Received November 5, 2018, accepted November 28, 2018, date of publication December 5, 2018, date of current version January 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2885164

Rich Feature Combination for Cost-Based Broad Learning System

TIAN-LUN ZHANG, RONG CHEN, (Member, IEEE), XI YANG, (Student Member, IEEE), AND SHIKAI GUO
College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China

Corresponding author: Rong Chen (rchen@dlmu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61672122 and Grant 61602077.

ABSTRACT Broad learning system (BLS) is an emerging learning algorithm for the connectionist models, which have enjoyed much popularity on many applications. As an alternative approach of learning in deep structure, the BLS develops an incremental learning neural network that can be modeled in a flexible way, and becomes a promising technique in the field of knowledge discovery and data engineering. To further improve the performance of BLS, our focus is to investigate these algorithms which can enhance the BLS. On one hand, from the viewpoint of feature engineering, unsupervised group-wise encoding is conducted for feature extraction, and broadly fused feature representation is used to improve the ability of BLS, in terms of the learning and reusing multi-level features. On the other hand, for imbalanced learning from disproportionate size of categories instances, a cost-sensitive BLS framework is proposed in this paper, which aims to minimize the total misclassifying cost in classification learning. Finally, we conduct extensive experiments on a wide range of datasets (e.g., computer vision and bug reports) to demonstrate the effectiveness of the proposed BLS framework.

INDEX TERMS Broad learning system, broadly-fused feature, imbalanced learning, neural network, cost-sensitive learning.

I. INTRODUCTION

With the continuous expansion of attention devoted to connectionist models, deep learning systems become critical to advance the developments in a wide range of applications, e.g., computer vision [1]–[4], natural language processing (NLP) [5], [6], and software engineering [7], [8]. In this field, there mainly exist two research topics, namely, deep structure design and deep learning approach. In regards to deep structure, a widely used network topology is the fully-connected architecture, in which one neuron is connected by all neurons in the previous layer [9]–[11]. The other remarkable neural model is convolutional neural network (CNN) which has been utilized as early as the nineties to address the task of handwriting character recognition [12]. During the past years, various variants of CNN have been extensively studied, and become dominant deep structure in widespread applications, such as ZFNet [13], VGG [14], GoogLeNet [15]. Apart from feedforward neural networks, the recurrent neural networks (RNNs) with cyclical connections are employed to solve the temporal tasks [16], [17]. The improved RNNs with memory block, e.g. LSTM [18] and GRU [19], are widely used in NLP [18]–[21] and speech recognition [22]. In addition, a lot of attention recently are focusing on residual unit [23] which is critical component for deeper neural networks (e.g., the network with 100 or 1000 layers), and is a significant milestone in the investigation of deep structure design [24], [25]. Meanwhile, the deep learning approaches also have enjoyed much popularity [26]–[28]. A standard optimization algorithm is the well-known backpropagation (BP [25], [29], [30]) which repeatedly applies chain rule for partial derivatives of parameters. For sequence neural models, backpropagation through time (BPTT [31], [32]) and connectionist temporal classification (CTC [33]) are proposed. To tackle the vanishing gradient problem encountered in training deep networks, Hinton *et al.* [34] proposed the early deep learning framework, in which the restricted Boltzmann machine [35], [36] acts as the building block for multiple hidden layers neural network. Afterwards, a series of excellent deep learning approaches have been proposed, such as Dropout [37], Batch Normalization [38] and Group Normalization [39]. It is worth mentioning that the random weight is an effectiveness and efficiency solution for training deep architectures [40], [41]. In particular, the universal approximation capability of extreme learning machine (ELM) has been proved by Huang *et al.* [42], [43]. Moreover, several valuable theories and applications [44]–[48] have been conducted on random weight.

However, the deep learning works still face several issues, especially, a time-consuming training process is the inherent disadvantage in this field. Moreover, when the learned network is inadequate to represent the distributive characteristics of the data, this time-consuming training process has to be repeated completely. To address this issue, an emerging connectionist model is proposed by Chen and Liu [49], namely, the broad learning system (BLS). The BLS is an efficiency incremental learning approach to quickly remodel a learned neural network without retraining process from beginning. Instead of increasing the depth of neural model, the BLS aims to expand the width of neural network for solving high dimensional data problem. Furthermore, the universal approximation capability of BLS has been proved by Chen *et al.* [50]. Based on the BLS, a novel neuro-fuzzy model (Fuzzy BLS) is proposed for classification and regression learnings [51]. In addition, a regularized robust BLS is developed to learn the characteristics of uncertain data. Due to the good generalization, flexible remodeling process and universal approximation capability, the BLS is a promising alternative approach of deep learning.

According to the basic theories of the BLS, we conduct some further investigations in this paper. One can easily see that the original BLS is a flat neural network, in which the orthogonal initialization is necessary to avoid redundant features. Besides, the level of features is simple in the original model. Recently, the fusion of multi-level features has successfully been applied to numerous existing researches, such as Inception architecture [15], [52], [53], densely connected CNN [54], and feature pyramid networks [55]. Motivated by those aforementioned works, we proposed rich feature combination for broad learning system (R-BLS). Unlike the cascaded BLS [50], instead of using random features, the proposed R-BLS is developed by using an encoder as the building block. For sparse and compact feature representations, the l_1 penalty is employed in this building block. The R-BLS thus can be considered as an improved variant of the cascaded BLS, with respect to the flexibility of modeling technique. In the R-BLS framework, the learning process contains unsupervised group-wise feature extraction and supervised learning with l_2-norm. The former provides broadly-fused feature representation which is used to improve the ability of BLS, in terms of the learning and reusing multi-level features.

In classification learning, analogously to most standard classifiers [56]–[59], when learning from imbalanced dataset, the BLS will fail to properly generalize inductive rules over the instance space, and resultantly make unfavorable decisions during evaluation phase. Imbalanced distribution thus presents a new challenge to the BLS community. To bridge the gap between BLS and imbalanced learning, the second topic of our study is cost-sensitive BLS. This model breaks the unrealistic hypothesis that the costs of all classification errors are equal. In practice, the costs among different misclassification errors trend to be drastically various, namely, the minority class instances usually have larger misclassifying costs than the majority class instances [60], [61]. And the optimization objective of the cost-sensitive model is to minimize the total misclassifying cost in classification learning. Note that this proposed imbalanced learning is a universal approach for all variants of BLS mentioned above. In sum, the main contributions of this paper are as follows.

1) A novel broad learning system with rich feature combination is proposed in this paper. To acquire sparse and compact feature representations, an unsupervised group-wise encoding process is developed via l_1-norm optimization. The broadly-fused feature representation in the proposed model exploits the potential of the BLS through feature reuse and multi-level feature learning.
2) A universal cost-sensitive framework is proposed to the BLS community. This framework is used to bridge the gap between BLS and imbalanced learning. In this case, the BLS is extended to a classification learning with a cost-sensitive matrix for different misclassifying errors. And the performances of BLS thus are boosted by imbalanced learning, when presented with complex class imbalance situation.
3) We show that the proposed R-BLS framework achieves promising performance in different kinds of computer tasks. Furthermore, the experiment of imbalanced learning is conducted on bug report triaging system which is the popular research in the field of intelligent software systems. Compared with prevalent imbalanced learnings, the proposed cost-sensitive BLS framework achieves better performance to address imbalanced problem.

The rest of this paper is organized as follows. In section II, we review the works of broad learning system, including the design of structure and the incremental learning algorithm. In section III, we describe the details of the proposed R-BLS and cost-sensitive framework for the BLS community. In section IV, comprehensive experiments are conducted on varying types of tasks, including character recognition, face recognition, bugs triaging and so on. At last, we conclude our work in the section V.

II. RELATED WORK

As an emerging connectionist model, the BLS is a valuable approach of learning in neural network structure. To facilitate the description of our proposed algorithm, we will briefly review the main concepts of BLS, including the fundamental network structure and incremental modeling approach. One can see the details referred to in [49].

A. BLS NETWORK STRUCTURE

Without loss of generality, classification learning is discussed in this section. To ease the presentation, we use S to denote a set of samples whose labels are known in advance. S is referred to as the training data, and can be defined as:

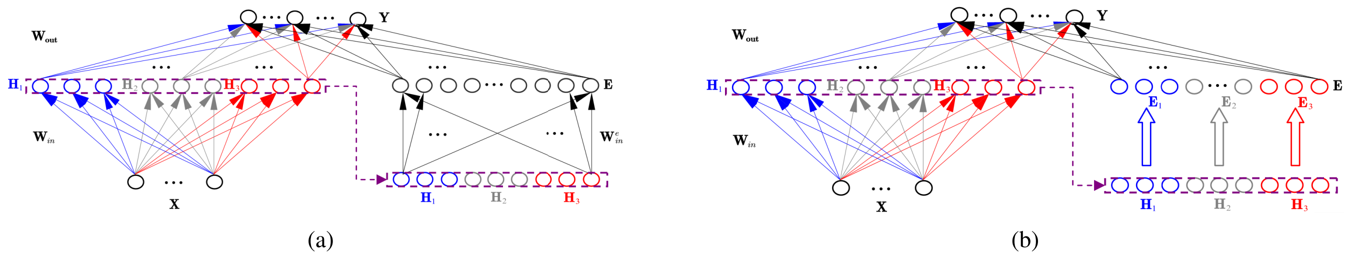


FIGURE 1. **(a) shows a broad learning system in which the weight between H and E is W_{in}^e; (b) shows the other connecting way which maps H to E group-wise.**

$S = \{(x_i, t_i)\}, i = 1, \cdots N$, where N is the total number of samples (i.e., $|S| = N$), $x_i \in \mathbb{R}^{1\times m}$ is a sample in the m-dimensional feature space, and $t_i \in \{1, \cdots C\}$ is the label associated with the ith sample x_i, C is the number of classes. As shown in Figure 1, the first part of BLS is a group of maps, each of them can be represented as

$$H_j = \varphi_j(XW_j + \gamma_j), \quad j = 1, \cdots, M \tag{1}$$

where X is one input matrix $[x_i]_{N\times m}$, $H_j \in \mathbb{R}^{N\times k}$ is the jth set of features learned from X, $W_j \in \mathbb{R}^{m\times k}$ is the weight matrix between X and H_j; $\gamma_j \in \mathbb{R}^{N\times k}$ is the bias, k is the number of latent neurons associated with H_j, and $\varphi_j(\cdot) : \mathbb{R}^{N\times k} \rightarrow \mathbb{R}^{N\times k}$ denotes the jth activation function applied to the excitation vector element-wise. Note that here the H_j can be viewed as a non-linear feature of the input data X, and the non-linear maps can be one of the popular activation functions (e.g., sigmoid function and hyperbolic tangent function), or be the combination of them. These various non-linear feature components then are concatenated to be a mixture feature representation $H = [H_1, \cdots, H_M]$, which is taken as input for the next part of BLS, namely the enhancement nodes. This part can be acquired from H in two ways, see Figure 1 (a) and (b). Mathematically, the maps are represented as follows

$$E = \psi(HW_{in}^e + \eta) \tag{2}$$

or

$$E_j = \psi_j(H_jW_j^e + \eta_j) \tag{3}$$

where, in the first way, H is directly mapped to the feature E of enhancement layer, W_{in}^e is the weight matrix, η is the bias of the enhancement layer; in the other way, each H_j is represented as its relevant feature E_j through a non-linear feature detector ψ_j, and $W_j^e \in W_{in}^e = \{W_1^e, W_2^e, \cdots, W_M^e\}$ is the weight connecting H_j and E_j, η_j is the corresponding bias. Obviously, the latter $E = [E_1, \cdots, E_M]$ has much more mixture representations than the former. The output of BLS hence can be computed by the following equation

$$\begin{aligned} Y &= [H_1, \cdots, H_M, E_1, \cdots, E_M]W_{out} \\ &= [H, E]W_{out} \end{aligned} \tag{4}$$

where $Y \in \mathbb{R}^{N\times C}$ is the output matrix of BLS, W_{out} denotes the weight matrix of output layer. The optimization objective of BLS is to minimize the expected loss as follows

$$J_{EL} = \mathbb{E}(J(\theta; S)) = \int_x J(\theta; x; t)p(x)d(x) \tag{5}$$

in which $\theta = (W_{in}, W_{in}^e, W_{out}; \gamma, \eta)$ are the parameters of BLS, $p(x)$ is the probability density function of observation x. Unfortunately, $p(x)$ is typically unknown and needs to be estimated from S, thus model parameters θ are often trained to optimize an empirical criteria. Here, the empirical criteria is mean square error (MSE) criterion

$$J_{MSE}(\theta; S) = \frac{1}{N}\sum_{i=1}^{N} J_{MSE}(\theta; x_i, t_i) \tag{6}$$

where

$$J_{MSE}(\theta; x_i, t_i) = \frac{1}{2}||y_i - t_i||^2 \tag{7}$$

where y_i is the hypothesis output of BLS corresponding to x_i. The matrix form of Equation (7) is

$$J_{MSE}(\theta; X, T) = \frac{1}{2}||Y - T||^2$$

where each row of T is a one-hot vector associated with one label t.

As mentioned in previous section, there traditionally are two algorithms to optimize the parameters. The first is BP that is an iterative process to reach the smallest training error. Another way, as used in ELM, is to solve the Moore-Penrose generalized inverse (MP inverse) of matrix $[H, E]$, and the output weight then is computed as:

$$W_{out} = [H, E]^{\dagger}T \tag{8}$$

where $[H, E]^{\dagger}$ is the MP inverse of $[H, E]$.

B. INCREMENTAL MODELING APPROACH

In most cases, once the trained neural model could not be good enough to represent the distributive characteristics underlying the training dataset, the whole training process from beginning will be repeated. This is a time-consuming and non-economic strategy. Furthermore, some meaningful features learned by former model will be forgotten after the new training process. Instead of retraining the whole model, the BLS remodels the trained structure in a dynamic way. Concretely, there are two incremental patterns, see the

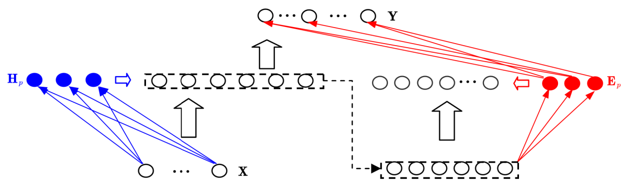


FIGURE 2. An illustration to show the BLS remodels a trained network in a dynamic way.

Figure 2, the simple one is to add an extra map between H and E, as follows

$$E_p = \psi_p(HW_p^e + \eta_p) \tag{9}$$

in which ψ_p is the extra map, W_p^e and η_p respectively represent the weight matrix and bias. The other pattern is to acquire a new H_p as follows

$$H_p = \varphi_p(XW_p + \gamma_p) \tag{10}$$

where φ_p denotes the new map, W_p and γ_p are weight matrix and bias, respectively. Then the H_p is mapped to a new E_p through Equation (9). Following the process to update $[H, E]$, the output weight W_{out} could be optimized in two ways mentioned above.

III. THE PROPOSED METHOD

In this section, a novel R-BLS framework is proposed for BLSes. The entire architecture of the R-BLS is to be introduced in detail, and a cost-based method is employed in this architecture to deal with imbalanced learning problem.

A. R-BLS FRAMEWORK

The framework of R-BLS is built in a group-wise manner, as shown in Figure 3. Unlike the existing BLS frameworks [49]–[51], one can see that the proposed R-BLS architecture is structurally composed of three separate phases: (1) unsupervised feature learning in group-wise fashion; (2) enhancement features learning; (3) supervised feature learning for final decision. For the first phase, an encoder is developed to extract useful features from each group input, which will be discussed in the following.

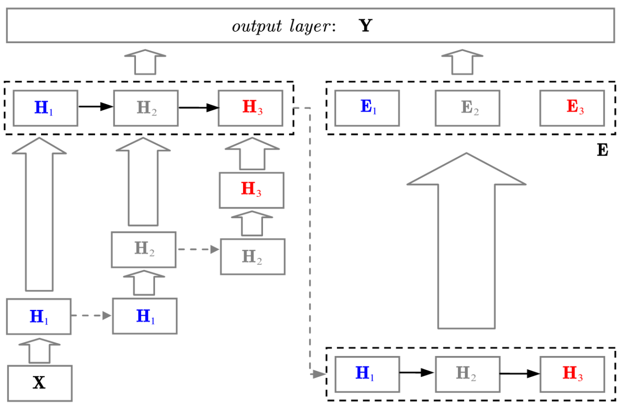


FIGURE 3. The proposed R-BLS model.

To acquire multiple high-level features from raw data, a M-group unsupervised learning is operated on the input, and lateral connections gradually transform features from lower to semantically stronger. As mentioned above, an encoder is utilized to extract compact representations from previous group output. Here, two popular kinds of encoder algorithms can be used. The one is restricted Boltzmann machine (RBM) [34], [35], which provides a closed-form representation of a target distribution underlying the input samples, and is a generative model arriving at a high-level representation. Another way is autoencoder (AE) [28], which is a typical single-hidden layer feed-forward neural network (SLFN [11], [40]). In our study, AE is adopted as the feature extractor. Concretely, AE maps input data $\mathbf{x}$ to a latent representation that is considered as a higher-level feature, as shown in Figure 4, this process can be defined as following deterministic mapping

$$\mathbf{b} = g(\mathbf{x}\alpha + \mu) \tag{11}$$

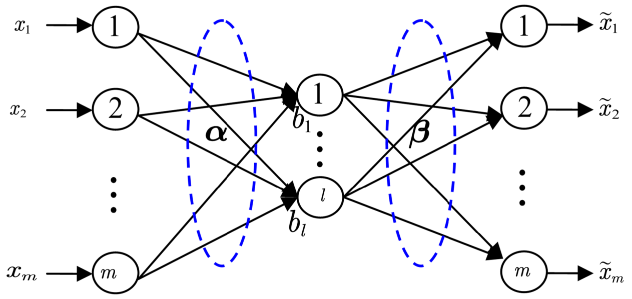


FIGURE 4. An illustration of AE.

where $\mathbf{b}$ is the hidden feature, α is the weight, μ is the bias, and g is an activation function. The resulting hidden feature $\mathbf{b}$ is then mapped to a reconstructed representation $\tilde{\mathbf{x}}$ in the original inputs space, as follows

$$\tilde{\mathbf{x}} = g(\mathbf{b}\beta + \mu') \tag{12}$$

where β is the weight matrix, μ' is the bias. Note that here the optimization objective of AE is to minimize the reconstruction errors between $\mathbf{x}$ and $\tilde{\mathbf{x}}$, and the latent representation $\mathbf{b}$ learned from nonlinear encoding would reflect the compact feature of the AE input [62]. There are different methods to optimize parameters of AE, e.g., [63], [64] and so on. For efficient training process, the random projection [65], [66] is used as the encoder which maps input data to a random variable space. In most existing studies, random features achieve good performance in wide range of applications [67], [68]. Concretely, the α are parameterized by searching the path from a random space, and will be fixed to extract latent features by using Equation (11). The optimization model of AE thus can be denoted as following equation

$$\beta^* = \arg\min_{\beta} ||\mathbf{b}\beta - \mathbf{x}||^2 + ||\beta||_{l_1} \tag{13}$$

in which $||\cdot||^2$ denotes the MSE, l_1-norm as a penalty term is applied for more sparse and compact latent information. To ease the presentation, $||\mathbf{b}\beta-\mathbf{x}||^2$ is defined as $\xi_1(\beta)$, $||\beta||_{l_1}$ is denoted as $\xi_2(\beta)$. Furthermore, the optimization problem of β can be solved by the fast iterative shrinkage-thresholding algorithm [69] whose implementation can be represented in Algorithm 1.

Algorithm 1 Fast Iterative Shrinkage-Thresholding Algorithm

```
Input:
    the Lipschitz constant $\gamma$ of $\nabla\xi_1$
    The maximum number of iterations $MaxIter$
Output: $\beta*$
1  initialization;
2  $\alpha_1 = \beta_0$; $t_1 = 1$;
3  $step = 1$;
4  while $step \leqslant MaxIter$ do
5      $\beta_{step}= \arg\min_{\beta}\{\frac{\gamma}{2}||\beta-(\alpha_{step}-\frac{1}{\gamma}\nabla\xi_1(\alpha_{step}))||^2+\xi_2(\beta)\}$;
6      $t_{step+1}=\frac{1+\sqrt{1+4t_{step}^2}}{2}$;
7      $\alpha_{step+1}=\beta_{step}+(\frac{t_{step}-1}{t_{step+1}})(\beta_{step}-\beta_{step-1})$;
8      $step$++;
9  end
10 return $\beta^* = \beta_{step}$;
```

When the above iterative steps complete, the transposition of resultant bases β^* is used as the lateral connection between two groups as shown in Figure 3. These groups as building blocks for overall architecture provide various levels of feature H_i of the original input data $X, i = 1, \cdots, M$. In particular, the output of one group can serve as input for next one. Generally, the latter feature is semantically stronger than previous ones. As proved by [53]–[55], the low-level features also are helpful for improvement of final decision. Thus, the output of all groups naturally can be fused into a feature set $H = [H_1, H_2, \cdots, H_M]$, which is totally different from the original BLS frameworks [49], and can be termed as broadly-fused feature representation (BFR) in which one component is computed as follows

$$H_i = g(H_{i-1}\beta_i^T + \mu_i), \quad i = 1, \cdots, M \tag{14}$$

where $H_0 = X$. Based on BFR, the proposed R-BLS eliminates the orthogonal initialization used in BLS. Due to the advantage of incremental learning algorithm, BLS can remodel the network without an entire retraining from beginning. Obviously, the BFR also maintains the ability to expand itself in two different incremental ways which are mathematically denoted as

$$H_p^i = g_i(H_p^{i-1}W_p^i + \gamma_p^i) \tag{15}$$

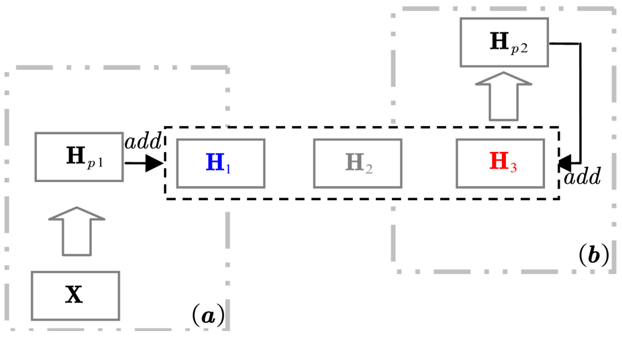


FIGURE 5. Two incremental learning ways in R-BLS.

Here, $i = 1, \cdots, m$, if $H_p^0 = X$, $H_{p1} = [H_p^i]_{i=1}^m$ is a set of features associated with an extra branch originating from input data; on the other hand, if $H_p^0 = H_M$, an extra branch follows the H, and the feature set is H_{p2}. In sum, the two methods can be illustrated in Figure 5, in which (a) represents the first way beginning from the original input data X, (b) shows the incremental process taking output of the last group as input. Note that the encoding process could be one of AE and RBM but also are both of them, in this case, BLS represents its strong flexibility. The final BFR as one H is transmitted to the next phase, namely enhancement features learning, which can be viewed as one non-linear feature detector for BFR. The enhancement nodes are acquired from BFR as follows

$$E = g(HW_{in}^e + \eta) \tag{16}$$

in which E is the enhancement feature, W_{in}^e is the weight matrix between H and E, η is the bias of enhancement layer. The combination of BFR and enhancement nodes is denoted as $[H, E]$, and R-BLS aims to minimize the training error but also the norm of output weights

$$Minimize : ||[H, E]W_{out} - T||^2 + ||[H, E]||_{l_2} \tag{17}$$

where T is the target output matrix, in order to improve the stability and avoid overfitting, l_2-norm $||\cdot||_{l_2}$ as one penalty term for big weights is applied as a component in objective function. According to Moore-Penrose generalized inverse, the calculation of optimization solution W_{out} can be given as

$$W_{out} = (\varepsilon I + [H, E]^T[H, E])^{-1}[H, E]^T T \tag{18}$$

where ε is a positive value to adjust the singularity of matrix, I is an identity matrix. Apart from the backbone of R-BLS, in next section, a cost-sensitive R-BLS will be proposed to handle imbalanced data.

B. COST BASED R-BLS

In classification learning, analogously to most standard classifiers, BLSes fail to achieve favorable accuracies when presented with imbalanced dataset, because of balanced class distribution or equal misclassification costs assumption [60], [61]. We address this by developing a cost-sensitive R-BLS which contributes a common paradigm of imbalanced learning to other BLSes.

		True Class	
		min	maj
Predicted Class	min	$\boldsymbol{C}(\text{min, min})$	$\boldsymbol{C}(\text{maj, min})$
	maj	$\boldsymbol{C}(\text{min, maj})$	$\boldsymbol{C}(\text{maj, maj})$

FIGURE 6. Confusion matrix for performance evaluation.

Instead of attempting to balance distributions through sampling methods, cost-sensitive method targets the imbalanced problem by utilizing different misclassifying costs which constitute a cost matrix, as shown in Figure 6. To ease the notation, we consider a binary classification scenario, in which $S_{min} \subset S$ denotes the minority class subset, $S_{maj} \subset S$ is the set of majority class instances, the $C(min, maj)$ denotes the cost misclassifying a minority class instance as majority class, the contrary case is defined as $C(maj, min)$. Typically, there is no cost for $C(maj, maj)$ and $C(min, min)$, and $C(min, maj) > C(maj, min) > 0$. To incorporate the cost-sensitive method into R-BLS, we define classification weight as $k_{min} = C(min, min) + C(min, maj)$ and $k_{maj} = C(maj, min) + C(maj, maj)$. Then the training dataset is sorted according to class, and the $[H, E]$ is written as $[H_{min}, E_{min}; H_{maj}, E_{maj}]$, let $H_1 = [H_{min}, E_{min}], H_2 = [H_{maj}, E_{maj}]$. Thus the optimization problem is formulated below

$$\begin{aligned}\arg\min_{W_{out}} &\left\| \begin{pmatrix} k_{min}H_1W_{out} \\ k_{maj}H_2W_{out} \end{pmatrix} - \begin{pmatrix} k_{min}T_{min} \\ k_{maj}T_{maj} \end{pmatrix} \right\| \\ &= \arg\min_{W_{out}} (k_{min}||H_1W_{out} - T_{min}|| \\ &\quad + k_{maj}||H_2W_{out} - T_{maj}||) \end{aligned} \tag{19}$$

which is a weighted least square problem. The solution can be denoted as

$$W_{out} = (\varepsilon I + H_c^T H_c)^{-1} H_c^T T_c \tag{20}$$

where $H_c = [k_{min}H_1; k_{maj}H_2]$ and $T_c = [k_{min}T_{min}; k_{maj}T_{maj}]$.

IV. EXPERIMENT

Comprehensive experiments are conducted to demonstrate the efficiency and effectiveness of the proposed R-BLS framework, which is compared with original BLS and several state-of-the-art algorithms. In addition, cost-sensitive method is combined with BLS and R-BLS, which are also compared with relevant imbalanced learning algorithms.

A. COMPARISON BETWEEN R-BLS AND BLS

In this part, classification results are reported on several widely used datasets, which are available in UCI [70]. The names and associated abbreviates of these datasets are listed as follows: Balance Scale (balance), Connect-4 (connect), letter recognition (letter), Optical Recognition of Handwritten Digits (ORHD), Pen-Based Recognition of Handwritten Digits (pen), Wall-Following Robot Navigation (robot), Satlog Shuttle (sat), USPS, Waveform Database Generator 1 (wave1), Waveform Database Generator 2 (wave2). We also conduct an ablation experiment on Fashion-MNIST (FMNIST [71]) dataset consisting of 60000 training instances and 10000 testing instances, in which each instance is a gray-level image with 28×28 pixels, as illustrated in Figure 8(a). For fair comparison, in each group of experiments, the experimental settings of R-BLS are the same as BLS. Concretely, the settings are list in Table 1, in which *hids* represents the number of nodes in each group, *enhs* denotes the number of enhancement nodes. The datasets ORHD, pen, USPS and F-MNIST have given the standard partitions of training set and testing set. For the others, our general scheme is to randomly select 90% of the examples from each category as training data, and the remaining 10% of each category as the testing data.

TABLE 1. Results of BLS and R-BLS on benchmark datasets.

Dataset	BLS(%)	R-BLS(%)	*hids*	*enhs*
balance	76.67	**80.00**	(100,100,200)	500
connect	54.08	**59.66**	(100,100,500)	500
letter	88.70	**91.05**	(100,100,200)	500
ORHD	97.94	**98.44**	(800,800,1000)	200
pen	97.54	**97.66**	(500,500,1000)	1000
robot	86.36	**87.27**	(800,800,1000)	1000
sat	82.05	**82.25**	(100,100,500)	500
USPS	95.57	**95.81**	(800,800,2000)	2000
wave1	80.80	**81.60**	(300,300,500)	500
wave2	84.80	**86.40**	(300,300,800)	500
FMNIST	86.16	**87.48**	(500,500,1000)	1000

Through observing the results shown in Table 1, the performance of original BLS has indeed been boosted by broadly-fused feature representation, in terms of classification accuracy. These comparisons indicate that the features BFR are more robust than simple ones, and devote to a better performance. Furthermore, sparse autoencoder provides more compact and sparse feature representation which is the other main reason for better results. For in-depth investigation, more complicated experiments are conducted below.

B. COMPARISON WITH STATE-OF-THE-ART APPROACHES

In this section, experiments are conducted on two relevant computer vision datasets, namely *Mixed National Institute of Standards and Technology* (MNIST) handwriting character dataset [72] and *Olivetti Research Laboratory* (ORL) face dataset [73]. Besides, we compare R-BLS with several state-of-the-art algorithms including Deep Belief Networks (DBN), Stacked Auto Encoders (SAE), stacked autoencoder (SDA), Random Forest (RF), fuzzy

RBM (FRBM), SVM, MLP, CNN. To avoid unobjective training for these algorithms, the best results reported in existing works are referred in our experiments.

1) EXPERIMENT ON ORL DATASET

The ORL dataset consists of 400 gray-level facial images with ten poises in each of 40 different people, see Figure 7 for an illustration. We randomly select 5,7 or 9 images from a person for training dataset, and the remaining images are the testing dataset. All effects of data preprocessing techniques (e.g., PCA and whitened) are avoided, and we conduct experiments on the raw dataset which has 92×112 pixels for each image. The settings of *hids* and *enhs* are (500,500,800) and 1000, respectively. The results are reported in Table 2, we can see that the R-BLS achieves better accuracies on three situations.

FIGURE 7. An illustration of ORL.

TABLE 2. Results of different algorithms on ORL.

Method	5 images (%)	7 images (%)	9 images (%)
RF	91	93.33	95
CNN	86.5	91.67	95
SVM	80.5	82.5	85
KNN	76	83.33	92.5
BLS	90.5	95.83	92.5
R-BLS	**91**	**95.83**	**95**

2) EXPERIMENT ON MNIST DATASET

The MNIST handwriting dataset is composed of 60000 training instances and 10000 testing instances. Each instance is an image with one digit belonging to an integer set 0-9. As shown in Figure 8(b), each gray-level image with 28×28 pixels has the uniform background. The original image patches are transformed to a matrix in which each sample is a 784-dimensional vector, and one target label is a 10-dimensional one-hot vector. The settings of *hids* and *enhs* are (800,800,10000) and 10000, respectively. The testing results of different algorithms on MNIST are reported in Table 3. Compared with other methods, R-BLS leads to a promising better performance with 98.95% accuracy. It is mentionable that the R-BLS also performs better than cascaded BLS whose accuracy is 98.83%.

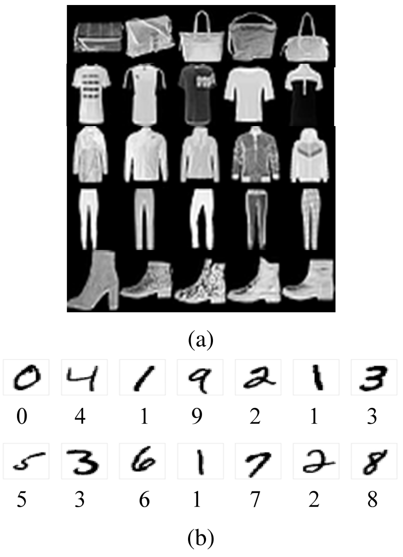


FIGURE 8. (a) is an illustration of Fashion-MNIST; (b) shows an illustration of MNIST.

TABLE 3. Results of different algorithms on MNIST.

Method	Accuracy (%)
DBN	98.75
SVM	98.6
RF	96.8
SAE	98.6
SDA	98.72
MLP	97.39
CNN	95.63
FRBM	97.44
BLS	98.74
R-BLS	**98.95**

True class

Hypothesis output	C_1	C_2
Y	TP True Positives	FP False Positives
N	FN False Negatives	TN True Negatives
Column counts:	P_C	N_C

FIGURE 9. The matrix of performance evaluation.

C. COMPARISON WITH OTHER IMBALANCED LEARNING METHODS

1) EVALUATION ON BENCHMARK DATASETS

In this study, the predicted labels are defined as $\{Y, N\}$ associated with the true labels $\{C_1, C_2\}$, a confusion matrix representing classification performance can be illustrated in Figure 9.

Based on the table above, true positives rate (TP_rate) and false positives rate (FP_rate) can be defined as:

$$TP\_rate = \frac{TP}{P_C}; \quad FP\_rate = \frac{FP}{N_C}. \tag{21}$$

In this section, we first conduct ablation experiments on some public imbalanced datasets which are available at UCI and KEEL [74]. The names and imbalance ratios (IR) are list as follows: MAGIC Gamma Telescope (magic, IR: 1.84), page_blocks0 (page, IR: 8.79), segment0 (segment, IR: 6.02), yeast1 (IR: 2.46), yeast3 (IR: 8.10), yeast4 (IR: 28.10). To give visual comparison, we report the results in *Receiver Operating Characteristics* (ROC) graphs that are formed by plotting true positives rate over false positives rate. The nearer a ROC curve is to the coordinate (1,0) in the graph, the better the predictions are. Meanwhile, the other measure is *Area Under Curve* (AUC). An AUC close to 1.0 indicates that the prediction is perfect. From Figure 10, a winner in all cases is the CR-BLS. Besides, the C-BLS achieves better performances than BLS on imbalanced datasets.

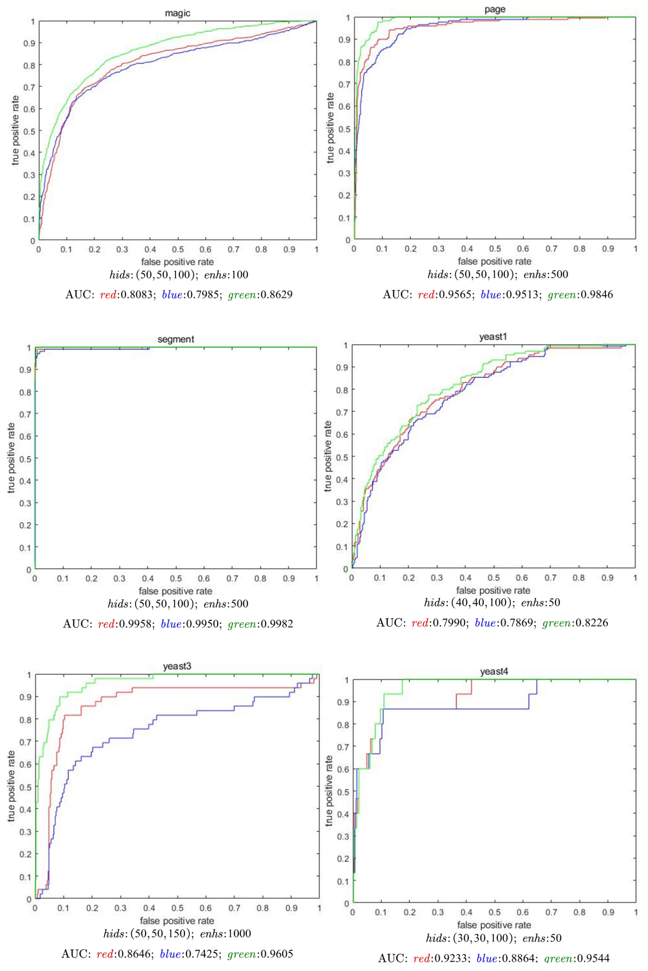


FIGURE 10. ROC graphs and AUC results on benchmark datasets. The red lines are results of C-BLS; blue lines are results of BLS; green lines are results of CR-BLS.

2) EVALUATION ON REAL-WORLD DATASETS

In this part, we focus on a practical application on automatic bug triaging system. With the great influx of attention concentrating on the system triaging bugs, the research of automated recognition of bug reports gradually becomes an overwhelming trend [75]–[77]. The entire steps of a bug triaging process are shown in Figure 11, in which

TABLE 4. The details of bug report datasets from Eclipse.

Product	Name		IR
Eclipse	1	CDT_cdt-core	4.0175
	2	JDT_Core	2.5684
	3	JDT_Debug	2.4261
	4	PDE_UI	2.6633
	5	Platform_Debug	1.7414
	6	Platform_SWT	4.9232
	7	Platform_UI	2.5422

TABLE 5. The details of bug report datasets from GNOME.

Product	Name		IR
GNOME	1	ekiga_general	9.5
	2	Evolution_Calendar	4.6262
	3	Evolution_Contacts	2.7764
	4	Evolution_Mailer	2.902
	5	Evolution_Shell	2.4444
	6	gnome-panel_Panel	3.9424
	7	gnome-terminal_general	12.6742

TABLE 6. The details of bug report datasets from Moizlla.

Product	Name		IR
Moizlla	1	Core_Printing	7.018
	2	Core_XPCOM	2.0201
	3	Core_XPConnect	5.3
	4	Core_XUL	4.0902
	5	Thunderbird_General	4.0511
	6	Core_Layout	2.8615
	7	Core_Security_UI	6.2672

classification algorithms aim to distinguish non-severe from severe bugs which a software developer must fix as soon as possible. In practice, bug report datasets with disproportionate number of category examples commonly hinder the classification learning. We address this by using cost-sensitive R-BLS which is compared with relevant imbalanced learning algorithms including random oversampling (ROS), random undersampling (RUS), the synthetic minority oversampling technique (SMOTE). For fair comparison, the experimental settings among different methods are strictly same.

Based on Figure 9, the *accuracy* can be defined as follows

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{22}$$

which is a simple way to describe the performance of a classifier. In addition, another popular metric is $F - measure$ defined as

$$F\!-\!Measure = \frac{(1 + \lambda^2) \cdot Recall \cdot Precision}{\lambda^2 \cdot Recall + Precision} \tag{23}$$

TABLE 7. Accuracies of different imbalanced learning methods on bug report datasets from Eclipse.

ACC	BLS	RUS-BLS	ROS-BLS	SMOTE-BLS	C-BLS	CR-BLS
Eclipse_1	0.7511	0.6567	0.7725	0.5107	0.7897	**0.7897**
Eclipse_2	0.6257	0.5478	0.6448	0.5642	0.6762	**0.7117**
Eclipse_3	0.6775	0.59	0.685	0.5875	0.7125	**0.7225**
Eclipse_4	0.6613	0.5469	0.6705	0.5881	0.6819	**0.7346**
Eclipse_5	0.6938	0.6434	0.6667	0.7016	0.7054	**0.7364**
Eclipse_6	0.8105	0.6947	0.8073	0.8057	0.8121	**0.8154**
Eclipse_7	0.7333	0.6685	0.7105	0.7201	**0.7357**	0.7339

TABLE 8. Weighted *F-measures* of different imbalanced learning methods on bug report datasets from Eclipse.

WF	BLS	RUS-BLS	ROS-BLS	SMOTE-BLS	C-BLS	CR-BLS
Eclipse_1	0.7038	0.6891	0.7056	0.5473	0.7153	**0.7153**
Eclipse_2	0.6382	0.5668	0.6371	0.586	0.6689	**0.6981**
Eclipse_3	0.6787	0.6084	0.6614	0.6056	0.6843	**0.6968**
Eclipse_4	0.6675	0.5677	0.6705	0.61	0.6695	**0.7146**
Eclipse_5	0.6934	0.6486	0.6044	0.6726	0.7047	**0.71**
Eclipse_6	**0.8004**	0.725	0.7836	0.7927	0.7946	0.7964
Eclipse_7	0.7224	0.6813	0.7067	0.7129	0.7241	**0.728**

TABLE 9. Accuracies of different imbalanced learning methods on bug report datasets from GNOME.

ACC	BLS	RUS-BLS	ROS-BLS	SMOTE-BLS	C-BLS	CR-BLS
GNOME_1	0.9163	0.8767	0.8965	0.9041	0.9193	**0.9285**
GNOME_2	0.844	0.7468	0.8028	0.8241	0.8496	**0.8532**
GNOME_3	0.7544	0.6858	0.7226	0.7216	0.7564	**0.7697**
GNOME_4	0.7839	0.7259	0.7364	0.7386	0.7779	**0.8014**
GNOME_5	0.805	0.7815	0.8138	0.7683	0.8152	**0.8226**
GNOME_6	0.8315	0.8132	0.807	0.7856	0.8453	**0.853**
GNOME_7	0.935	0.9337	0.935	0.8922	0.9496	**0.9509**

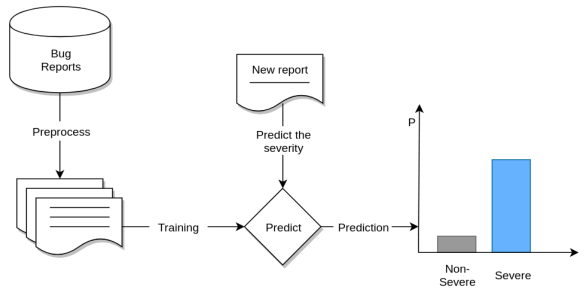


FIGURE 11. The entire steps of a bug triaging process.

where λ is a coefficient of importance, *Precision* and *Recall* respectively are

$$Precision = \frac{TP}{TP + FP} \tag{24}$$

$$Recall = \frac{TP}{TP + FN} \tag{25}$$

According to Equation (23), we can notice that the *F-measure* is a weighted combination of *Precision* and *Recall*, and provides insight into the functionality of a classification algorithm. Like most studies in this field [78], we use an effective measure, namely weighted *F-measure*, to act the second metric.

In this part, bug reports are from three major open-source projects, i.e. Eclipse [79], Moizlla [80] and Gnome [81]. Bugzilla is the common bug tracking system used by these projects. Seven components are selected from each project, and the details of those components are presented in Table 4-6 including the number of severe and non-severe cases, and the imbalance ratio.

As shown in Table 7-12, the RUS tends to perform badly on most datasets. This main reason is that the BLS is a data-driven model. The size of training dataset is cut down by RUS and becomes inadequate to train the data-driven models. This situation thus leads to underfitting causing bad performance of RUS. The other reason is that some potentially

TABLE 10. Weighted *F-measures* of different imbalanced learning methods on bug report datasets from GNOME.

WF	BLS	RUS-BLS	ROS-BLS	SMOTE-BLS	C-BLS	CR-BLS
GNOME_1	0.9103	0.8667	0.8645	0.8856	0.9202	**0.9292**
GNOME_2	0.8274	0.7749	0.8201	0.8316	0.8479	**0.8488**
GNOME_3	0.7319	0.704	0.7382	0.7283	0.7646	**0.7741**
GNOME_4	0.7656	0.7433	0.7529	0.7525	0.7876	**0.809**
GNOME_5	0.8005	0.7911	0.8206	0.7742	0.8201	**0.821**
GNOME_6	0.8211	0.8275	0.8175	0.792	0.8472	**0.8521**
GNOME_7	0.9176	0.9416	0.9398	0.9001	0.9481	**0.951**

TABLE 11. Accuracies of different imbalanced learning methods on bug report datasets from Moizlla.

ACC	BLS	RUS-BLS	ROS-BLS	SMOTE-BLS	C-BLS	CR-BLS
Moizlla_1	0.842	0.8197	0.829	0.8346	0.8643	**0.8662**
Moizlla_2	0.8347	0.6061	0.8099	0.8264	0.8347	**0.8375**
Moizlla_3	0.8137	0.6275	0.8431	0.8529	0.8529	**0.8529**
Moizlla_4	0.8095	0.5913	0.8095	0.7897	0.8095	**0.8175**
Moizlla_5	0.6274	0.4418	0.7107	0.6399	0.7107	**0.7877**
Moizlla_6	0.752	0.5957	0.7453	0.7554	**0.7601**	0.748
Moizlla_7	0.8614	0.4484	0.8466	0.8525	**0.8643**	0.8614

TABLE 12. Weighted *F-measures* of different imbalanced learning methods on bug report datasets from Moizlla.

WF	BLS	RUS-BLS	ROS-BLS	SMOTE-BLS	C-BLS	CR-BLS
Moizlla_1	0.8155	0.8203	0.8111	0.8164	0.8192	**0.8203**
Moizlla_2	0.7778	0.6547	0.7778	0.7919	**0.7978**	0.7937
Moizlla_3	0.7819	0.6757	0.788	0.8078	0.8078	**0.8078**
Moizlla_4	0.7593	0.6266	0.7495	0.7458	**0.7848**	0.7694
Moizlla_5	0.6629	0.4691	0.719	0.6739	0.7239	**0.7408**
Moizlla_6	0.7398	0.6191	0.7331	0.7441	0.7444	**0.7464**
Moizlla_7	0.8123	0.5144	0.8176	0.8246	0.8261	**0.8337**

crucial data may be deleted by RUS. On the contrary, the ROS has good performance on individual datasets, but the redundant data replicated by ROS increase the risk of overfitting. The SMOTE augments the size of training dataset by using synthetic samples, which tends to lead overlapping between two classes, and synthetic samples could not follow the distributive characteristics of minority class. The performances of SMOTE thus are not stable on most datasets. One can easily observe that the C-BLS and CR-BLS (i.e. cost-sensitive BLS and cost-sensitive R-BLS) have remarkable improvements against the others in most datasets, in terms of accuracy and weighted *F-measure*. Furthermore, the cost-sensitive BLS with *BFR* generally generate better results than C-BLS.

V. CONCLUSION

In this paper, we propose a new BLS framework named R-BLS, which achieves various level representations through group-wise encoding. Due to the broadly-fused feature representation, the proposed method improves the original BLS which just contains a monotonous mechanism to extract features from raw input data. In addition, a cost-sensitive method for BLSes is proposed to handle imbalanced learning problem. Extensive experiments have been conducted to verify the effectiveness of the R-BLS and CR-BLS. In these experiments, the proposed methods achieve more stable and better performance than other methods.

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[3] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao, "DehazeNet: An end-to-end system for single image haze removal," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5187–5198, Nov. 2016.

[4] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 694–711.

[5] Q. Yin, W. Zhang, Y. Zhang, and T. Liu. (2016). "A deep neural network for Chinese zero pronoun resolution." [Online]. Available: https://arxiv.org/abs/1604.05800

[6] B. Wang, K. Liu, and J. Zhao, "Conditional generative adversarial networks for commonsense machine comprehension," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 4123–4129.

[7] K. Madala, D. Gaither, R. Nielsen, and H. Do, "Automated identification of component state transition model elements from requirements," in *Proc. IEEE 25th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2017, pp. 386–392.

[8] A. Joulin and T. Mikolov, "Inferring algorithmic patterns with stack-augmented recurrent nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 190–198.

[9] X.-Z. Wang, Q.-Y. Shao, Q. Miao, and J.-H. Zhai, "Architecture selection for networks trained with extreme learning machine using localized generalization error model," *Neurocomputing*, vol. 102, pp. 3–9, Feb. 2013.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2016.

[11] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: Theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, May 1992.

[12] Y. Le Cun, L. Bottou, and Y. Bengio, "Reading checks with multilayer graph transformer networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 1, Apr. 1997, pp. 151–154.

[13] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.* Zürich, Switzerland: Springer, 2014, pp. 818–833.

[14] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[15] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[16] H. Yu, S. Cheng, B. Ni, M. Wang, J. Zhang, and X. Yang, "Fine-grained video captioning for sports narrative," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6006–6015.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[18] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," in *Proc. 9th Int. Conf. Artif. Neural Netw.*, 1999, pp. 850–855.

[19] K. Cho *et al.* (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." [Online]. Available: https://arxiv.org/abs/1406.1078

[20] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 652–663, Apr. 2017.

[21] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Proc. AAAI*, 2016, pp. 2741–2749.

[22] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification and recognition," in *Proc. Int. Conf. Artif. Neural Netw.* Warsaw, Poland: Springer, 2005, pp. 799–804.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[24] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Amsterdam, The Netherlands: Springer, 2016, pp. 630–645.

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[27] S. Ruder. (2016). "An overview of gradient descent optimization algorithms." [Online]. Available: https://arxiv.org/abs/1609.04747

[28] A. Ng, "Sparse autoencoder," Stanford Univ., Stanford, CA, USA, Lect. Notes CS294A, 2011.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[30] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Netw.*, vol. 1, no. 4, pp. 339–356, Oct. 1988.

[31] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," *Backpropagation, Theory, Archit., Appl.*, vol. 1, pp. 433–486, 1995.

[32] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5, pp. 602–610, 2005.

[33] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 369–376.

[34] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[35] A. Fischer and C. Igel, "Training restricted Boltzmann machines: An introduction," *Pattern Recognit.*, vol. 47, no. 1, pp. 25–39, 2014.

[36] C.-Y. Zhang, C. L. P. Chen, D. Chen, and K. T. Ng, "MapReduce based distributed learning algorithm for restricted Boltzmann machine," *Neurocomputing*, vol. 198, pp. 4–11, Jul. 2016.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[38] S. Ioffe and C. Szegedy. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167

[39] Y. Wu and K. He. (2018). "Group normalization." [Online]. Available: https://arxiv.org/abs/1803.08494

[40] 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. B, Pattern Recognit. Methodol. Syst.*, vol. 2, Aug./Sep. 1992, pp. 1–4.

[41] 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.

[42] 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, 2006.

[43] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[44] J. Zhai, S. Zhang, and C. Wang, "The classification of imbalanced large data sets based on mapreduce and ensemble of ELM classifiers," *Int. J. Mach. Learn. Cybern.*, vol. 8, no. 3, pp. 1009–1017, 2017.

[45] J.-H. Zhai, H.-Y. Xu, and X.-Z. Wang, "Dynamic ensemble extreme learning machine based on sample entropy," *Soft Comput.*, vol. 16, no. 9, pp. 1493–1502, 2012.

[46] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2015.

[47] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 31–34, Nov. 2013.

[48] X.-Z. Wang, T. Zhang, and R. Wang, "Noniterative deep learning: Incorporating restricted Boltzmann machine into multilayer random weight neural networks," *IEEE Trans. Syst., Man, Cybern. Syst.*, to be published, doi: 10.1109/TSMC.2017.2701419.

[49] C. L. P. Chen and Z. L. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[50] C. L. P. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2018.2866622.

[51] S. Feng and C. L. P. Chen, "Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2018.2857815.

[52] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2818–2826.

[53] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, vol. 4, 2017, p. 12.

[54] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, vol. 1, Jul. 2017, pp. 2261–2269.

[55] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, vol. 1, Jul. 2017, pp. 936–944.

[56] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1473–1480.

[57] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 2008.

[58] H.-Y. Ji, X.-Z. Wang, Y.-L. He, and W.-L. Li, "A study on relationships between heuristics and optimal cuts in decision tree induction," *Comput. Elect. Eng.*, vol. 40, no. 5, pp. 1429–1438, 2014.

[59] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *Proc. AAAI Workshop Learn. Text Categorization*, vol. 752, 1998, pp. 41–48.

[60] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[61] C. Zhang, K. C. Tan, H. Li, and G. S. Hong, "A cost-sensitive deep belief network for imbalanced classification," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2018.2832648.

[62] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[63] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[64] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

[65] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *Proc. Eur. Conf. Comput. Vis.* Amsterdam, The Netherlands: Springer, 2016, pp. 69–84.

[66] D. Ulyanov, A. Vedaldi, and V. Lempitsky. (2017). "Deep image prior." [Online]. Available: https://arxiv.org/abs/1711.10925

[67] W. Zong, G.-B. Huang, and L. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, Feb. 2013.

[68] G.-B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 18–29, May 2015.

[69] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[70] A. Asuncion and D. J. Newman, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, Irvine, CA, USA, Jan. 2010. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[71] H. Xiao, K. Rasul, and R. Vollgraf. (2017). "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms." [Online]. Available: https://arxiv.org/abs/1708.07747

[72] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[73] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. 2nd IEEE Workshop Appl. Comput. Vis.*, Dec. 1994, pp. 138–142.

[74] KEEL. Accessed: Feb. 2, 2018. [Online]. Available: http://sci2s.ugr.es/keel/imbalanced.php

[75] T. Zhang, J. Chen, G. Yang, B. Lee, and X. Luo, "Towards more accurate severity prediction and fixer recommendation of software bugs," *J. Syst. Softw.*, vol. 117, pp. 166–184, Jul. 2016.

[76] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in *Proc. 7th IEEE Work. Conf. Mining Softw. Repositories (MSR)*, May 2010, pp. 1–10.

[77] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing mining algorithms for predicting the severity of a reported bug," in *Proc. 15th Eur. Conf. Softw. Maintenance Reeng. (CSMR)*, Mar. 2011, pp. 249–258.

[78] Y. Zhou, Y. Tong, R. Gu, and H. Gall, "Combining text mining and data mining for bug report classification," *J. Softw., Evol. Process*, vol. 28, no. 3, pp. 150–176, 2016.

[79] Eclipse. Accessed: Feb. 2, 2018. [Online]. Available: http://bugs.eclipse.org/bugs

[80] Mozilla. Accessed: Feb. 2, 2018. [Online]. Available: http://bugzilla.mozilla.org

[81] GNOME. Accessed: Feb. 2, 2018. [Online]. Available: http://bugzilla.gnome.org

[82] X.-Z. Wang, H.-J. Xing, Y. Li, Q. Hua, C.-R. Dong, and W. Pedrycz, "A study on relationship between generalization abilities and fuzziness of base classifiers in ensemble learning," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 5, pp. 1638–1654, Oct. 2015.

[83] T. Zhang and X. Yang. (2018). "G-SMOTE: A GMM-based synthetic minority oversampling technique for imbalanced learning." [Online]. Available: https://arxiv.org/abs/1810.10363

[84] W. Deng, H. Zhao, X. Yang, J. Xiong, M. Sun, and B. Li, "Study on an improved adaptive PSO algorithm for solving multi-objective gate assignment," *Appl. Soft Comput.*, vol. 59, pp. 288–302, Oct. 2017.

[85] W. Deng, H. Zhao, L. Zou, G. Li, X. Yang, and D. Wu, "A novel collaborative optimization algorithm in solving complex optimization problems," *Soft Comput.*, vol. 21, no. 15, pp. 4387–4398, 2017.

[86] W. Deng, R. Yao, H. Zhao, X. Yang, and G. Li, "A novel intelligent diagnosis method using optimal LS-SVM with improved PSO algorithm," *Soft Comput.*, to be published, doi: 10.1007/s00500-017-2940-9.

[87] W. Deng, S. Zhang, H. Zhao, and X. Yang, "A novel fault diagnosis method based on integrating empirical wavelet transform and fuzzy entropy for motor bearing," *IEEE Access*, vol. 6, pp. 35042–35056, 2018.

[88] H. Zhao, R. Yao, L. Xu, Y. Yuan, G. Li, and W. Deng, "Study on a novel fault damage degree identification method using high-order differential mathematical morphology gradient spectrum entropy," *Entropy*, vol. 20, no. 9, p. 682, 2018.

[89] H. M. Zhao, M. Sun, W. Deng, and X. Yang, "A new feature extraction method based on EEMD and multi-scale fuzzy entropy for motor bearing," *Entropy*, vol. 19, no. 1, p. 14, 2017.

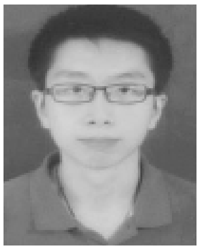
[90] S. Guo, R. Chen, M. Wei, H. Li, and Y. Liu, "Ensemble data reduction techniques and multi-RSMOTE via fuzzy integral for bug report classification," *IEEE Access*, vol. 6, pp. 45934–45950, 2018.

[91] S. Guo, R. Chen, H. Li, J. Gao, and Y. Liu, "Crowdsourced Web application testing under real-time constraints," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 28, no. 6, pp. 751–779, 2018.

[92] J. Zhai, L. Zang, and Z. Zhou, "Ensemble dropout extreme learning machine via fuzzy integral for data classification," *Neurocomputing*, vol. 275, pp. 1043–1052, Jan. 2018.

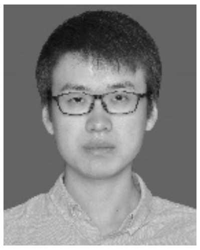
[93] J. Zhai, X. Wang, and X. Pang, "Voting-based instance selection from large data sets with mapreduce and random weight networks," *Inf. Sci.*, vols. 367–368, pp. 1066–1077, Nov. 2016.

[94] J. Zhai, T. Li, and X. Wang, "A cross-selection instance algorithm," *J. Intell. Fuzzy Syst.*, vol. 30, no. 2, pp. 717–728, 2016.



TIAN-LUN ZHANG received the B.Sc. degree in information management and information system and the M.Sc. degree in software engineering. He is currently pursuing the Ph.D. degree in computer science and technology with the Information Science and Technology College, Dalian Maritime University, Dalian, China. His current research interests include fuzzy measures and integrals, computer vision, and imbalance learning from big data.

RONG CHEN received the M.S. and Ph.D. degrees in computer software and theory from Jilin University, China, in 1997 and 2000, respectively. He is currently a Professor with the College of Information Science and Technology, Dalian Maritime University, and has previously held positions at Sun Yat-sen University, China. His research interests include software diagnosis, collective intelligence, activity recognition, the Internet, and mobile computing.



XI YANG received the B.S. degree in software engineering from the Information Science and Technology College, Qingdao University of Science and Technology, Qingdao, China. He is currently pursuing the M.S. degree in computer science and technology with the Information Science and Technology College, Dalian Maritime University, Dalian, China. His research interests include computer vision, deep learning, fuzzy measures and integrals, and imbalance learning from big data.

SHIKAI GUO received the B.Sc. degree in computer science from the Information Science and Technology College, Dalian Maritime University, Dalian, China, in 2012, where he is currently pursuing the Ph.D. degree in computer science and technology. His research interests include mining software repositories, search-based software engineering, fuzzy measures and integrals, and imbalance learning from big data.