓				✓		⋆
ELM-MapReduce	✓				✓						⋆⋆
PELM		✓			✓						⋆
ELM*		✓			✓				✓		⋆⋆
A-ELM*		✓			✓				✓		⋆⋆⋆
E^{2}LM		✓		✓	✓				✓		⋆⋆⋆
POS-ELM		✓		✓	✓						⋆⋆⋆
BPOS-ELM		✓		✓	✓						⋆⋆⋆
PEOS-ELM	✓			✓	✓						⋆⋆
AAELM+MLR		✓			✓			✓			⋆
WE-ELM	✓	✓			✓						⋆⋆
DW-ELM		✓			✓				✓		⋆⋆
IDW-ELM		✓			✓				✓		⋆⋆⋆
DC-ELM		✓		✓	✓				✓	✓	⋆
ELM-Tree		✓			✓			✓			⋆
DK-ELM		✓	✓		✓				✓		⋆⋆
OS-ELM-Ker			✓	✓	✓						⋆
P-ELM	✓		✓								⋆
SELM		✓			✓						⋆⋆⋆
U-DELM		✓			✓				✓		⋆⋆⋆
PASS-ELM					✓						⋆⋆
MR-ELM	✓				✓					✓	⋆
CNN-ELM	✓				✓				✓		⋆
DELM	✓										⋆
PR-ELM	✓	✓					✓				⋆⋆
DGR-ELM				✓			✓				⋆
SLT-based ELM	✓				✓						⋆
GPH-ELM					✓			✓		✓	⋆⋆

[1] The algorithm satisfies this condition.
[2] The number of stars represents the degree of improvement of this algorithm over base-line.

cost, especially in the field of high-performance computing. Phusomsai *et al.* [84] used histogram gradient for feature extraction based on tumor shape images, and ensemble them into ELM as a classifier. After that, they used the parallel feasibility study and implementation to accelerate the traditional ELM on the GPU by 3 times and 7 times in the classification stage. Chen *et al.* [85] were the first to combine the memory cluster computing platform Flink and GPU to parallelize Hierarchical ELM (H-ELM), which integrates the excellent characteristics of memory cluster computing and GPU. Vanli *et al.* [86] introduced an ELM algorithm based on the gradient of the distribution formula.This algorithm provides a guaranteed upper bound for SLFN performance of each agent, and proves that each independent SLFN can asymptotically achieve the optimal SLFN performance for centralized batch processing.

There are some methods for parallelizing ELM. Although distributed processing of ELM is not implemented, some parallelization methods combined with ELM are adopted to optimize ELM.Some random hidden nodes may play an important role in the network output. Yang *et al.* [87] applied the parallel algorithm to ELM and proposed an incremental ELM based on the parallel chaos search (PC-ELM), which is used to discover hidden nodes in the network output. Ahmad and Janahiraman *et al.* [88] proposed a parallel ELM (PIPSO-ELM) based on particle swarm optimization for modeling and prediction of surface roughness and power consumption in manufacturing. PIPSO-ELM is divided into two separate algorithm blocks, each representing surface roughness and power consumption, and then the two basic ELM based performance models are combined with the selected input weight and the hidden bias of PSO. In order to improve the performance of ELM in dealing with regression problems, the existing research proposes to apply the double-parallel structure to ELM. He *et al.* [89] applied a data-attribute-space-oriented double parallel (DASODP) structure with data-oriented attribute space to ELM (DASODP-ELM). The double-parallel structure enables DASODP-ELM's output layer to receive not only information from neurons in the hidden layer, but also direct information from neurons in the input layer. Compared with ELM, DASODP-ELM with fewer parameters can achieve better performance.

Random vector functional-link (RVEL) networks can be regarded as a single hidden layer feedforward neural network resulting in a linear combination of nonlinear extensions of the original input. ELM is exactly proposed for the single hidden layer feedforward neural network. Therefore, the distributed learning algorithm proposed for RVEL may be applied to ELM. Scardapane *et al.* [90] proposed a distributed learning algorithm for training data distributed in a random vector functional-link network with a decentralized information structure. They proposed two algorithms based on decentralized average consensus (DAC) and alternating direction multiplier machine (ADMM) strategies. These algorithms work in a completely distributed manner and do not need the coordination of the central agent in the learning process. Scardapane *et al.* [91] investigated the problem of music classification when training data is distributed throughout a network of interconnected agents, and it is available in a sequential stream. Under the considered setting, the target is for all the nodes, after receiving any training data block without relying on the master node, to agree on a single classifier in a decentralized fashion. For a special class neural networks Scardapane *et al.* [92] proposed a RVEL algorithm based on the alternating direction method of multiplier optimization. The algorithm allows learning an RVFL network from multiple distributed data sources while limiting communication to a single operation that computations a distributed average.

Field Programmable Gate Arrays(FPGAs) had the potential for flexible acceleration of many workloads and had been used to accelerate large-scale tasks, providing significant performance improvements and substantial power savings. It demonstrated that they have the potential for efficient large-scale computation [93]. Yeung *et al.* [94] proposed an implementation of MapReduce library that supports parallel FPGAs and Graphics Processing Units (GPUs) to provide up to 100 times performance improvement. Choi and So [95] proposed the design and implementation of k-means clustering algorithm for computer cluster based on FPGAs acceleration. They implemented a MapReduce programming model in which both map and reduce functions executed autonomously to the CPU on multiple FPGAs and developed a hardware/software framework to manage gateway execution on multiple FPGAs across clusters. However, this performance improvement brought a significant cost because of the long development cycle required to leverage FPGAs resources. Ghasemi and Chow [96] incorporated FPGAs acceleration into Spark. They provided easy access to FPGAs resources for ordinary application developers and retained the functionality and user interfaces of currently popular distributed platform such as Spark. With the application of FPGAs in distributed platforms, it can further accelerate distributed ELM processing.

IV. CONCLUSION

Distributed ELM, as one of the hot research directions, has attracted the attention of a large number of researchers. Table 1 compares the existing distributed ELM. As can be seen from the table, the parallelization method of ELM is mainly to ensemble and decompose the multiplication operator of the most expensive Moore-Penrose generalized inverse matrix in ELM calculation. Due to the limitations of the ensemble algorithm, most of the existing ELM distributed processing methods adopt matrix operation. Moreover, the implementation of distributed ELM on big data platforms such as MapReudce has become the mainstream. OS-ELM and kernel ELM, as important varieties of ELM, have received less attention in current research. In the future, more attention can be paid to some important varieties of ELM. According to the investigation, the test time of classical ELM is 4653s, while the test time of distributed ELM is 487s on Iris dataset, and the correct rate is basically the same. Therefore, while increasing the computing speed, distributed ELM will not reduce the accuracy rate. Instead, it will only increases the hardware cost and network communication. Although distributed ELM has been successfully applied to classification, regression and other problems, and some research have been made on some improved ELM, there are still many problems to be solved:

(1) Limitation of hardware. At present, the main limitation of distributed ELM is the hardware. Computing large amounts of data using distributed ELM requires excellent hardware configuration support. In the future, we will consider to apply more advanced hardware devices such as FPGAs to distributed ELM to improve computing efficiency.

(2) Dose not apply well to specific problems. Distributed ELM is applied to the processing and analysis of big data, but due to different data application scenarios and data with different characteristics, related problems cannot be solved well. With the development of ELM research, many variants of ELM have been proposed to solve problems in different scenario. In the future, we will consider the new ELM variants for distributed processing to better apply to different problems in big data.

(3) For the new distributed environment. Existing research on distributed ELM mainly focus on the MapReduce framework. With the development of distributed computing framework, the mainstream distributed platforms have gradually evolved from MapReduce and Hadoop to Spark, Flink and so on. Although there have been studied on distributed ELM on Spark and Flink, these studies do not take full advantage of these frameworks. There are many directions to conduct distributed processing on ELM on these platforms.

REFERENCES

[1] C. Lynch, "How do your data grow?" *Nature*, vol. 455, no. 7209, pp. 28–29, Sep. 2008.

[2] R. Ranjan, L. Wang, A. Y. Zomaya, D. Georgakopoulos, X.-H. Sun, and G. Wang, "Recent advances in autonomic provisioning of big data applications on clouds," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 101–104, Apr. 2015.

[3] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, "Trends in big data analytics," *J. Parallel Distrib. Comput.*, vol. 74, no. 7, pp. 2561–2573, 2014.

[4] L. Liu and H. Jia, "Trust evaluation via large-scale complex service-oriented online social networks," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 11, pp. 1402–1412, Nov. 2015.

[5] W. F. Schmidt, M. A. Kraaijveld, and R. P. W. Duin, "Feedforward neural networks with random weights," in *Proc. 11th IAPR Int. Conf. Pattern Recognit.*, 1992, pp. 1–9.

[6] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, Apr. 1994.

[7] W. Cao, X. Wang, Z. Ming, and J. Gao, "A review on neural networks with random weights," *Neurocomputing*, vol. 275, pp. 278–287, Jan. 2018, doi: 10.1016/j.neucom.2017.08.040.

[8] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, no. 3, pp. 321–355, 1988.

[9] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.

[10] W. Cao, M. J. A. Patwary, P. Yang, X. Wang, and Z. Ming, ''An initial study on the relationship between meta features of dataset and the initialization of NNRW,'' in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[11] W. Cao, J. Gao, Z. Ming, and S. Cai, ''Some tricks in parameter selection for extreme learning machine,'' in *Proc. Int. Conf. Artif. Intell. Appl. Technol. (AIAAT)*, Silver, HI, USA, 2017, Art. no. 012002.

[12] W. Cao, J. Gao, M. Zhong, S. Cai, and Z. Hua, ''Impact of probability distribution selection on rvfl performance,'' in *Proc. Int. Conf. Smart Comput. Commun.*, 2017, pp. 114–124.

[13] A. Castaño, F. Fernández-Navarro, and C. Hervás-Martínez, ''PCA-ELM: A robust and pruned extreme learning machine approach based on principal component analysis,'' *Neural Process. Lett.*, vol. 37, no. 3, pp. 377–392, Jun. 2013, doi: 10.1007/s11063-012-9253-x.

[14] W. Cai, J. Yang, Y. Yu, Y. Song, T. Zhou, and J. Qin, ''PSO-ELM: A hybrid learning model for short-term traffic flow forecasting,'' *IEEE Access*, vol. 8, pp. 6505–6514, 2020, doi: 10.1109/ACCESS.2019.2963784.

[15] G. S. Krishnan and S. K. S., ''A novel GA-ELM model for patient-specific mortality prediction over large-scale lab event data,'' *Appl. Soft Comput.*, vol. 80, pp. 525–533, Jul. 2019, doi: 10.1016/j.asoc.2019.04.019.

[16] G.-B. Huang, D. H. Wang, and Y. Lan, ''Extreme learning machines: A survey,'' *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 107–122, Jun. 2011.

[17] X.-G. Zhao, G. Wang, X. Bi, P. Gong, and Y. Zhao, ''XML document classification based on ELM,'' *Neurocomputing*, vol. 74, no. 16, pp. 2444–2451, Sep. 2011.

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

[19] L. Zhang, X. Wang, G.-B. Huang, T. Liu, and X. Tan, ''Taste recognition in E-Tongue using local discriminant preservation projection,'' *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 947–960, Mar. 2019, doi: 10.1109/TCYB.2018.2789889.

[20] L. Zhang and D. Zhang, ''Robust visual knowledge transfer via extreme learning machine-based domain adaptation,'' *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4959–4973, Oct. 2016, doi: 10.1109/TIP.2016.2598679.

[21] L. Zhang and P. Deng, ''Abnormal odor detection in electronic nose via self-expression inspired extreme learning machine,'' *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 10, pp. 1922–1932, Oct. 2019, doi: 10.1109/TSMC.2017.2691909.

[22] Y. Li, S. Zhang, Y. Yin, W. Xiao, and J. Zhang, ''A novel online sequential extreme learning machine for gas utilization ratio prediction in blast furnaces,'' *Sensors*, vol. 17, no. 8, p. 1847, Aug. 2017, doi: 10.3390/s17081847.

[23] Y. Li, S. Zhang, Y. Yin, J. Zhang, and W. Xiao, ''A soft sensing scheme of gas utilization ratio prediction for blast furnace via improved extreme learning machine,'' *Neural Process. Lett.*, vol. 50, no. 2, pp. 1191–1213, Oct. 2019, doi: 10.1007/s11063-018-9888-3.

[24] G. Wang, Y. Zhao, and D. Wang, ''A protein secondary structure prediction framework based on the extreme learning machine,'' *Neurocomputing*, vol. 72, nos. 1–3, pp. 262–268, Dec. 2008.

[25] R. Zhang, G.-B. Huang, N. Sundararajan, and P. Saratchandran, ''Multicategory classification using an extreme learning machine for microarray gene expression cancer diagnosis,'' *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 4, no. 3, pp. 485–495, Jul. 2007.

[26] J. Zhang, W. Xiao, Y. Li, S. Zhang, and Z. Zhang, ''Multilayer probability extreme learning machine for device-free localization,'' *Neurocomputing*, vol. 396, pp. 383–393, Jul. 2020.

[27] J. Cao, J. Hao, X. Lai, C.-M. Vong, and M. Luo, ''Ensemble extreme learning machine and sparse representation classification,'' *J. Franklin Inst.*, vol. 353, no. 17, pp. 4526–4541, Nov. 2016.

[28] J. Zhang, W. Xiao, Y. Li, and S. Zhang, ''Residual compensation extreme learning machine for regression,'' *Neurocomputing*, vol. 311, pp. 126–136, Oct. 2018, doi: 10.1016/j.neucom.2018.05.057.

[29] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, ''Extreme learning machine: Algorithm, theory and applications,'' *Artif. Intell. Rev.*, vol. 44, no. 1, pp. 103–115, Jun. 2015.

[30] G. Huang, G.-B. Huang, S. Song, and K. You, ''Trends in extreme learning machines: A review,'' *Neural Netw.*, vol. 61, pp. 32–48, Jan. 2015.

[31] J. Cao and Z. Lin, ''Extreme learning machines on high dimensional and large data applications: A survey,'' *Math. Problems Eng.*, vol. 2015, pp. 1–13, Dec. 2015.

[32] T. Guo, L. Zhang, and X. Tan, ''Neuron pruning-based discriminative extreme learning machine for pattern classification,'' *Cognit. Comput.*, vol. 9, no. 4, pp. 581–595, Aug. 2017, doi: 10.1007/s12559-017-9474-4.

[33] Y. Liu, L. Zhang, P. Deng, and Z. He, ''Common subspace learning via cross-domain extreme learning machine,'' *Cognit. Comput.*, vol. 9, no. 4, pp. 555–563, Aug. 2017, doi: 10.1007/s12559-017-9473-5.

[34] L. Zhang and D. Zhang, ''Evolutionary cost-sensitive extreme learning machine,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 12, pp. 3045–3060, Dec. 2017, doi: 10.1109/TNNLS.2016.2607757.

[35] L. Zhang, Z. He, and Y. Liu, ''Deep object recognition across domains based on adaptive extreme learning machine,'' *Neurocomputing*, vol. 239, pp. 194–203, May 2017, doi: 10.1016/j.neucom.2017.02.016.

[36] G. B. Huang, Q. Y. Zhu, and C. K. Siew, ''Extreme learning machine: A new learning scheme of feedforward neural networks,'' in *IEEE Int. Joint Conf. Neural Netw.*, 2005, pp. 985–990.

[37] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, ''A fast and accurate online sequential learning algorithm for feedforward networks,'' *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

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

[39] Q. Liu, Q. He, and Z. Shi, ''Extreme support vector machine classifier,'' in *Proc. 12th Pacific-Asia Conf.*, Osaka, Japan, May 2008, pp. 222–233.

[40] B. Frénay and M. Verleysen, ''Using SVMs with randomised feature spaces: An extreme learning approach,'' in *Proc. 18th Eur. Symp. Artif. Neural Netw.*, Bruges, Belgium, Apr. 2010, pp. 1–8.

[41] D. Opitz and R. Maclin, ''Popular ensemble methods: An empirical study,'' *J. Artif. Intell. Res.*, vol. 11, pp. 169–198, Aug. 1999.

[42] J. Xin, Z. Wang, C. Chen, L. Ding, G. Wang, and Y. Zhao, ''Elm: Distributed extreme learning machine with mapreduce,'' *World Wide Web*, vol. 17, no. 5, pp. 1189–1204, 2014.

[43] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse, ''GPU-accelerated and parallelized ELM ensembles for large-scale regression,'' *Neurocomputing*, vol. 74, no. 16, pp. 2430–2437, Sep. 2011.

[44] Y. Sun, Y. Yuan, and G. Wang, ''An OS-ELM based distributed ensemble classification framework in P2P networks,'' *Neurocomputing*, vol. 74, no. 16, pp. 2438–2443, Sep. 2011.

[45] X.-L. Wang, Y.-Y. Chen, H. Zhao, and B.-L. Lu, ''Parallelized extreme learning machine ensemble based on min–max modular network,'' *Neurocomputing*, vol. 128, pp. 31–41, Mar. 2014.

[46] J. Chen, G. Zheng, and H. Chen, ''ELM-MapReduce: MapReduce accelerated extreme learning machine for big spatial data analysis,'' in *Proc. 10th IEEE Int. Conf. Control Autom. (ICCA)*, Hangzhou, China, Jun. 2013, pp. 400–405.

[47] D. Liang, L. Shuai, H. Chen, and W. Zhu, ''Improvement of ELM algorithm for multi-object identification in gesture interaction,'' Tech. Rep., 2016.

[48] S. Huang, B. Wang, J. Qiu, J. Yao, G. Wang, and G. Yu, ''Parallel ensemble of online sequential extreme learning machine based on MapReduce,'' *Neurocomputing*, vol. 174, pp. 352–367, Jan. 2016.

[49] Y. Li, S. Zhang, Y. Yin, W. Xiao, and J. Zhang, ''Parallel one-class extreme learning machine for imbalance learning based on Bayesian approach,'' *J. Ambient Intell. Humanized Comput.*, Sep. 2018, pp. 1–8.

[50] B.-L. Lu and M. Ito, ''Task decomposition and module combination based on class relations: A modular neural network for pattern classification,'' *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1244–1256, Dec. 1999.

[51] J. Chen, H. Chen, X. Wan, and G. Zheng, ''MR-ELM: A MapReduce-based framework for large-scale ELM training in big data era,'' *Neural Comput. Appl.*, vol. 27, no. 1, pp. 101–110, Jan. 2016.

[52] B. Wu, T. H. Yan, X. S. Xu, B. He, and W. H. Li, ''A MapReduce-based ELM for regression in big data,'' in *Proc. 17th Int. Conf.*, Yangzhou, China, Oct. 2016, pp. 164–173.

[53] F. Ö. Çatak, ''Classification with boosting of extreme learning machine over arbitrarily partitioned data,'' *Soft Comput.*, vol. 21, no. 9, pp. 2269–2281, May 2017.

[54] A. Budiman, M. I. Fanany, and C. Basaruddin, ''Distributed averaging CNN-ELM for big data,'' *CoRR*, vol. abs/1610.02373, pp. 1–15, Dec. 2016.

[55] Q. He, T. Shang, F. Zhuang, and Z. Shi, ''Parallel extreme learning machine for regression based on MapReduce,'' *Neurocomputing*, vol. 102, pp. 52–58, Feb. 2013.

[56] J. Xin, Z. Wang, L. Qu, G. Yu, and Y. Kang, ''A-ELM: Adaptive distributed extreme learning machine with MapReduce,'' *Neurocomputing*, vol. 174, pp. 368–374, Jan. 2016.

[57] J. Xin, Z. Wang, L. Qu, and G. Wang, "Elastic extreme learning machine for big data classification," *Neurocomputing*, vol. 149, pp. 464–471, Feb. 2015.
[58] V. Tejasviram, H. Solanki, V. Ravi, and S. Kamaruddin, "Auto associative extreme learning machine based non-linear principal component regression for big data applications," in *Proc. 10th Int. Conf. Digit. Inf. Manage. (ICDIM)*, Oct. 2015, pp. 223–228.
[59] R. Wang, Y.-L. He, C.-Y. Chow, F.-F. Ou, and J. Zhang, "Learning ELM-tree from big data based on uncertainty reduction," *Fuzzy Sets Syst.*, vol. 258, pp. 79–100, Jan. 2015.
[60] L. Oneto, F. Bisio, E. Cambria, and D. Anguita, "Statistical learning theory and ELM for big social data analysis," *IEEE Comput. Intell. Mag.*, vol. 11, no. 3, pp. 45–55, Aug. 2016.
[61] L. Oneto, F. Bisio, E. Cambria, and D. Anguita, "SLT-based ELM for big social data analysis," *Cognit. Comput.*, vol. 9, no. 2, pp. 259–274, Apr. 2017.
[62] M. Duan, K. Li, X. Liao, and K. Li, "A parallel multiclassification algorithm for big data using an extreme learning machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2337–2351, Jun. 2018.
[63] C. Chen, K. Li, A. Ouyang, and K. Li, "A parallel approximate SS-ELM algorithm based on MapReduce for large-scale datasets," *J. Parallel Distrib. Comput.*, vol. 108, pp. 85–94, Oct. 2017.
[64] Z. Wang, L. Qu, J. Xin, H. Yang, and X. Gao, "A unified distributed ELM framework with supervised, semi-supervised and unsupervised big data learning," *Memetic Comput.*, vol. 11, no. 3, pp. 305–315, Sep. 2019.
[65] Z. Wang, J. Xin, S. Tian, and Y. Ge, "Distributed weighted extreme learning machine for big imbalanced data learning," in *Proc. ELM*, 2016, pp. 319–332.
[66] Z. Wang, J. Xin, H. Yang, S. Tian, G. Yu, C. Xu, and Y. Yao, "Distributed and weighted extreme learning machine for imbalanced big data learning," *Tsinghua Sci. Technol.*, vol. 22, no. 2, pp. 160–173, Apr. 2017.
[67] A. Wu and W. Chen, "ELM-based distributed cooperative learning over networks," *CoRR*, vol. abs/1504.00981, pp. 1–7, Dec. 2015.
[68] B. Wang, S. Huang, J. Qiu, Y. Liu, and G. Wang, "Parallel online sequential extreme learning machine based on MapReduce," *Neurocomputing*, vol. 149, pp. 224–232, 2015.
[69] H. Shan, B. Wang, Y. Chen, G. Wang, and Y. Ge, "Efficient batch parallel online sequential extreme learning machine algorithm based on MapReduce," in *Proc. ELM*, 2016, pp. 1–9.
[70] S. Huang, B. Wang, Y. Chen, G. Wang, and G. Yu, "An efficient parallel method for batched OS-ELM training using MapReduce," *Memetic Comput.*, vol. 9, no. 3, pp. 183–197, Sep. 2017.
[71] X. Bi, X. Zhao, G. Wang, P. Zhang, and C. Wang, "Distributed extreme learning machine with kernels based on MapReduce," *Neurocomputing*, vol. 149, pp. 456–463, Feb. 2015.
[72] N. Karthick and D. Kalarani, "Efficient big data classification through distributed kernel-based extreme learning machine approach," *Indian J. Innov. Develop.*, vol. 5, no. 4, pp. 1–13, 2016.
[73] N. Pandeeswari, D. Vignesh, R. Pushpalakshmi, and Varadharajan, "Online sequential extreme learning algorithm with kernels for bigdata classification," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 1–8.
[74] X. Zhao, Z. Ma, and Z. Zhang, "A novel recommendation system in location-based social networks using distributed ELM," *Memetic Comput.*, vol. 10, no. 3, pp. 321–331, Sep. 2018.
[75] Z. Qiu, H. Zou, H. Jiang, L. Xie, and Y. Hong, "Consensus-based parallel extreme learning machine for indoor localization," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[76] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
[77] W. Xiao, J. Zhang, Y. Li, S. Zhang, and W. Yang, "Class-specific cost regulation extreme learning machine for imbalanced classification," *Neurocomputing*, vol. 261, pp. 70–82, Oct. 2017, doi: 10.1016/j.neucom.2016.09.120.
[78] S. Ding, B. Mirza, Z. Lin, J. Cao, X. Lai, T. V. Nguyen, and J. Sepulveda, "Kernel based online learning for imbalance multiclass classification," *Neurocomputing*, vol. 277, pp. 139–148, Feb. 2018, doi: 10.1016/j.neucom.2017.02.102.
[79] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, Feb. 2013.
[80] D.-H. Han, X. Zhang, and G.-R. Wang, "Classifying uncertain and evolving data streams with distributed extreme learning machine," *J. Comput. Sci. Technol.*, vol. 30, no. 4, pp. 874–887, Jul. 2015.
[81] Y. Wang, Y. Dou, X. Liu, and Y. Lei, "PR-ELM: Parallel regularized extreme learning machine based on cluster," *Neurocomputing*, vol. 173, pp. 1073–1081, Jan. 2016.
[82] Y. Kokkinos and K. G. Margaritis, "Big data regression with parallel enhanced and convex incremental extreme learning machines," *Comput. Intell.*, vol. 34, no. 3, pp. 875–894, Aug. 2018.
[83] Y. Rizk and M. Awad, "On the distributed implementation of unsupervised extreme learning machines for big data," in *Proc. Conf. Big Data*, San Francisco, CA, USA, Aug. 2015, pp. 167–174.
[84] W. Phusomsai, C. So-In, C. Phaudphut, C. Thammasakorn, and W. Punjaruk, "Brain tumor cell recognition schemes using image processing with parallel ELM classifications on GPU," in *Proc. 13th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE)*, Jul. 2016, pp. 1–6.
[85] C. Chen, K. Li, A. Ouyang, Z. Tang, and K. Li, "GPU-accelerated parallel hierarchical extreme learning machine on flink for big data," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 10, pp. 2740–2753, Oct. 2017.
[86] N. D. Vanli, M. O. Sayin, I. Delibalta, and S. S. Kozat, "Sequential nonlinear learning for distributed multiagent systems via extreme learning machines," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 546–558, Mar. 2017.
[87] Y. Yang, Y. Wang, and X. Yuan, "Parallel chaos search based incremental extreme learning machine," *Neural Process. Lett.*, vol. 37, no. 3, pp. 277–301, Jun. 2013.
[88] N. Ahmad and T. V. Janahiraman, "Modelling and prediction of surface roughness and power consumption using parallel extreme learning machine based particle swarm optimization," in *Proc. ELM*, vol. 2, 2015, pp. 1–9.
[89] Y.-L. He, Z.-Q. Geng, and Q.-X. Zhu, "A data-attribute-space-oriented double parallel (DASODP) structure for enhancing extreme learning machine: Applications to regression datasets," *Eng. Appl. Artif. Intell.*, vol. 41, pp. 65–74, May 2015.
[90] S. Scardapane, D. Wang, M. Panella, and A. Uncini, "Distributed learning for random vector functional-link networks," *Inf. Sci.*, vol. 301, pp. 271–284, Apr. 2015, doi: 10.1016/j.ins.2015.01.007.
[91] S. Scardapane, R. Fierimonte, D. Wang, M. Panella, and A. Uncini, "Distributed music classification using random vector functional-link nets," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8, doi: 10.1109/IJCNN.2015.7280333.
[92] S. Scardapane, M. Panella, D. Comminiello, and A. Uncini, "Learning from distributed data sources using random vector functional-link networks," in *Proc. Conf. Big Data*, San Francisco, CA, USA, Aug. 2015, pp. 468–477, doi: 10.1016/j.procs.2015.07.324.
[93] A. Putnam, "A reconfigurable fabric for accelerating large-scale datacenter services," *Commun. ACM*, vol. 59, no. 11, pp. 114–122, Oct. 2016.
[94] J. H. C. Yeung, C. C. Tsang, K. H. Tsoi, B. S. H. Kwan, C. C. C. Cheung, A. P. C. Chan, and P. H. W. Leong, "Map-reduce as a programming model for custom computing machines," in *Proc. 16th Int. Symp. Field-Program. Custom Comput. Mach.*, Stanford, Palo Alto, CA, USA, Apr. 2008, pp. 149–159, doi: 10.1109/FCCM.2008.19.
[95] Y. Choi and H. So, "Map-reduce processing of k-means algorithm with fpga-accelerated computer cluster," Tech. Rep., 2014.
[96] E. Ghasemi and P. Chow, "Accelerating apache spark big data analysis with fpgas," Tech. Rep., 2016.

ZHIQIONG WANG (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science and technology from Northeastern University, China, in 2008 and 2014, respectively. She visited the National University of Singapore and The Chinese University of Hong Kong, in 2010 and 2013, respectively, as the Academic Visitor. She is currently an Associate Professor with the College of Medicine and Biological Information Engineering, Northeastern University. She has published more than 60 articles. Her main research interests are the computer-aided diagnosis, medicine information, big health data analysis, cloud computing, and machine learning.

LING SUI received the bachelor's degree in computer science and technology from Liaoning Shihua University, in 2018. She is currently pursuing the M.D. degree with the College of Computer Science and Engineering, Northeastern University. Her main research interests are big data development, machine learning, and gene regulatory network.

JUNCHANG XIN received the B.Sc., M.Sc., and Ph.D. degrees in computer science and technology from Northeastern University, China, in 2002, 2005, and 2008, respectively. He visited the National University of Singapore as Postdoctoral Visitor (April 2010–April 2011). He is currently a Professor with the School of Computer Science and Engineering, Northeastern University, China. He has published more than 60 research papers. His research interests include big data, uncertain data, and bioinformatics.

LUXUAN QU received the bachelor's degree in automatics from Northeast Dianli University, in 2010, and the master's degree in biomedical engineering from Northeastern University, in 2014. She is currently pursuing the Ph.D. degree with the College of Medicine and Biological Information Engineering. Her main research interests are gene regulatory networks, machine learning, and cloud computing.

YUDONG YAO (Fellow, IEEE) received the B.Eng. and M.Eng. degrees from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1982 and 1985, respectively, and the Ph.D. degree from Southeast University, Nanjing, in 1988, all in electrical engineering. From 1989 and 1990, he was at Carleton University, Ottawa, Canada, as a Research Associate, working on mobile radio communications. From 1990 to 1994, he was with Spar Aerospace Ltd., Montreal, Canada, where he was involved in research on satellite communications. From 1994 to 2000, he was with Qualcomm Inc., San Diego, CA, USA, where he participated in research and development in wireless code-division multiple-access (CDMA) systems. He has been with the Stevens Institute of Technology, Hoboken, NJ, USA, since 2000, and is currently a professor and the Department Director of Electrical and Computer Engineering. He is also a Professor with the College of Medicine and Biological Information Engineering, Northeastern University, and the Director of the Stevens' Wireless Information Systems Engineering Laboratory (WISELAB). He holds one Chinese patent and twelve U.S. patents. His research interests include wireless communications and networks, spread spectrum and CDMA, antenna arrays and beamforming, cognitive and software-defined radio (CSDR), and digital signal processing for wireless systems. He was an Associate Editor of the IEEE COMMUNICATIONS LETTERS and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and an Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.