Research Frontier

JongHyok Ri
Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, China
Institute of Information Technology, Kim Il Song University, Pyongyang, DPR of Korea

Liang Liu
Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, China

Yong Liu
State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, China

Huifeng Wu
College of Computer Science and Technology, Zhejiang Dianzi University, Zhejiang, China

Wenliang Huang
China Unicom Ltd. Beijing, China

Hun Kim
Department of Computer Science, Kim Il Song University, DPR of Korea

Optimal Weighted Extreme Learning Machine for Imbalanced Learning with Differential Evolution

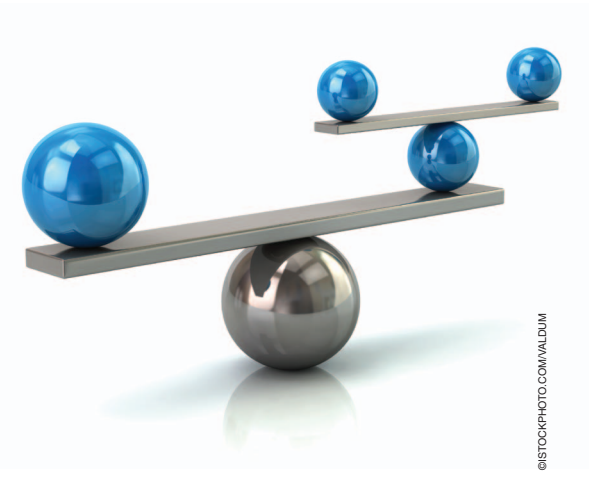
Abstract

In this paper, we present a formal model for the optimal weighted extreme learning machine (ELM) on imbalanced learning. Our model regards the optimal weighted ELM as an optimization problem to find the best weight matrix. We propose an approximate search algorithm, named weighted ELM with differential evolution (DE), that is a competitive stochastic search technique, to solve the optimization problem of the proposed formal imbalanced learning model. We perform experiments on standard imbalanced classification datasets which consist of 39 binary datasets and 3 multiclass datasets. The results show a significant performance improvement over standard ELM with an average *Gmean* improvement of 10.15% on binary datasets and 1.48% on multiclass datasets, which are also better than other state-of-the-art methods. We also demonstrate that our proposed algorithm can achieve high accuracy in representation learning by performing experiments on MNIST, CIFAR-10, and YouTube-8M, with feature representation from convolutional neural networks.

I. Introduction

Extreme learning machine (ELM) [1]–[4], is an effective and efficient machine learning technique that has attracted attention in various fields. The essential advantage of ELM is that the hidden neuron parameters are randomly assigned which may be independent of training data and the output weights can be analytically decided by the Moore-Penrose generalized inverse [5], [6]. Thus, it provides simpler and faster implementation than other machine learning techniques.

In recent years, the imbalanced learning problem [7]–[10] has drawn a significant amount of interest from academia, industry, and government funding agencies as data continues to accumulate. As most of the standard learning algorithms assume the distributions among all classes are equal, the equal misclassification costs are acceptable on algorithms learning balanced datasets. However, when dealing with a complex imbalanced dataset, standard algorithms [2]–[4] will fail to represent the distribution characteristics of the data and thus will lead to unfavorable accuracies across the data classes [11]. Unfortunately, classic ELM does not solve the problem of imbalanced data



Digital Object Identifier 10.1109/MCI.2018.2840707
Date of publication: 18 July 2018

Corresponding Author: Yong Liu (Email: yongliu@iipc.zju.edu.cn).

1556-603X/18©2018IEEE

distribution. When using classic ELM for imbalanced data, the majority class tends to push the separating boundary toward the minority side to gain better classification results for itself. Therefore data in the minority class will be easily misclassified.

The most straightforward solution for imbalanced data learning is to assign the misclassification cost inversely to the class distribution, which may be simply calculated as the number of samples in each class. Thus Zong et al. [7] proposed weighted ELM to overcome the disadvantages of the original ELM for an imbalanced data problem. Their solution is based on the work regarding weighted regularized ELM presented by Toh [9] and Deng et al. [10], and the key essence of weighted ELM in Zong et al. [7] is to assign an extra weight to each sample to strengthen the impact of the minority class while weakening the relative impact of the majority class. Experimental results in their work showed superior performance of weighted ELM compared with original ELM on various imbalanced datasets. However, the two weighting schemes used in their approach can only obtain empirical sub-optima, and global optima cannot be guaranteed. Following work [8] introduced the boosting method to obtain better weighting schemes; however, how to set the optimal weighting scheme remains an open problem.

In this paper, we present a formal model for optimal weighted ELM applied to imbalanced learning. Our model assumes that the optimal weighted ELM can be presented as an optimization problem to search the optimal weight matrix for the weighted ELM. We also present an approximate solution for the optimal weight matrix searching problem based on differential evolution (DE) [12], which is a competitive stochastic search technique that performs well in various standard test functions and real-world optimization problems. We also evaluated the effectiveness of our learning machine algorithm by performing experiments on various standard classification datasets, which consist of 39 binary datasets and 3 multiclass datasets: the MNIST [13], CIFAR-10 [14], and YouTube-8M [15] datasets.

The contributions of our approach are as follows,

- ❏ We present a formal mathematical model to obtain the optimal weighting scheme. We introduce DE to calculate the approximate optimal solution.
- ❏ Our approach can achieve significant improvement in classification performance compared with other state-of-the-art methods on various imbalanced datasets.
- ❏ Our approach can narrow the search range greatly compared with other state-of-the-art methods, which indicates our approach may be more efficient for the practical imbalanced learning problem.

The remaining sections are organized as follows. Sections II and III introduce the theoretical background of ELM and related ELM methods on imbalanced learning. Section IV presents the proposed method. Section V reports the experimental results and performance analysis. Finally, the conclusions are summarized in Section VI.

II. Theoretical Background

This section introduces a brief theoretical background of ELM.

ELM [1], [2] was originally proposed for single-hidden layer feedforward neural networks (SLFNs) and then extended to 'generalized' SLFNs where the hidden layer does not require tuning [3].

The main feature of ELM is the random generation of hidden nodes which may be independent of the training data and the analytical calculation of output weights by the Moore-Penrose generalized inverse [3]. The hidden layer output (with l nodes) can be presented by a row vector $h(x) = [h_1(x), ..., h_l(x)]$ where x is the input sample. Given n training samples (x_i, t_i), the mathematical model of the SLFNs is

$$H\beta = T. \tag{1}$$

where H is the hidden layer output matrix, β is the output weight and T is the target vector.

The least squares solution with minimal norm is analytically determined using the Moore-Penrose "generalized" inverse [3], [16] as follows,

$$\begin{aligned} n < l &: \beta = H^{\dagger}T = H^T\left(\frac{I}{C} + HH^T\right)^{-1}T \\ l < n &: \beta = H^{\dagger}T = \left(\frac{I}{C} + H^TH\right)^{-1}H^TT. \end{aligned} \tag{2}$$

ELM can also be explained from the optimization view. ELM tries to minimize both $\|H\beta - T\|^2$ and $\|\beta\|^2$. Therefore, a solution for formula (1) can be obtained [2] from

$$\begin{aligned} &\text{Minimize}: L_{\text{ELM}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{n}\|\xi_i\|^2 \\ &\text{Subject to}: h(x_i)\beta = t_i^T - \xi_i^T, i = 1, ..., n. \end{aligned} \tag{3}$$

where $\xi_i = [\xi_{i,1}, ..., \xi_{i,m}]$ is the training error vector of the m output nodes corresponding to training sample x_i. C is the trade-off regularization parameter between the minimization of training errors and the maximization of the marginal distance. Based on the Karush-Kuhn-Tucker (KKT) theorem [16], we can solve the optimization problem of formula (3) and obtain the same solution as formula (2).

Given a new sample x, the output function of ELM is obtained as follows.

$$f(x) = \begin{cases} h(x)H^T\left(\frac{I}{C} + HH^T\right)^{-1}T, & \text{when } n < l \\ h(x)\left(\frac{I}{C} + H^TH\right)^{-1}H^TT, & \text{when } n \geq l. \end{cases} \tag{4}$$

Here, $f(x) = [f_1(x), ..., f_m(x)]$ is the output function vector. Users may determine the predication label of x as follows.

$$label(x) = \arg\max_i f_i(x), \ i \in [1, ..., m]. \tag{5}$$

Recently, further focus has been placed on ELM algorithms and applications [17]–[19], and some advanced algorithms [20]–[23] have been proposed to improve performance. In [20], a Bayesian-based ELM (BELM), which incorporates the advantages of both ELM and Bayesian models, is proposed. It can build the corresponding confidence

***Gmean* is a conventional evaluation metric in the case of imbalanced learning; it is the geometric mean of the recall values of all *m* classes.**

interval without using additional methods such as bootstrap, and requires low computational cost.

Bai et al. [21] proposed a sparse ELM (S-ELM) by replacing the equality constraints in traditional ELM model with inequality constraints, which can reduce the storage space and testing time. Bai et al. [22] also proposed a S-ELM for regression analysis that can reduce the storage space and testing time. In addition, they have developed an efficient training algorithm based on iterative computation, which scales quadratically with respect to the number of training samples. In [23], a random projection-based ELM (RP-ELM), which is estimated on the analysis of the random projection feature mapping schema in the ELM, is proposed. RP-ELM can significantly reduce the number of neurons in the hidden layer without affecting the accuracy of the generalization performance. As a result, the final learning machine will benefit from a considerable simplification in the feature-mapping stage.

III. Related ELM Methods on Imbalanced Learning

A. Weighted ELM [7]

The main goal of the ELM classifier is to find a boundary to separate data from two or multiple parts with maximal margin distance between any two parts. This separating boundary is supposed to be pushed toward the side of the minority class for imbalanced data so that the minority classes are easy misclassified. To resolve this issue, weighted ELM (WELM) [7] has been recently proposed.

WELM improves the classification performance for data with imbalanced class distribution while maintaining the advantages of the original ELM stated above. Specifically, each training sample is assigned with an extra weight. Mathematically, an $n \times n$ diagonal matrix W associated with every training sample x_i is defined. Usually, if x_i comes from a minority class, the associated weight w_{ii} is relatively larger than samples from a majority class. Therefore, the impact of the minority class is strengthened while the relative impact of the majority class is weakened. Considering the diagonal weight matrix W, the optimization formula of ELM can be revised [7] as

$$\begin{aligned} &\text{Minimize}: \\ &L_{\text{WELM}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{n}(w_{ii}\|\xi_i\|^2) \\ &\text{Subject to}: \\ &h(x_i)\beta = t_i^T - \xi_i^T, i = 1, \ldots, n. \end{aligned} \tag{6}$$

According to the KKT theorem [16], the solution to formula (6) is

$$\beta = H^{\dagger}T = \begin{cases} H^T\left(\frac{I}{C} + WHH^T\right)^{-1}WT, \text{when } n < l \\ \left(\frac{I}{C} + H^TWH\right)^{-1}H^TWT, \text{when } n \geq l. \end{cases} \tag{7}$$

They also proposed two empirical weighting schemes [7] as follows,

$$\begin{aligned} W_1&: \quad w_{ii} = \frac{1}{\#t_i} \\ W_2&: \quad w_{ii} = \begin{cases} \frac{0.618}{\#t_i} \text{ if } \#t_i > AVG(\#t_i) \\ \frac{1}{\#t_i} \text{ if } \#t_i \leq AVG(\#t_i). \end{cases} \end{aligned} \tag{8}$$

where $\#t_i$ is the number of samples belonging to class t_i, and $AVG(\#t_i)$ represents the average number of samples for all classes.

Although WELM proposes two weighting schemes in formula (8) for imbalanced data, these matrices are not optimal, which can be proven by the following toy experiment. In the toy experiment, we first randomly generate an imbalanced dataset containing two categories with a category ratio of 10:1, and we also apply the standard ELM, WELM with W_1, WELM with W_2 and two other WELMs with randomly selected weights. The original dataset and the classification results are shown in Fig. 1, where the ground truth labels of every data instance are denoted by different symbols and the classification results are denoted by different colors. The results in Fig. 1 show that although WELM with W_1 and W_2 policies can achieve better results than the ELM method in this dataset, their performances are worse than the two WELMs with randomly selected weights, which indicates the policies with W_1 and W_2 in WELM are not optimal. In this paper, these two WELM classifiers are referred to as WELM-W1 and WELM-W2.

WELM [7] can improve the performance of ELM greatly for imbalanced data. The classification error of the class with fewer elements was reduced by setting an unequal cost distribution for each class with the weight matrix. However, the approach relied on empirical weighting schemes, which were designed according to the element number of each class. Thus, it can also be called experienced WELM.

B. Boosting Weighted ELM [8]

To overcome this shortcoming of experienced WELM, boosting-based WELM (BWELM) [8] was proposed. BWELM tries to determine the optimal weight matrix using an AdaBoost algorithm [24].

Inspired by the distribution weights updating mechanism of AdaBoost, they embedded WELM seamlessly into a modified AdaBoost framework. Intuitively, the distribution weights in AdaBoost, which reflect the importance of training samples, are input into WELM as training sample weights. Furthermore, such training sample weights are dynamically updated during iterations of AdaBoost.

Considering the characteristics of imbalanced learning, they modify the original AdaBoost.M1 [24] in two aspects. First, the initial distribution weights are set to be asymmetric to make AdaBoost converge at a faster speed.

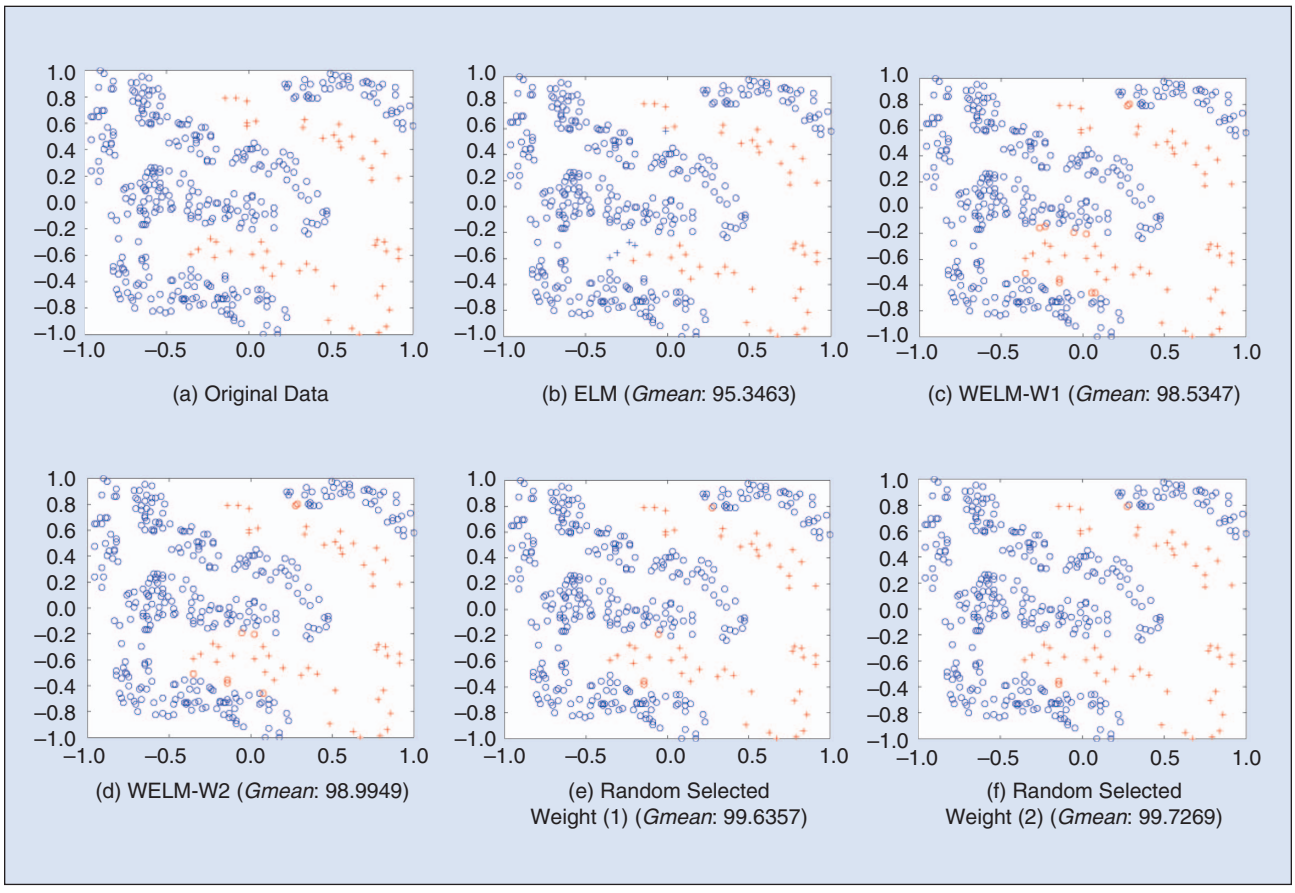


FIGURE 1 Classification results produced by ELM, WELM-W1, WELM-W2, and WELM with random selected weights on a randomly-generated dataset with an imbalanced ratio of 1:10. Blue circles and red circles represent the correctly classified instances and incorrectly classified instances in the majority class, respectively. Red crosses and blue crosses represent the correctly classified instances and incorrectly classified instances in the minority class, respectively.

This results in a boosting classifier with a smaller number of WELM classifiers and can save much computational time. Second, the distribution weights are updated separately for different classes to avoid destroying the distribution weights' asymmetry.

In this paper, we also address the problem of obtaining the best weighted matrix for the ELM-based imbalanced learning problem, and we propose a DE-based WELM (DE-WELM) for imbalanced dataset learning.

DE was first presented by R. Storn and Price [12], and it has been recognized as a powerful method for solving optimization problems. It resembles the structure of evolutionary algorithms, but differs from their traditional versions in the generation of new candidate solutions and the use of a greedy selection scheme. As shown in previous research [25], DE is far more efficient and robust (with respect to reproducing the results in several runs) compared to other evolutionary computation algorithms such as particle swarm optimization (PSO) [26] and evolutionary programming (EP) [27]. In addition, it has few parameters to set, and the same settings can be adapted to many different problems. Thus, this method has been actively used in various fields [28]–[30]. Some works [31]–[33] have focused on updating ELM using DE. However, they only use DE to search the hidden neuron parameters instead of searching the weighted matrix for imbalanced learning.

As parameters such as *NP*, *F*, and *CR* in the DE algorithm are critical for its performance, there are also many improved DE algorithms [34]–[40], and [41] which can adaptively control those parameters.

IV. Our Approach

Although the two weighting schemes in formula (8) can obtain superior results compared with the original ELM, they are only empirical schemes and cannot guarantee the optima, which was proven in the toy experiment. In this section, we present the formal mathematical model to obtain the optimal weighting scheme, and we also introduce the DE method [12] to calculate the approximate optimal weight matrix used in WELM, as the calculation for the formal model is infeasible.

A. Mathematical Model for Optimal Weighted Scheme

The problem of calculating the optimal weight matrix in WELM can be formally defined as follows, Given training data $\Omega = \{(x_i, t_i), t_i \in \{1, 2, ..., m\}, i = 1, 2, ..., n\}$, and there are m classes in Ω, the

We compare the performance of our proposed method with four state-of-the-art methods, i.e., the Softmax classifier [43], the classic ELM classifier [1], the experienced WELM classifier [7], and the BWELM classifier [8].

optimal weight matrix can be obtained by optimizing the following formula,

$$W^* = \arg\min_{W} (1 - Gmean(H\beta(\Omega, W), T))$$
Subject to:
$$\beta(\Omega, W) = \begin{cases} H^T\left(\frac{I}{C} + WHH^T\right)^{-1} WT, \text{when } n < l \\ \left(\frac{I}{C} + H^T WH\right)^{-1} H^T WT, \text{when } n \geq l. \end{cases} \quad (9)$$

where H is the hidden layer output matrix, T is the target vector, and l is the number of hidden layer nodes in WELM. As β is the output weight computed in WELM using W as the weight matrix, this formula can be considered as the function of W and Ω. $Gmean$ is a conventional evaluation metric in the case of imbalanced learning; it is the geometric mean of the recall values of all m classes and is defined as follows,

$$Gmean(Y, T) = \left[\prod_{j=1}^{m} \frac{q_j}{p_j}\right]^{\frac{1}{m}} \quad (10)$$

where Y is the class prediction vector, q_j is the number of elements belonging in class j correctly classified among Y, and p_j is the number of elements belonging in class j among T. Although we may use the Levenberg Marquardt (LM) algorithm [42] to search for the optimal solution to this problem, the results are quite sensitive to the initial values, so we introduce the DE algorithm to calculate the approximate optimal solution.

B. DE-based Approximate Weight Calculation Algorithm for Weighted ELM

DE is a method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality; it has been successfully applied in many domains [28]–[30]. There are three major control factors in the DE algorithm, i.e., population size (NP), scaling factor (F), and crossover rate (CR). There are also three operations involved, i.e., mutation operation, crossover operation, and selection operation. We present our approximate weight calculation algorithm based on the three factors and three operations in DE.

There are two stages in our algorithm: the initial stage and the update stage.

1) Initial Stage

The initial stage generates an initial population that contains NP candidates in the weight matrix W, and searches for the best weight from the current candidate set.

First, the values in training set Ω are normalized to [-1,1], and we also select a subset $\Omega_{vd} \subset \Omega$ as the validation set, which is used to avoid overfitting.

The initial population contains NP candidate weight matrixes, $NP - 2$ candidates are generated randomly in [0,1], and the remaining two candidates are set as W_1 and W_2, respectively, which are presented in WELM [7]. Thus the initial population W_R consists of NP n-dimensional vectors as follows,

$$W_R = \{W_{i,R} \mid W_{i,R} = (w_{i,R}^1, w_{i,R}^2, ..., w_{i,R}^n), i = 1, 2, ..., NP - 2\} \cup \{W_1, W_2\}. \quad (11)$$

Note that the weight matrix in WELM is a diagonal matrix, so all elements are zero except for the diagonal elements. Therefore we will express a diagonal matrix using a vector.

Second, we transform $W_{i,R} (i = 1, 2, ..., NP)$ into a weight matrix $diag(W_{i,R})$[1] and compute the error $E_{W_{i,R}} (i = 1, ..., NP)$ of the WELM classifier on validation set Ω_{vd}, and the candidate optimization weight vector which produces the lowest error is defined as $W_{best,R}$.

$$E_{W_{i,R}} = 1 - Gmean(H\beta(\Omega_{vd}, diag(W_{i,R})), T)$$
$$W_{\text{best},R} = \arg\min_{W_{i,R}} E_{W_{i,R}} (i = 1, 2, ..., NP). \quad (12)$$

2) Update Stage

The update stage will generate a new candidate population based on the results of the previous iteration and performs the operations of mutation, crossover, and selection.

The input of the update stage is a candidate population of the previous stage and the candidate optimal weight vector from the previous stage, and the output of the update stage is a candidate population of the next stage and a candidate optimal weight vector for the next stage.

First, we transform candidate population W_R to mutant population V_R with the mutation operation. Each mutant vector of V_R is calculated according to the following equation.

$$V_R = \{V_{i,R} \mid V_{i,R} = W_{i,R} + F(W_{\text{best},R} - W_{i,R}) + F(W_{r_1^i,R} - W_{r_2^i,R}), i = 1, 2, ..., NP\}. \quad (13)$$

The indices r_1^i and r_2^i are mutually exclusive integers randomly generated within the range $[1, NP]$, which are also different from the index i. Indices are randomly generated once for each mutant vector. The scaling factor F is a positive control parameter for scaling the differences among vectors.

After the mutation operation, we apply the crossover operation to each pair of target vector $W_{i,R}$ and its corresponding mutant vector $V_{i,R}$ to generate a trial vector $U_{i,R} = (u_{i,R}^1, u_{i,R}^2, ..., u_{i,R}^n)$ as follows,

[1] $diag(\cdot)$ is the function to map input vectors to a diagonal matrix as diagonal elements and all elements are zero except for the diagonal elements.

$$u_{i,R}^{j} = \begin{cases} v_{i,R}^{j} & \text{if } (\text{rand}_j[0,1) \le CR) \text{ or } (j = j_{\text{rand}}) \\ w_{i,R}^{j} & \text{otherwise} \end{cases} \quad (j = 1,2,...,n, \quad i = 1,2,...,NP). \tag{14}$$

Here if the values of the newly generated trial vectors exceed the corresponding upper and lower bounds, they will be re-normalized into the range $[0,1]$.

CR is defined by users as a constant within the range $[0,1]$, which controls the fraction of parameter values copied from the mutant vector. j_{rand} is a randomly chosen integer in the range $[1,n]$. This operation copies the jth parameter of the mutant vector $V_{i,R}$ to the corresponding element in trial vector $U_{i,R}$ if $(\text{rand}_j[0,1) \le CR)$ or $(j = j_{\text{rand}})$. Otherwise, the parameter is copied from the corresponding target vector $W_{i,R}$. The condition $j = j_{\text{rand}}$ is introduced to ensure that the trial vector $U_{i,R}$ will be different from its corresponding target vector $W_{i,R}$ by at least one parameter.

We then use $diag(U_{i,R})$ $(i = 1, 2, ..., NP)$ as a weight matrix to compute the error of the WELM classifier on validation set Ω_{vd}. The element which has the lowest error is defined to $U_{\text{best},R}$. We can calculate the error of $U_{i,R}$ as follows.

$$E_{U_{i,R}} = 1 - Gmean(H\beta(\Omega_{vd}, diag(U_{i,R})), T) \quad (i = 1,...,NP). \tag{15}$$

Then the next candidates population $W_{R+1} = \{W_{i,R+1}, i = 1,2,...,NP\}$ and its best candidate can be decided according to the following select operation.

$$W_{i,R+1} = \begin{cases} U_{i,R} & \text{if } E_{U_{i,R}} \le E_{W_{i,R}} \\ W_{i,R} & \text{otherwise} \end{cases}$$
$$W_{best,R+1} = \begin{cases} W_{\text{best},R} & \text{if } E_{W_{\text{best},R}} < E_{U_{\text{best},R}} \\ U_{\text{best},R} & \text{otherwise.} \end{cases} \tag{16}$$

We compare the $E_{U_{i,R}}$ of each trial vector $U_{i,R}$ with the $E_{W_{i,R}}$ of its corresponding target vector in the current population. If the trial vector has less or equal error value to the corresponding target vector, the trial vector will replace the target vector and enter the next candidate population. Otherwise, the target vector will remain in the population for the next generation.

The update stage will be executed iteratively several times, and we obtain the approximate optimal weight matrix $diag(W_{\text{best},R_{\max}})$ once the algorithm completes, where $R_{\max}$ is the number of iterations in the DE algorithm.

In short, our method obtains the initial population and the candidate weight vectors in the initial stage and calculates an approximate optimized matrix by updating the population using the DE algorithm in the update stage. The pseudo-code of our proposed algorithm is given in Algorithm 1.

V. Performance Evaluation

In this section, we conduct comparison experiments to evaluate the classification capability of our DE-WELM on imbalanced learning problems and the analyses on the results are also presented. We compare the performance of our proposed method with four state-of-the-art methods, i.e., the Softmax classifier [43], the classic ELM classifier [1], the experienced WELM classifier [7], and the BWELM classifier [8].

We select a subset of the KEEL (Knowledge Extraction based on Evolutionary Learning) imbalanced dataset repository[2] to evaluate the methods. We also use the datasets of MNIST [13] (handwritten digit image dataset), the CIFAR-10 [14] (tiny image dataset), and the YouTube-8M database in our experiments. The results are averaged over ten runs. Brief descriptions of the datasets are provided in the following subsections.

In our experiments, we also introduce the imbalanced ratio (IR) [7] to quantitatively measure the imbalanced degree of a dataset.

$$\text{Binary}: IR = \frac{\#(+1)}{\#(-1)}$$
$$\text{Multiclass}: IR = \frac{\min(\#(t_i))}{\max(\#(t_i))}, \quad i = 1,...,m. \tag{17}$$

Above, $\#(+1)$ is the number of samples in a minority class, $\#(-1)$ is the number of samples in the majority class, and $\#(t_i)$ is the number of samples in class t_i for a multiclass dataset.

The attributes of all the datasets are normalized to [–1,1].

In ELM theory, there are many choices for feature mapping. In our experiments, we use the sigmoid additive based feature mapping function [4], which is a popular choice for researchers. There are two parameters

[2] http://sci2s.ugr.es/keel/study.php?cod=24

Algorithm 1 DE-WELM.

Input: Training set $\Omega = \{(x_i, t_i), t_i \in \{1,2,...,m\}, i = 1, 2,...,n\}$, $R_{\max}$.
Output: Approximate optimal weight matrix W^*.

1: All data values x_i of training set Ω are normalized in [–1,1], and then select validation set Ω_{vd}. Set $R = 0$.
2: Generate initial population $W_R = \{W_{i,R}, i = 1,2,...,NP\}$ by formula (11).
3: Compute the error $E_{W_{i,R}}(i = 1,...,NP)$ of WELM classifier with weight matrix $diag(W_{i,R})$ on validation set Ω_{vd} and the candidate optimal weight vector $W_{\text{best},R}$ by formula (12).
4: **while** $R < R_{\max}$ **do**
5: Compute the trial vectors $U_{i,R}(i = 1,...,NP)$ from W_R by formulas (13) and (14).
6: Compute the error $E_{U_{i,R}}(i = 1,...,NP)$ of WELM classifier with weight matrix $diag(U_{i,R})$ on validation set Ω_{vd} by formula (15).
7: Obtain next stage population W_{R+1} and a candidate optimal weight vector of the next stage $W_{\text{best},R+1}$ by formula (16).
8: $R = R + 1$.
9: **end while**
10: $W^* = diag(W_{\text{best},R_{\max}})$.

TABLE 1 Specification of binary classification problems.

DATASETS	# ATTRI	# TRAIN	# TEST	*IR*
YEAST05679VS4	8	422	106	0.1047
YEAST1458VS7	8	554	139	0.0453
YEAST1289VS7	8	385	97	0.0327
YEAST1VS7	7	367	92	0.0700
ECOLI1	7	268	68	0.2947
ECOLI2	7	268	68	0.1806
ECOLI3	7	268	68	0.1167
ECOLI4	7	268	68	0.0635
YEAST2VS4	8	411	103	0.1078
YEAST2VS8	8	385	97	0.0434
GLASS0123VS456	9	171	43	0.3053
GLASS016VS2	9	153	39	0.0929
GLASS016VS5	9	147	37	0.0500
PIMA	8	614	154	0.5350
YEAST1	8	1187	297	0.4064
YEAST3	8	1187	297	0.1230
YEAST4	8	1187	297	0.0349
YEAST5	8	1187	297	0.0304
SHUTTLEC0VSC4	9	1463	366	0.0726
SHUTTLEC2VSC4	9	103	26	0.0404
SEGMENT0	19	1846	462	0.1661
WISCONSIN	9	546	137	0.5380
HABERMAN	3	244	62	0.3556
IRIS0	4	120	30	0.5000
VOWEL0	13	790	198	0.1002
NEW-THYROID1	5	172	43	0.1944
NEW-THYROID2	5	172	43	0.1944
PAGE-BLOCKS1	10	377	95	0.0620
GLASS0	9	173	43	0.4786
GLASS1	9	171	43	0.5405
GLASS2	9	171	43	0.0823
GLASS4	9	171	43	0.0621
GLASS5	9	171	43	0.0427
GLASS6	9	171	43	0.1554
VEHICLE0	18	676	170	0.3075
VEHICLE1	18	676	170	0.3439
VEHICLE2	18	676	170	0.3466
VEHICLE3	18	676	170	0.3330
YEAST6	8	1187	297	0.0243

to tune for ELM, i.e., the trade-off constant C and the number of hidden nodes l. In our experiments, these two parameters for DE-WELM are set as follows,

$$C \in \{2^{-6}, 2^{-4}, 2^{-2}, 2^{0}, 2^{2}, 2^{4}, 2^{6}\}$$
$$l \in \{10, 110, 210, ..., 810, 910\}. \quad (18)$$

We set the control parameters to $NP = 50, F = 1, CR = 0.8$ in our approach. The number of iterations $R_{\max}$ is set to 10. The validation set is randomly assigned as a subset with half the elements of the training set.

In the experiments involving ELM, experienced WELM [7], and BWELM, the grid search ranges for C and l are conducted over $\{2^{-18}, 2^{-16}, ..., 2^{48}, 2^{50}\}$ and $\{10, 20, ..., 990, 1000\}$, respectively.

The search range of λ for the Softmax classifier is set as $\{-8, -7, -6, ..., 1, 2, 3\}$.

To compare our algorithm with other computational methods, we also introduce the approximate weight search method based on PSO [44], [45]. Particle swarm optimization is one of the most popular nature-inspired metaheuristic optimization algorithms, developed by Kennedy and Eberhart in 1995 [44], [45]. Since its development, many variants have also been developed for solving practical issues related to optimization [46]–[48]. In our paper, we use the PSO proposed in [26].

In the following experiments involving the PSO method, two acceleration parameters c_1 and c_2 are set to 2, the inertia factor w is set to $[0.4, 0.9]$, and the maximal iteration is set to 50. The size of the initial population (swarm) that is the initial set of candidate weights is set to 50, and W_1, W_2, which are presented in [7] are included in the initial population (swarm) and the remaining elements are randomly generated between [0,1]. The grid search ranges for C and l are also conducted over $\{2^{-18}, 2^{-16}, ..., 2^{48}, 2^{50}\}$ and $\{10, 20, ..., 990, 1000\}$, respectively. We call this algorithm "PSO-WELM" in the following sections.

A. Dataset Specification

1) Standard Classification Datasets

Similar to the data specifications mentioned in the experienced WELM [7] and BWELM [8], we have selected 39 binary datasets and 3 multiclass datasets from the KEEL dataset repository which have different degrees of imbalance. Details of the datasets used in our experiments are shown in Tables 1 and 2, where the number of attributes

(#ATTRI), the number of classes (#CLASS), the number of training samples (#TRAIN), the number of test samples (#TEST), and *IR* are listed. The *IR* of the binary datasets selected from the KEEL dataset varies from 0.0243 to 0.5405, and the *IR* of the multiclass datasets varies from 0.0061 to 0.5882.

2) MNIST Dataset [13]

The MNIST dataset (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning [13], [49]–[52]. The MNIST dataset contains 60,000 training images and 10,000 test images with each image labeled by an integer, and 7,000 images per class. Each example is a 28 × 28 single channel grayscale image. Example images of the MNIST dataset are shown in Fig. 2(a).

TABLE 2 Specification of multiclass classification problems.

DATASETS	# ATTRI	# CLASS	# TRAIN	# TEST	*IR*
HAYES-ROTH	4	3	105	27	0.5882
NEW-THYROID	5	3	172	43	0.2066
PAGE-BLOCKS	10	5	438	110	0.0061

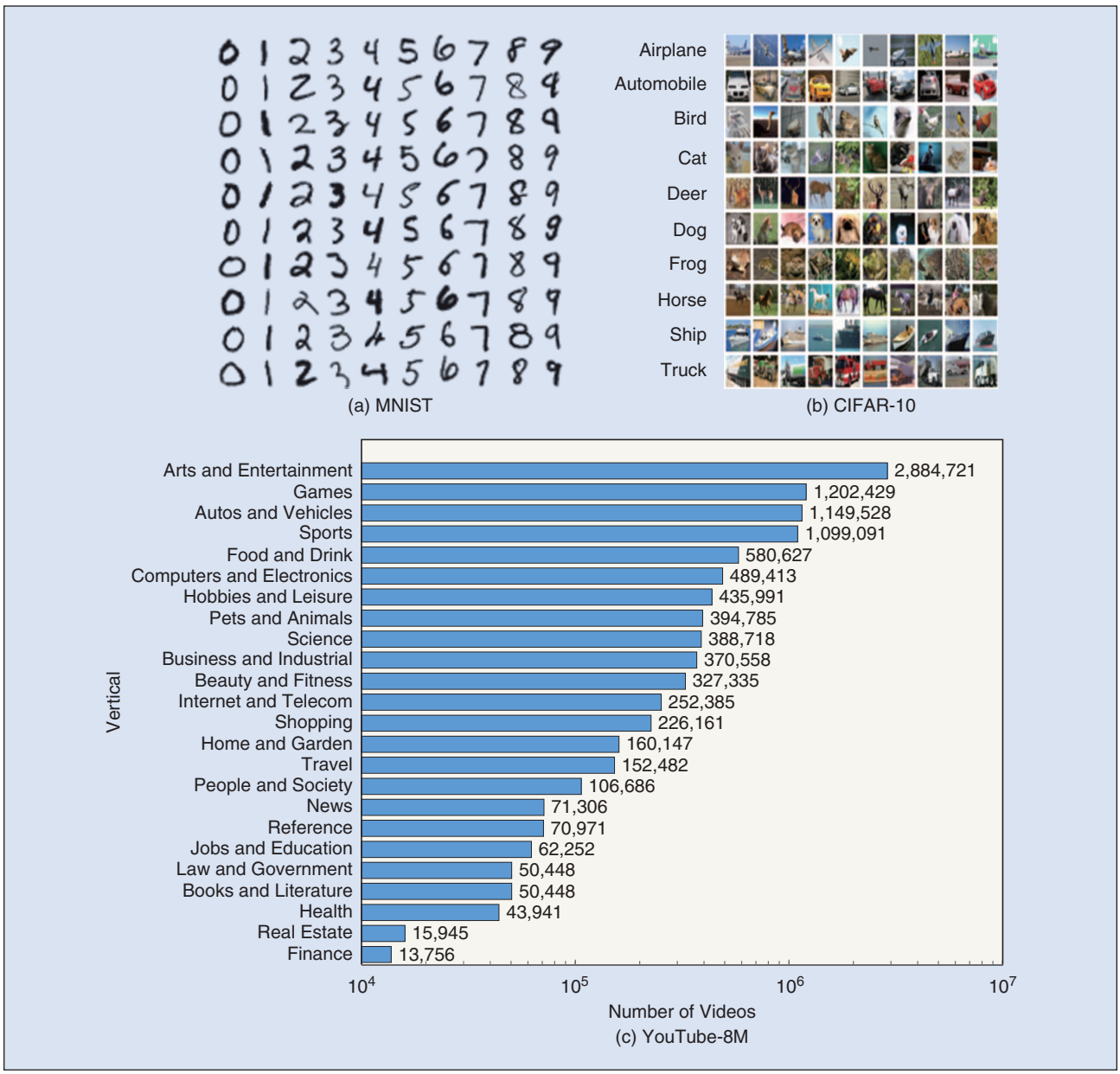


FIGURE 2 Example digital images of MNIST database and CIFAR-10 database, and number histograms of top training videos in YouTube-8M.

TABLE 3 Experimental results of binary problem.

DATASETS	*GMEAN* (SIGMOID MODE)						
	SOFTMAX (%)	ELM (%)	WELM-W1 (%)	WELM-W2 (%)	BWELM (%)	PSO-WELM (%)	DE-WELM (%)
YEAST05679VS4	62.58	84.13	86.02	80.02	85.56	85.69	**86.53**
YEAST1458VS7	45.12	61.07	67.10	64.26	68.09	68.55	**72.18**
YEAST1289VS7	46.52	59.23	75.83	73.92	74.21	78.48	**90.16**
YEAST1VS7	49.72	74.70	76.65	81.36	86.23	**88.24**	85.45
ECOLI1	86.40	87.77	90.69	90.26	93.02	90.80	**95.13**
ECOLI2	91.82	89.44	90.19	88.64	90.80	94.50	**95.04**
ECOLI3	73.70	77.38	90.17	90.00	93.21	92.84	**93.61**
ECOLI4	85.15	91.96	97.83	95.90	**100**	**100**	99.19
YEAST2VS4	85.28	86.25	91.56	90.02	93.78	97.75	**98.01**
YEAST2VS8	81.63	72.83	75.56	76.01	81.32	80.32	**81.65**
GLASS0123VS456	97.10	95.34	95.66	95.55	**100**	**100**	**100**
GLASS016VS2	49.28	67.78	83.77	83.06	92.85	94.11	**95.26**
GLASS016VS5	100	92.41	98.55	98.70	**100**	**100**	**100**
PIMA	74.32	70.10	74.74	71.51	75.79	74.97	**78.06**
YEAST1	54.19	63.26	72.57	70.32	73.36	73.37	**77.06**
YEAST3	73.42	80.75	93.25	91.08	92.52	93.92	**94.06**
YEAST4	40.54	65.52	87.92	73.92	83.51	**93.05**	93.01
YEAST5	61.02	81.04	95.39	95.15	98.25	98.79	**99.04**
SHUTTLEC0VSC4	**100**	**100**	**100**	**100**	**100**	**100**	**100**
SHUTTLEC2VSC4	**100**	93.54	**100**	**100**	**100**	**100**	**100**
SEGMENT0	**100**	99.24	99.75	99.70	99.78	**100**	**100**
WISCONSIN	95.58	97.85	97.62	96.95	97.34	98.25	**98.48**
HABERMAN	23.57	49.16	65.11	59.26	73.74	77.24	**78.95**
IRIS0	**100**	**100**	**100**	**100**	**100**	**100**	**100**
VOWEL0	85.58	**100**	**100**	**100**	**100**	**100**	**100**
NEW-THYROID1	98.64	98.24	99.44	99.72	**100**	**100**	**100**
NEW-THYROID2	100	95.55	99.72	**100**	**100**	**100**	**100**
PAGE-BLOCKS1	97.78	99.09	99.43	99.42	**100**	**100**	**100**
GLASS0	74.65	78.51	80.29	81.35	86.66	**88.98**	87.18
GLASS1	38.29	78.79	78.96	80.87	79.54	84.04	**86.09**
GLASS2	57.01	90.61	91.33	88.34	90.15	93.07	**94.08**
GLASS4	69.79	85.72	91.34	91.46	96.45	96.74	**97.33**
GLASS5	97.47	90.81	95.99	96.60	**100**	**100**	**100**
GLASS6	89.44	94.96	95.72	95.90	**100**	**100**	**100**
VEHICLE0	95.97	98.51	99.12	99.09	96.71	**100**	99.62
VEHICLE1	69.54	84.14	85.21	82.75	75.98	87.23	**87.47**
VEHICLE2	95.99	98.43	99.12	98.78	97.31	**100**	**100**
VEHICLE3	60.92	78.15	85.13	84.34	85.37	86.52	**87.37**
YEAST6	70.59	70.77	87.77	88.29	**90.68**	89.67	90.02
AVERAGE	76.37	84.18	89.60	88.52	91.07	92.49	**93.33**

3) CIFAR-10 Dataset [14]

The CIFAR-10 dataset is a labeled subset of the 80 Million Tiny Images dataset[3], containing 60,000 color images (32 × 32) in 10 classes, with 6,000 images per class. They are split into 50,000 training images and 10,000 test images. Example images of the CIFAR-10 database are shown in Fig. 2(b).

4) YouTube Dataset

The YouTube-8M dataset [15] contains approximately 8 million YouTube videos. Each video is annotated with one or multiple tags for a total of 4,716 classes. All 4,716 classes can be grouped into 24 top-level vertical categories. In Fig. 2(c), we show histograms with the number of training videos in each top-level vertical (or top-level) category.

B. Experimental Results on Standard Classification Datasets

The experimental results of seven methods (Softmax, ELM, WELM-W1, WELM-W2, BWELM, PSO-WELM, and DE-WELM) under 39 binary imbalanced datasets and 3 multiclass imbalanced datasets are given in Tables 3 and 4, respectively. The results in both tables show that our DE-WELM can perform much better than all the other state-of-the-art methods, and the evaluation metric *Gmean* can improve by 30% compared to the classic ELM classifier. The average performance metrics also indicate significant improvement using our approach.

1) Comparison with Softmax Classifier

As can be seen in Tables 3 and 4, our DE-based algorithm can achieve satisfactory performance for all datasets compared with the Softmax classifier. Specifically, on the datasets *YEAST-1289VS7*, *HABERMAN*, *YEAST4*, *GLASS1*, and *GLASS016VS2*, DE-WELM can achieve an improvement of more than 40% on *Gmean*. There are six datasets where both classifiers achieve perfect 100% *Gmean* values. The Softmax classifier can perform much better than other algorithms except for our DE-WELM on the datasets of *ECOLI2* and *GLASS-0123VS456*, which also indicates that WELM-W1, WELM-W2, and BWELM are not optimal.

2) Comparison with Experienced Weighted ELM

Based on the results in Table 3, our DE-WELM can perform much better in all datasets compared with experienced WELM, especially on the datasets *YEAST1289VS7*, *YEAST4*, and *GLASS016VS2*, achieving an improvement of more than 10% on *Gmean*. Regarding the balanced datasets such as *GLASS1*, *WISCONSIN*, *PIMA*, etc., the experimental results of DE-WELM are also better than experienced WELM, which shows that our method is applicable to not only imbalanced datasets, but also to balanced datasets. It also indicates that the two experienced weighting schemes employed in [7] are not optimal.

3) Comparison with Boosting Weighted ELM

As shown in Table 3, DE-WELM can achieve superior performance on 39 datasets (92.85%) versus BWELM except for *ECOLI4*, *YEAST6*, and *YEAST1VS7*. There are 11 datasets (26.19%) where both DE-WELM and BWELM achieve perfect 100% *Gmean* values, which indicates that optimal weights are reached for those datasets. We further focus on the three datasets where DE-WELM performs worse than BWELM. The *Gmean* values of our approach are all slightly lower (less than 1%) than BWELM. The *Gmean* values of our approach on other datasets are better than the values of BWELM, where the *Gmean* values in some datasets, such as *VEHICLE1*, *YEAST4*, and *YEAST1289VS7*, are 10% higher than the values of BWELM. The overall comparable results indicate that DE-WELM potentially represents a superior solution as an optimal WELM classifier compared with BWELM. This conclusion is further supported by the results for average *Gmean* values.

4) Multiclass Imbalanced Learning

The experimental results on multiclass data are shown in Table 4, the results also show that DE-WELM can achieve much better performance compared with other methods except on *NEW-THYROID*, where the *Gmean* value of our method is 0.04% lower than the value of BWELM. The results in Table 4 indicate that our DE-WELM potentially represents the best solution as an optimal WELM classifier in multiclass imbalanced datasets.

5) Analysis on the Search Ranges

In ELM related methods, the best parameters for the trade-off constant C and the number of hidden nodes l are normally obtained by a grid search policy, which pre-sets search ranges for these two parameters and exhaustively searches all possible combinations of these two parameters. Obviously, the computation efficiencies are negatively related to the search ranges, while the probabilities to obtain the optimal solution are positively related to the search ranges.

In the beginning of our experiment section, we mentioned that the settings

TABLE 4 Experimental results of multiclass problems.

DATASETS	*GMEAN* (SIGMOID MODE)						
	SOFTMAX (%)	ELM (%)	WELM-W1 (%)	WELM-W2 (%)	BWELM (%)	PSO-WELM (%)	DE-WELM (%)
HAYES-ROTH	42.51	67.94	69.65	63.52	71.65	71.20	**72.20**
NEW-THYROID	80.69	93.11	92.83	92.58	93.15	**93.51**	93.11
PAGE-BLOCKS	88.46	94.41	89.71	94.33	93.52	92.85	**94.60**
AVERAGE	70.55	85.15	84.06	83.47	86.10	85.85	**86.63**

[3]http://groups.csail.mit.edu/vision/TinyImages/

of C and l in our DE-WELM are based over $\{2^{-6}, 2^{-4}, 2^{-2}, 2^{0}, 2^{2}, 2^{4}, 2^{6}\}$ and $\{10, 110, 210, \ldots, 810, 910\}$, respectively, and the settings of C and l in other methods are $\{2^{-18}, 2^{-16}, \ldots, 2^{48}, 2^{50}\}$ and $\{10, 20, \ldots, 990, 1000\}$, respectively. The experimental results in Tables 3 and 4 indicate that DE-WELM uses a smaller search range compared to other state-of-the-art methods, and can achieve much better performance compared to other state-of-the-art methods. Thus, our DE-WELM represents a more efficient optimal WELM classifier.

6) Analysis of the Convergence on Iterations

In this section, we will further evaluate the convergence on iteration of our DE-WELM. We carry out two experiments on six datasets, i.e., *WISCONSIN*, *HABERMAN*, *GLASS5*, *SEGMENT0*, *GLASS0*, and *PIMA*. We use two sets of C and l parameters for the two experiments and plot the training *Gmean* values after every iteration of the DE algorithm. The results are shown in Fig. 3. As shown in Fig. 3, a minimum of 9 iterations are required for the training process to converge in *WISCONSIN*, 10 in *PIMA*, 6 in *GLASS5*, 7 in *SEGMENT0*, 10 in *HABERMAN*, and 10 in *GLASS0*.

The results in Fig. 3 indicate that although the two sets of parameters are quite different, DE-WELM will quickly converge to the optimum within 10 iterations on all the test datasets, which indicates DE-WELM can easily converge toward the optimal solution. Therefore, we set the number of iterations to $R_{\max} = 10$ in the previous experiments.

7) Comparison with Other Computational Method Based WELM

As shown in Tables 3 and 4, our DE-WELM can achieve superior performance to PSO-WELM on 36 datasets (85.71%) except for *ECOLI4*, *YEAST4*, *GLASS0*, *VEHICLE0*, *YEAST1VS7*, and *NEW-THYROID*. Among all datasets, there are 13 datasets (30.95%) where both methods achieve perfect 100% *Gmean* values. Comparing with PSO-WELM, there are six datasets (14.29%) (*YEAST1458VS7*, *YEAST1289VS7*, *ECOLI1*, *PIMA*, *YEAST1*, *GLASS1*) where DE-WELM achieves an improvement of over 2% on *Gmean*, 11 datasets (26.19%) (*YEAST05679VS4*, *ECOLI2*, *ECOLI3*, *YEAST2VS8*, *GLASS016VS2*, *HABERMAN*, and so on) where DE-WELM can achieve an improvement greater than 0.5% on *Gmean*, and six datasets (14.29%) where DE-WELM achieves an improvement of more than 0.14% on *Gmean*. DE-WELM achieves an improvement above 0.84% regarding average *Gmean* for binary datasets, and an improvement greater than 0.78% on average *Gmean* for multiclass datasets compared with PSO-WELM.

8) Temporal Complexity Analysis

Regarding the temporal complexity, assuming that the temporal computation cost of a single WELM on the given parameters of C, l is denoted as t_{WELM}, then the temporal complexities of WELM based methods can be expressed

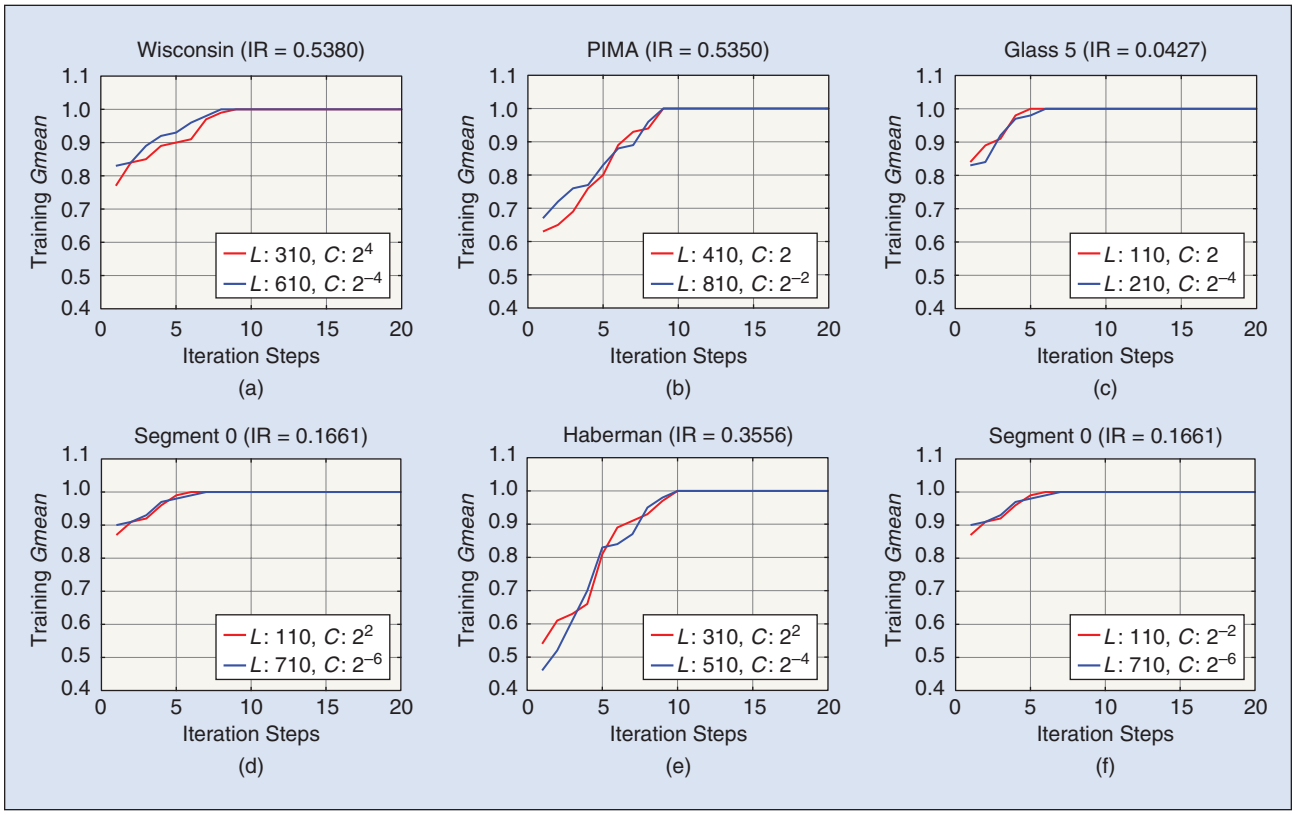


FIGURE 3 *Gmean* values after every iteration of our DE-based method on six standard classification datasets.

as the search numbers multiplied by the basic temporal computation cost of a single ELM as follows,

$$T = t_{\text{WELM}} \times (P_{\text{size}} \times I_{\max} \times C_{\text{number}} \times L_{\text{number}}) \quad (19)$$

where P_{size} is the size of the candidate population, $I_{\max}$ is the number of iterations, C_{number} is the number of possible trade-off constants C, and L_{number} is the number of possible hidden nodes l.

Considering the above equation and our experiment settings, the temporal complexity of our algorithm is $t_{WELM} \times 50 \times 10 \times 7 \times 10 = t_{WELM} \times 35,000$ and the temporal complexity of PSO-WELM is $t_{WELM} \times 50 \times 100 \times 35 \times 100 = t_{WELM} \times 17,500,000$. Considering that the initial candidate weight of BWELM is one and requires 10 iterations, the temporal complexity of BWELM is $t_{WELM} \times 1 \times 10 \times 35 \times 100 = t_{WELM} \times 35,000$. That is to say, our DE-WELM is 500 times faster than PSO-WELM and is equal to BWELM in terms of temporal complexity.

As mentioned for BWELM in [8] and considering the very fast learning speed of ELM, such costs for DE-WELM are quite acceptable.

9) Analysis of the Convergence on Different Initializations

Both our DE-WELM and PSO-WELM employ candidate weight populations that are randomly generated at the beginning of the algorithm. In the case of BWELM, the initial weights are determined directly by the data. In this section, we will further evaluate the convergence on different initial candidate weight populations. We set the same initial population, which is generated randomly, for the former two methods and perform comparative experiments on two datasets (*ECOLI1*, *GLASS2*). The experiments were carried out ten times with different initializations. At the same time, corresponding experiments on BWELM were also performed. The results can be seen in Fig. 4.

In Fig. 4, the horizontal axis indicates the sequence number of the different initial candidate weight populations and the vertical axis denotes the resulting *Gmean* values obtained in each experiment. Among 10 experiments for *ECOLI1*, DE-WELM achieved better classification performance than PSO-WELM in 9 experiments (90%), and in 7 experiments (70%) compared to BWLEM. Among the 10 experiments for *GLASS2*, DE-WELM achieved better classification performance than PSO-WELM in 8 experiments (80%), and in 9 experiments (90%) compared to BWLEM.

C. Experimental Results on MNIST Dataset

We introduce the LeNet-5 model to evaluate the performance of DE-WELM versus three state-of-the-art methods (classic ELM classifier, experienced WELM classifier, and BWELM) on the MNIST dataset. The LeNet-5 model [13] is a classic convolutional neural network (CNN) architecture proposed by LeCun et al. in 1998, which was applied in handwritten digit character recognition.

We perform experiments with the LeNet-5 model regarding the first few layers of the network as feature extractors to evaluate DE-WELM.

We first arbitrarily select three datasets that do not overlap and balance among the 70,000 MNIST datasets (60,000 training sets, 10,000 test sets). These three datasets are referred to as RES_DATA (representation training set), C_TRAIN (classifier training set), and C_TEST (classifier test set). The details of the selected datasets are provideds in Table 5,

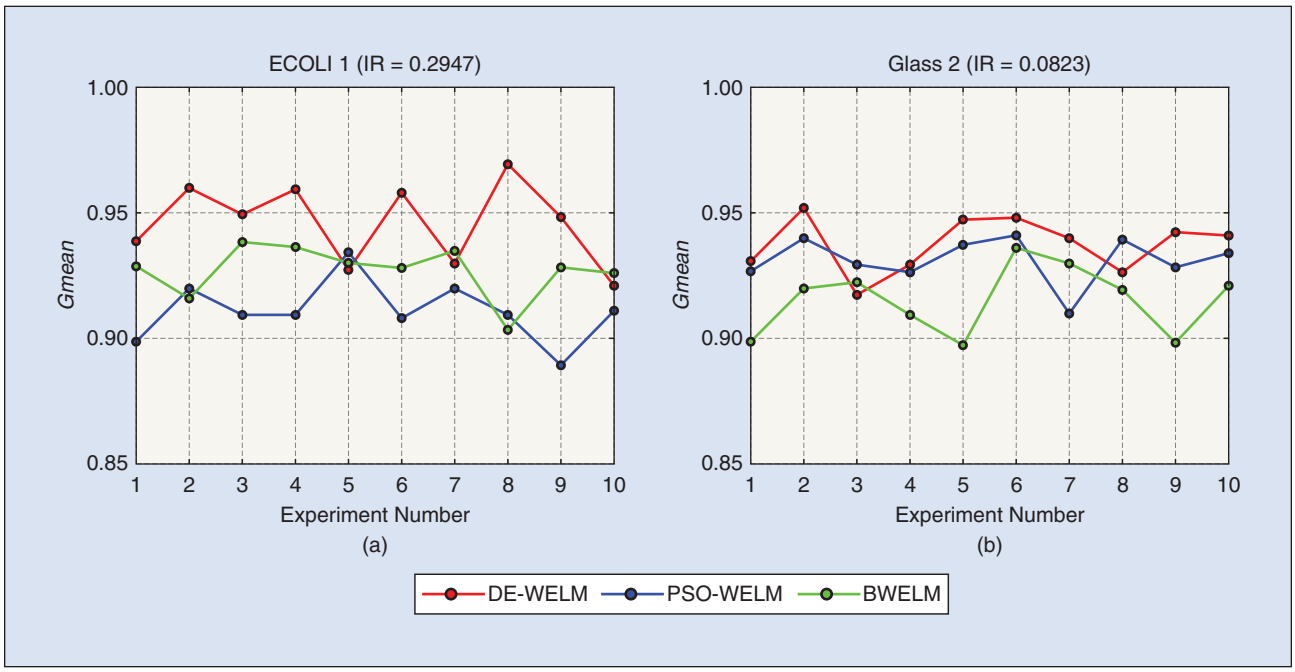


FIGURE 4 Experimental results of the convergence on initialization of weight on two datasets (*ECOLI1*, *GLASS2*).

We perform experiments with the LeNet-5 model regarding the first few layers of the network as feature extractors to evaluate DE-WELM.

where the number of attributes (#ATTRI), the number of classes (#CLASS), the number of selected data (#SAMPLE) and, *IR* are listed.

Then we train the LeNet-5 model on Res_Data by a popular deep learning framework known as Caffe [53]. The parameters we used for training the CNN are the default values in Caffe.

The LeNet-5 model outputs feature images of original image data which have 500 dimensions in its first fully connected layer. We regard the output of the LeNet-5 model's first fully connected layer on C_TRAIN as the training set to evaluate our classifier algorithm, and we take the output of LeNet-5 model's first fully connected layer on C_TEST as the test set to evaluate our DE-WELM. We also evaluate the classification of LeNet-5 using the above two data to compare with DE-WELM.

The experimental results of six methods (LeNet-5, ELM, WELM-W1, WELM-W2, BWELM, and DE-WELM) under selected datasets are given in Table 7. The parameters used in the experiment were the same as those used for the standard classification problem datasets. Table 7 indicates that DE-WELM achieves better performance than the other methods.

D. Experimental Results on CIFAR-10 Dataset

We introduce the CifarNet model [14] to evaluate the performance of our DE-WELM with four state-of-the-art methods (ELM, WELM-W1, WELM-W2, and BWELM) on CIFAR-10. Proposed by Alex Krizhevsky, CifarNet is the state-of-the-art method for object classification on the CIFAR-10 dataset. It has three convolution layers and three pooling layers for feature extraction, and a fully connected layer on top for classification. We use the simplest implementation to train a CifarNet model as a baseline model without any preprocessing such as image translations or transformations.

We perform experiments on CifarNet regarding the first few layers of the network as feature extractors to evaluate our classifier algorithm.

We first arbitrarily select three datasets that do not overlap and balance (among 60,000 images from the CIFAR-10 dataset: 50,000 for the training set, 10,000 for the test set), as RES_DATA (representation training set), C_TRAIN (classifier training set) and C_TEST (classifier test set). The details of the selected datasets are shown in Table 6, where the number of attributes (#ATTRI), the number of classes (#CLASS), the number of selected data (#SAMPLE), and *IR* are listed.

We train CifarNet on RES_DATA using Caffe and its default parameters [53].

We extract and reshape the output of the "Pool3" layer, which has 1024 dimensions, and use the extracted features of the CIFAR-10 dataset to evaluate five classifiers (ELM, WELM-W1, WELM-W2, BWELM, and DE-WELM). We regard the output of CifarNet model's "Pool3" layer on C_TRAIN as the training set, and the output of CifarNet model's "Pool3" layer on C_TEST as the test set to evaluate DE-WELM. We also evaluated classification by the CifarNet model using the above two datasets to compare with our DE-WELM.

The experimental results for six methods (CifarNet, ELM, WELM-W1, WELM-W2, BWELM, and DE-WELM) under selected imbalanced datasets are given in Table 7. The parameters in the experiment were the same as previous experiments. As shown in Table 7, DE-WELM achieves better performance than other methods except for CifarNet.

TABLE 5 Selected imbalanced datasets from MNIST.

DATASETS	# ATTRI	# CLASS	# SAMPLE	*IR*
RES_DATA	500	10	13470	0.0848
C_TRAIN	500	10	3329	0.0872
C_TEST	500	10	4991	0.0872

TABLE 6 Selected imbalanced datasets from CIFAR-10.

DATASETS	# ATTRI	# CLASS	# SAMPLE	*IR*
RES_DATA	1024	10	12351	0.0839
C_TRAIN	1024	10	3121	0.0918
C_TEST	1024	10	4679	0.0918

TABLE 7 Experimental results on MNIST and CIFAR-10.

DATABASES	GMEAN (SIGMOID MODE)						
	LENET-5 (%)	CIFARNET (%)	ELM (%)	WELM-W1 (%)	WELM-W2 (%)	BWELM (%)	DE-WELM (%)
MNIST	96.82		95.57	97.58	97.5	97.48	**97.72**
CIFAR-10		**58.84**	37.33	53.94	55.73	55.54	57.25

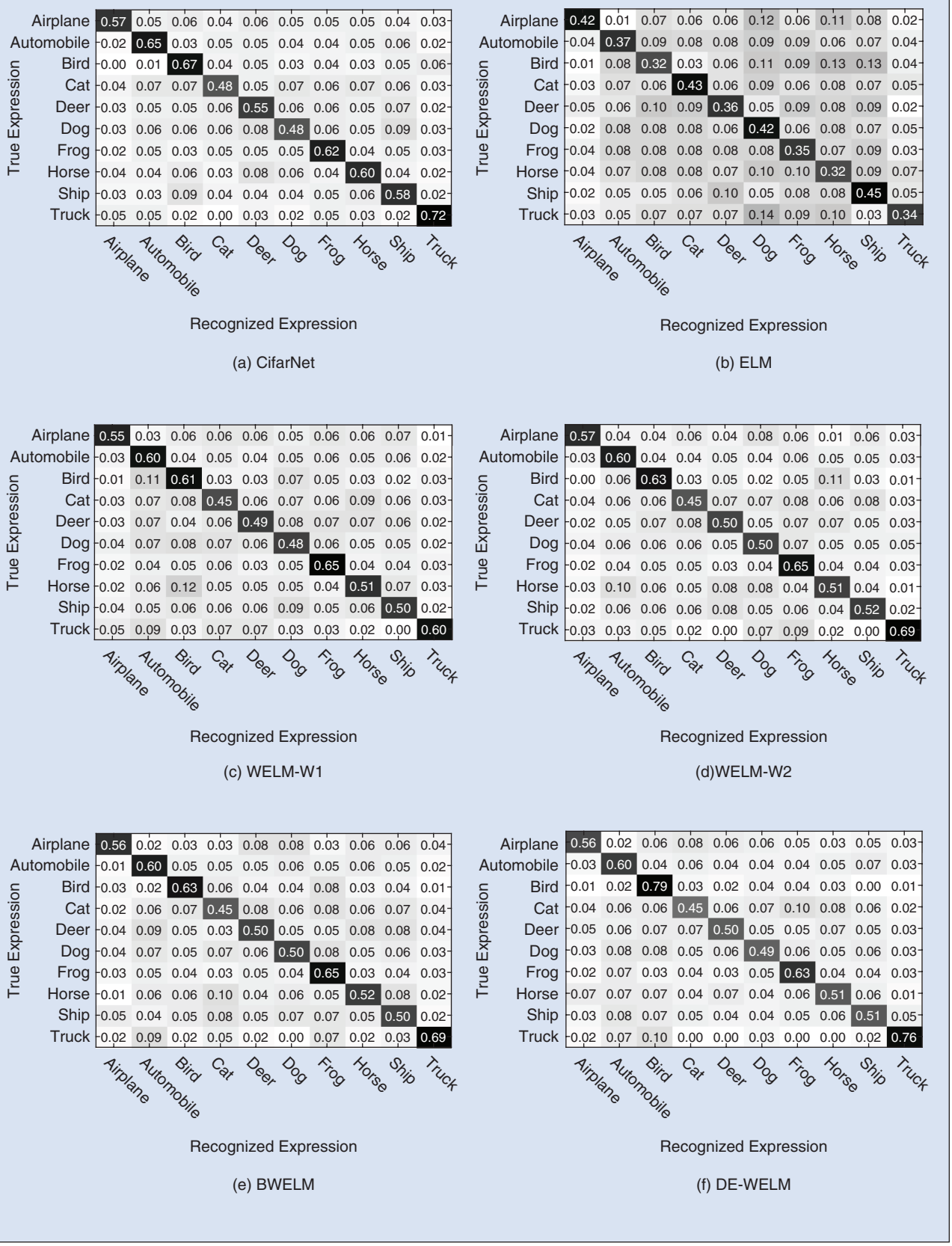


FIGURE 5 Confusion matrices obtained by applying six algorithms (CifarNet, ELM, WELM-W1, WELM-W2, BWELM, DE-WELM) on CIFAR-10 dataset.

TABLE 8 Selected imbalanced datasets from YouTube.

DATASETS	# ATTRI	# CLASS	# SAMPLE	*IR*
TRAINDATA	1024	24	10534	0.0018
TESTDATA	1024	24	5267	0.0018

TABLE 9 Experimental results on YouTube dataset.

EXPERIMENT INDEX	WELM-W1	WELM-W2	BWELM	PSO-WELM	DE-WELM
GMEAN(%)	47.78	47.40	46.25	47.84	**49.51**
TOTAL RUNNING TIME(S)	2287.57	2236.45	23301.35	3535679.05	24805.23

The *Gmean* value of our approach is 1.59% lower than CifarNet with the original Softmax classifier. We can argue that the model has been custom-trained in the network training stage for the Softmax classifier layer by back-propagation from the classification layer to the feature extraction part. Hence the original structure of the classifier works better for the feature extractor than our independent classifier. A relatively similar work to this paper is [54]. The authors compared a "one-shot" fine-tuning with Softmax to a multi-stage training method using support vector machines (SVMs) and the results indicated that Softmax slightly outperforms SVMs. On the other hand, the number of parameters in the classifier is much smaller than the number in the feature extraction layers, which leads to a fast recovery for the Softmax layer during the classifier training stage. To some degree, the Softmax layer recalls the samples in the network training set. In the case with an independent classifier (DE-WELM), the results are close to the original classifier performance of CifarNet, indicating the effectiveness and robustness of our algorithm.

In Fig. 5, we show the confusion matrices obtained by applying the six methods (CifarNet, ELM, WELM-W1, WELM-W2, BWELM, and DE-WELM). As can be observed, our method achieves better performance than CifarNet for four classes (bird, dog, frog, truck). It is worth considering here that "bird" and "truck" classes belong to the minority class. The sample numbers of "bird" and "truck" in C_TRAIN are respectively 95 and 58, which are the two smallest classes among the classes of C_TRAIN. This shows that our DE-WELM is more suitable for imbalanced data classification.

The experimental results on the selected feature data also show that DE-WELM achieves better performance compared with the other four methods.

E. Experimental Results on YouTube Dataset

In this dataset, visual and audio features are pre-extracted. Visual features are obtained using the Google Inception CNN, which is pre-trained on ImageNet [55], and those features are then reduced by PCA-compression into a 1024-dimensional vector. The audio features are extracted from a pre-trained VGG [56] network. In the official split, the dataset is divided into three parts: 70% for training, 20% for validation, and 10% for testing. In our experiment, we randomly selected 15,801 videos, which have unique top-level vertical categories with visual features from the official dataset, to validate our algorithm on video classification problems. The details of the selected datasets are shown in Table 8.

The experimental results of these five methods, i.e., WELM-W1, WELM-W2, BWELM, PSO-WELM, and DE-WELM, under selected imbalanced datasets are given in Table 9. The parameters for the experiment were the same as the ones in the experiment on standard classification datasets. As shown in Table 9, our DE-WELM achieves better performance than the other methods.

The experimental results indicate that our DE-WELM model is an effective and efficient algorithm for solving the class imbalanced problem in varied datasets. However, owing to an embedded optimization search procedure in the algorithm, DE-WELM requires more training time. Table 9 provides the total running time of various learning algorithms on the YouTube datasets. From Table 9, the execution time of DE-WELM is several times higher than WELM-W1 and WELM-W2. As our proposed DE-WELM employs the validation set in the training step, our algorithm is a little slower than BWELM.

VI. Conclusions

In this paper, we present the formal mathematical model to obtain the optimal weight scheme in WELM for imbalanced learning problems. We also propose DE-WELM to calculate the approximate optimal weight matrix of the model. The effectiveness of DE-WELM is proven by experiments conducted using 39 binary datasets and 3 multiclass datasets which have different degrees of imbalance, as well as two large-scale image datasets from MNIST and CIFAR-10, and one large-scale video dataset from YouTube-8M.

Our DE-based approximate weight calculation algorithm requires only 10 iterations to converge to a solution and uses a smaller search range than other state-of-the-art methods for the trade-off between constant C and the number of hidden nodes l. As the initial population in our approach contains two experienced weighting schemes, W_1 and W_2, our algorithm can preserve the advantages of WELM [7]. We also performed experiments comparing other computational methods, and the experimental results show that DE-WELM can improve overall classification performance significantly with less temporal complexity. Future work will focus on more effectively structuring the initial population into the DE algorithm, and applying DE-WELM to datasets with large variety in class distributions.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant U1509210 and U1609210, and the National Key Research and Development Program of China under Grant 2017YFB1302003.

References

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

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

[3] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing,* vol. 70, no. 16–18, pp. 3056–3062, Aug. 2007.

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

[5] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and its Applications*. New York, NY: Wiley, 1971.

[6] D. Serre, *Matrices: Theory and Applications*. New York, NY: Springer-Verlag, 2002.

[7] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, no. 3, pp. 229–242, Aug. 2013.

[8] K. Li, X. Kong, Z. Lu, L. Wenyin, and J. Yin, "Boosting weighted ELM for imbalanced learning," *Neurocomputing*, vol. 128, no. 5, pp. 15–21, May 2014.

[9] K.-A. Toh, "Deterministic neural classification," *Neural Comput.*, vol. 20, no. 6, pp. 1565–1595, Apr. 2008.

[10] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *Proc. IEEE Symp. Computational Intelligence and Data Mining*, Nashville, TN, Apr. 2009, pp. 389–395.

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

[12] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute, Berkeley, CA, Tech. Rep. TR-95-012, May 1995.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[14] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, ON, Tech. Rep., Apr. 2009.

[15] S. Abu-El-Haija, N. Kothari, J. Lee, A. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *CoRR*, vol. abs/1609.08675, Nov. 2016.

[16] R. Fletcher, *Practical Methods of Optimization, Volume 2: Constrained Optimization*. New York, NY: Wiley, 1981.

[17] E. Cambria, G.-B. Huang, L. L. C. Kasun, H. Zhou, C. M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li, V. C. M. Leung, L. Feng, Y.-S. Ong, M.-H. Lim, A. Akusok, A. Lendasse, F. Corona, R. Nian, Y. Miche, P. Gastaldo, R. Zunino, S. Decherchi, X. Yang, K. Mao, B.-S. Oh, J. Jeon, K.-A. Toh, A. B. J. Teoh, J. Kim, H. Yu, Y. Chen, and J. Liu, "Extreme learning machines [trends & controversies]," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 30–59, Nov. 2013.

[18] G.-B. Huang, E. Cambria, K. A. Toh, B. Widrow, and Z. Xu, "New trends of learning in computational intelligence [guest editorial]," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 16–17, May 2015.

[19] I. Chaturvedi, E. Ragusa, P. Gastaldo, R. Zunino, and E. Cambria. (2017, July). Bayesian network based extreme learning machine for subjectivity detection, *J. Franklin Inst.* [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0016003217303009

[20] E. Soria-Olivas, J. Gomez-Sanchis, J. D. Martin, J. Vila-Frances, M. Martinez, J. R. Magdalena, and A. J. Serrano, "BELM: Bayesian extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 22, no. 3, pp. 505–509, Mar. 2011.

[21] Z. Bai, G.-B. Huang, D. Wang, H. Wang, and M. B. Westover, "Sparse extreme learning machine for classification," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1858–1870, Oct. 2014.

[22] Z. Bai, G.-B. Huang, and D. Wang, "Sparse extreme learning machine for regression," in *Proc. Extreme Learning Machine,* Hangzhou, China, pp. 471–490, vol. 2, Dec. 2015.

[23] P. Gastaldo, R. Zunino, E. Cambria, and S. Decherchi, "Combining ELMs with random projections," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 46–48, Nov. 2013.

[24] R. E. Schapire, "The boosting approach to machine learning: An overview," in *Nonlinear Estimation and Classification*. New York, NY: Springer, 2003, pp. 149–171.

[25] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. IEEE Congr. Evolutionary Computation,* Portland, OR, June 2004, vol. 2, pp. 1980–1987.

[26] M. N. Alam, B. Das, and V. Pant, "A comparative study of metaheuristic optimization approaches for directional overcurrent relays coordination," *Electr. Power Syst. Res.*, vol. 128, pp. 39–52, June 2015.

[27] S. M. Lucas, "Evolving finite state transducers: Some initial explorations," in *Proc. European Conf. Genetic Programming*, Essex, Apr. 2003, pp. 130–141.

[28] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Process. Lett.*, vol. 17, no. 1, pp. 93–105, Mar. 2003.

[29] R. Storn, "On the usage of differential evolution for function optimization," in *Proc. North American Fuzzy Information Processing,* Berkeley, CA, June 1996, pp. 519–523.

[30] T. Rogalsky, S. Kocabiyik, and R. W. Derksen, "Differential evolution in aerodynamic optimization," *Can. Aeronaut. Space J.*, vol. 46, no. 4, pp. 183–190, Apr. 2000.

[31] J. Cao, Z. Lin, and G.-B. Huang, "Self-adaptive evolutionary extreme learning machine," *Neural Process. Lett.*, vol. 36, no. 3, pp. 285–305, July 2012.

[32] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, and G.-B. Huang, "Evolutionary extreme learning machine," *Pattern Recog.*, vol. 38, no. 10, pp. 1759–1763, Oct. 2005.

[33] Y. Qu, C. Shang, W. Wu, and Q. Shen, "Evolutionary fuzzy extreme learning machine for mammographic risk analysis," *Int. J. Fuzzy Syst.*, vol. 13, no. 4, pp. 282–291, Dec. 2011.

[34] V. L. Huang, A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for constrained real-parameter optimization," in *Proc. IEEE Congr. Evolutionary Computation,* Vancouver, BC, Canada, July 2006, pp. 17–24.

[35] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, May 2009.

[36] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, May 2011.

[37] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 689–707, May 2014.

[38] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, May 2009.

[39] Y. Wang, Z. Cai, S. Member, Q. Zhang, and S. Member, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Mar. 2011.

[40] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE Congr. Evolutionary Computation,* Hong Kong, China, June 2008, pp. 1110–1116.

[41] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, June 2009.

[42] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, Aug. 1963.

[43] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[44] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, Perth, Australia, Nov. 1995, vol. 4, pp. 1942–1948.

[45] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Machine and Human Science,* Nagoya, Japan, Oct. 1995, pp. 39–43.

[46] J.-H. Seo, C.-H. Im, C.-G. Heo, J.-K. Kim, H.-K. Jung, and C.-G. Lee, "Multimodal function optimization based on particle swarm optimization," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 1095–1098, Apr. 2006.

[47] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832–847, Dec. 2011.

[48] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Trans. Syst., Man, Cybern., Syst. B*, vol. 42, no. 3, pp. 627–646, June 2012.

[49] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition,* Providence, RI, June 2012, pp. 3642–3649.

[50] D. Keysers, T. Deselaers, C. Gollan, and H. Ney, "Deformation models for image recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1422–1435, Nov. 2007.

[51] A. T. Visual, P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks," in *Proc. IEEE Int. Conf. Document Analysis and Recogntion,* Edinburgh, Scotland, Aug. 2003, pp. 958–962.

[52] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets for handwritten digit recognition," *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, Dec. 2010.

[53] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia,* Orlando, FL, Nov. 2014, pp. 675–678.

[54] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Computer Vision*, Santiago, Chile, Dec. 2015, pp. 1440–1448.

[55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition,* Miami, FL, Nov. 2009, pp. 248–255.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition, in *Proc. IEEE Int. Conf. Learning Representations,* CO, June 2015, pp. 1150–1210.