Adaptive learning cost-sensitive convolutional neural network

Yun Hou | Hong Fan | Li Li | Bailin Li

Department of Mechanical Engineering, Southwest Jiaotong University, Chengdu, Sichuan, China

Correspondence
Yun Hou, Department of Mechanical Engineering, Southwest Jiaotong University, Chengdu, Sichuan 610031, China. Email: yun.hou@foxmail.com

Funding information
National Natural Science Foundation Youth Fund, Grant/Award Number: 51705436, Sichuan Province Science and Technology Support Programme Project, Grant/Award Number: 2016GZ0194 and Grant 2018GZ0361

Abstract
Real-world classification often encounters a problem called class imbalance. When the data of some classes are redundant than that of other classes, traditional classifiers usually bias their decision boundaries to the redundant majority classes. Most proposed cost-sensitive strategies often ignore the hard-to-learn examples or have a large amount of hyper-parameters. This article proposes an adaptive learning cost-sensitive convolutional neural network to solve this problem. During the training process, the proposed method embeds a class-dependent cost to each class in the global error, making the decision boundary bias to the minority classes. Meanwhile, a distribution weight is assigned to each example to enhance the learning of the hard-to-learn examples. Both the class-dependent costs and distribution weights are learnt automatically in the net. This cost-sensitive approach makes the algorithm focus on the examples in the minority classes as well as the hard-to-learn examples in each class. Besides, this approach can be applied to both binary and multi-class image classification problems without any modification. Experiments are conducted on four image classification datasets to evaluate this algorithm. The experimental results show that the proposed method achieves better performance than the baseline algorithms and some other algorithms.

1  INTRODUCTION

A dataset is imbalanced if some of its classes have fewer data than other classes. This situation can be seen in many real-world applications, such as financial crisis prediction [1], medical diagnosis [2] and e-mail filtering [3]. In these scenarios, the minority class examples are so rare that it is difficult for researchers to collect sufficient data to represent them. Since the data distribution of the training set is asymmetric, traditional algorithms that aim to minimise the global error often misclassify the minority class examples into majority classes. Actually, the minority classes are often the objects that the researchers are more interested in [1–3].

Many approaches have been proposed to solve the imbalance problem. These approaches can be divided into two categories: data-level approaches and algorithm-level approaches. Data-level approaches change the data distribution by under-sampling the majority classes or over-sampling the minority classes. Among them, the under-sampling methods [4, 5], which solve the class imbalance problem by removing some examples from the majority classes, may lose some useful information. The over-sampling methods, which solve the class imbalance problem by replicating some examples from the minority classes, may result in over-fitting of the model in its build process. To overcome over-fitting, Chawla et al. [6] proposed a synthetic minority over-sampling technique (SMOTE) to generate synthetic minority class examples by linear interpolation. However, SMOTE may result in a problem called overgeneralization since some of its generated minority class examples locate inside the convex hull of the majority class examples. To solve this overgeneralization problem, several variants of SMOTE have been proposed such as SL-SMOTE [7], RSB-SMOTE [8], B-SMOTE [9], LNSMOTE [10], etc. However, a main shortage of these variants is the large computational cost. Algorithm-level approaches modify the traditional classifiers for the balanced datasets such that the modified algorithms can be applied to the imbalanced datasets. A popular algorithm-level approach is a cost-sensitive strategy which assigns different misclassification costs to each class in the imbalanced dataset. Kukar et al. [11] proposed four cost-sensitive approaches to modify the neural networks. Wang et al. [12] proposed boosting-SVM with an asymmetric cost algorithm to solve the imbalance problem. Arya et al. [13] proposed a cost-sensitive support vector machine algorithm by
extending the SVM hinge loss with cost information. Sun et al. [14] proposed three cost-sensitive boosting algorithms by inserting cost items into the adaptive-boosting algorithm (AdaBoost). Zhang et al. [15] proposed an evolutionary cost-sensitive extreme learning machine algorithm, which can adaptively learn the cost matrix through an evolutionary backtracking search algorithm. Ting et al. [16] introduced an instance-weighting method to induce cost-sensitive trees. Zhang et al. [17] integrated cost information into deep belief network and used an adaptive differential evolution method to estimate the misclassification costs based on the training set. However, all these methods are not able to deal with the class imbalance problem of convolutional neural network (CNN), which has been the most popular tool for image classification. In recent years, some new algorithms have been proposed to insert class-dependent costs into the deep CNN. Wang et al. [18] proposed a loss function method called mean false error together with its improved version mean squared false error to train deep neural networks on imbalanced dataset. Raj et al. [19] integrated a class-to-class separability score into deep CNN to improve the classification performance on an imbalanced dataset. However, these two methods mainly focus on the binary classification problem and cannot be applied to the multi-class classification problem directly. Chung et al. [20] designed a smooth one-sided regression loss function for the multi-class classification problem, which embeds the cost information into the training process of deep learning. However, the misclassification cost for each class is assigned based on expert judgement, which turns into a tedious task for a large number of classes. Khan et al. [21] proposed a cost-sensitive deep neural network in which costs are determined by data distribution, classification errors and class separability. Although Khan’s method can jointly optimise the neural network parameters and class-dependent costs, implementing this method is a complicated task. Besides, all these cost-sensitive methods do not take into account the differences between examples in the same class, which is an important factor affecting the performance of classifiers [22–24]. In recent years, other methods, such as online hard example mining (OHEM) [22], focal loss (FL) [25], gradient harmonising mechanism (GHM) [23] and distributional ranking loss (DR) [26] were proposed to solve the imbalance problem in object detection. These methods take into account the imbalance between majority and minority classes as well as that between easy and hard examples. However, these methods have many hyper-parameters, tuning them should be a tedious task during the training process.

An adaptive learning cost-sensitive convolutional neural network (Alcs-CNN) approach is proposed, which can avoid the setting of the costs artificially. Alcs-CNN incorporates a class-dependent cost for each class into the global error of CNN, so that the minority classes are being more concerned. Meanwhile, to enhance the learning of the hard-to-learn examples in each class, a distribution weight is assigned to each example during the training process. Both the class-dependent costs and distribution weights are updated automatically.

The innovations of this article are as follows: 1) A cost-sensitive strategy for CNN, which can avoid setting the costs artificially is proposed. The proposed strategy can be applied to both binary and multi-class image classification. 2) The strategy is embed into several baseline CNNs and conduct a large number of experiments on four image classification datasets. The results have shown that this approach performs better than the baseline CNNs and some other algorithms.

This article is organised as follows. Section 2 recalls the CNN and the cost-sensitive strategy. Section 3 describes our Alcs-CNN in detail. Section 4 compares our proposed approach with other algorithms, and Section 5 summarises this paper.

2 | BACKGROUND

2.1 | Convolutional neural network

Convolutional neural network is a feed-forward artificial neural network, which has been successfully used in classification [27], image recognition [28] and segmentation [29]. As the network avoids the complex feature extraction of images, it is possible to input the original images directly for processing. The core idea of convolutional neural network is to obtain an optimal model \( \theta^* \) by minimising the global error \( E(\theta) \) in back-propagation process.

\[
\theta^* = \arg\min \ E(\theta)
\]  

As indicated, a training set \( T = (X, d)_N \), \( X \) is the given data, \( d \) is the actual label, and \( N \) is the number of examples in the training set. The global error of training set \( T \) can be expressed as Equation (2).

\[
E = \frac{1}{N} \sum_{i=1}^{N} l(d^{(i)}, y^{(i)})
\]  

where \( y^{(i)} \) is a predicted label for example \( X_i \), and \( l(\bullet) \) is the loss function. There are several loss functions including mean squared error loss (MSE), cross entropy loss (CE) and SVM hinge loss.

(1) MSE loss

MSE loss is the squared error between actual label and predicted label. The mathematical formula can be expressed as Equation (3).

\[
l(d, y) = \frac{1}{2} \sum_{k=1}^{K} (d_k - y_k)^2
\]  

where \( K \) is the number of classes in training set, the actual label \( d \in \{0,1\}^{1 \times K} \) (s.t. \( \sum_k d_k = 1 \)) and the predicted label \( y_k \) is given by
\[ y_k = \frac{1}{1 + \exp(-a_k)} \]  
\no_k is the output of the previous layer.

(2) CE loss

CE loss measures the proximity between actual label and predicted label. The mathematical formula can be expressed as Equation (5).

\[ l(d, y) = \sum_{k=1}^{K} (-d_k \log y_k) \]  
\nwhere the predicted label \( y_k \) is given by

\[ y_k = \frac{\exp(a_k)}{\sum_{i=1}^{K} \exp(a_i)} \]  
\n(3) SVM hinge loss

SVM hinge loss measures the margin between each pair of classes. The mathematical formula can be expressed as Equation (7).

\[ l(d, y) = \sum_{k=1}^{K} \max(0, 1 - (2d_k - 1)y_k) \]  
\nwhere the predicted label \( y_k \) is given by

\[ y_k = o_k \]  

In artificial neural networks, back-propagation is applied to calculate a gradient that is needed in the update process of network weights. The weights \( \theta \) are updated according to the following equation.

\[ \theta^{(e+1)} = \theta^{(e)} - \eta \frac{\partial E}{\partial \theta} \]  
\nwhere \( \eta \) is the learning rate of \( \theta \).

2.2 Cost-sensitive strategy

During the training process of a CNN, the network weights are updated in order to minimise the global error of the training set. However, since the minority classes are underrepresented in an imbalanced dataset, they only account for a small percentage of the global error, which may cause the classifier to ignore these minority classes. It is undesirable and dangerous to misclassify the minority class examples. Cost-sensitive strategy is an effective way to deal with this imbalance problem [11–21]. It improves the accuracy of minority classes by penalising the misclassification of each class differently based on a given cost matrix \( C \). The cost matrix \( C \) is a \( K \times K \) matrix, where each element \( C(y_k, y) \in [0, \infty) \) denotes the cost for classifying a \( y \)-th class example as \( k \)-th class. In this cost matrix, the cost of misclassifying a majority class example is less than that of misclassifying a minority class example. Specially, \( C(y_k, y) = 0 \) in the matrix \( C \). Such a cost matrix can make the classifier focus more on the minority classes. In most cases, the value of each element in the cost matrix is designed based on expert judgement. This is a tedious task for a dataset with a large number of classes. Here we try to solve the imbalance problem in a novel and simple way to avoid setting the costs artificially.

2.3 Evaluation criterion

Average accuracy is the most popular evaluation criterion to evaluate the classification performance of classifier. Since this evaluation criterion usually results in misclassification of the minority class examples, it has been proved unsuitable for imbalance classification. For example, an existing dataset contains 999 positive examples and one negative example. An accuracy-oriented classifier would classify all the examples as positive examples. Therefore, the overall accuracy of the classifier is 99.9%, while the accuracy of negative examples is 0%. In order to overcome the shortage of average accuracy, a new evaluation criterion, G-mean, is introduced to measure the classification performance of classifier on imbalanced datasets [30].

\[ G - \text{mean} = \left( \prod_{k=1}^{K} R_i \right)^{\frac{1}{K}} \]  
\nwhere \( R_i \) is the recall value of \( i \)-th class.

\[ R_i = \frac{n_{ii}}{\sum_{j=1}^{K} n_{ij}} \]  
\nn_{ij} is the number of examples from \( i \)-th class classified as \( j \)-th class.

3 THE PROPOSED METHOD

In traditional CNN, the global error is the average error of all examples in the training set. Since the minority classes are underrepresented, they only account for a small percentage of the global error. This make the minority classes are easily misclassified in the training process of CNN. To bias CNN
to the minority classes, we introduce Global Mean Square Error Separation (GMSES) [31] to differentiate the errors of different classes. The core idea of GMSES is to achieve the expected sensitivity and specificity by calculating a weighted sum of the gradients for each class. In GMSES, the global error is the weighted sum of errors of all classes in the training set.

\[
E = \sum_{k=1}^{K} C_k \cdot E_k
\]  

(12)

where \( K \) is the number of classes in the training set \( T \). \( C_k \) is the class-dependent cost of \( k \)-th class. \( E_k \) is the error of \( k \)-th class.

**Algorithm 1** The Pseudo-code of Alcs-CNN

| Input: Training set: \( T = (X_T, d_T) \); learning rate: \( \eta_0 \); the number of epochs: \( M_{ep} \) and batch size: \( B \). |
| Output: Learnt parameters: \( \theta \) |
| Step 1. Net ← construct_CNN(\( \theta \)) |
| Step 2. \( \theta \) ← initialize_CNN(Net) |
| Step 3. \( C_i \) ← 1/\( K \); \( W_i(X_j) = 1/N_{k_i} \), \( X_j \in \text{class } k \). |
| Step 4. For \( e = 1, 2, \ldots, M_{ep} \) do |
| For \( b = 1, 2, \ldots, B \) do |
| \( y_b \) ← forward-pass(\( X_b, d_b, \) Net, \( \theta \)) |
| \( \text{grad}_b \) ← backward-pass(\( X_b, d_b, \) Net, \( \theta, y_b, C_b, W_b \)) |
| \( \theta^* \) ← update-NetParams(Net, \( \theta, \eta_0, \text{grad}_b \)) |
| \( \theta \leftarrow \theta^* \) |
| end for. |
| \( \text{errors} \) ← forward-pass(\( X_T, d_T, \) Net, \( \theta \)) |
| \( y \) ← softmax(errors) |
| \( G\text{-mean} \) ← Algorithm2(\( d_T, y \)) |
| \( C_i \) ← update_C(\( \text{errors} \)) |
| \( W_{t+1} \) ← update_W(\( G\text{-mean}, W_t \)) |
| end for. |
| Step 5. Output the parameters \( \theta \) |

**Algorithm 2** The subprogram for calculating G-mean

| Input: Actual labels: \( d_T \); predicted labels: \( y \). |
| Output: \( G\text{-mean} \). |
| For \( i = 1, 2, \ldots, K \) do |
| For \( j = 1, 2, \ldots, K \) do |
| If \( i = j \) do |
| \( n_{ii} \) ← statistical-results(\( d_T, y \)) |
| endif. |
| \( n_{ij} \) ← statistical-results(\( d_T, y \)); \( i \neq j \)) |
| endfor. |
| \( R_j \) ← calculate-recall(\( n_{ii}, n_{ij} \)) |
| endfor. |
| \( G\text{-mean} \) ← calculate-gmean(\( R \)) |
| Output the value: \( G\text{-mean} \). |

Since the minority classes are easily misclassified, their errors are usually larger than the majority classes. Here, we try to assign the class-dependent costs according to the error of each class directly. The goal is to increase the proportions of minority classes in the global error, forcing the classifier to focus on those minority classes. The details of Alcs-CNN are shown in Algorithm 1. Note that class-dependent costs are updated iteratively during the whole training process. The initial cost of each class is set uniformly to \( 1/K \). The class-dependent costs in next epoch are updated according to their own errors in the previous net.

\[
C_k = \frac{\exp(e_k)}{\sum_{l=1}^{K} \exp(e_l)}
\]  

(13)

where \( e_k \) is the mean error of \( k \)-th class.

\[
e_k = \frac{1}{N_k} \sum_{i=1}^{N_k} l_k(i)
\]  

(14)

where \( l_k(i) \) is the error (loss) of example \( i \) in \( k \)-th class, \( N_k \) is the number of examples in \( k \)-th class.

Class imbalance problem is an important factor that affects the classification, but not the only one. Different examples in the same class have different effects on the classification [22, 23, 25, 26]. The examples near the decision boundary often play more important roles than other examples in determining the boundary. Typically, these examples near the boundary are more difficult to learn. AdaBoost [32] is an effective boosting algorithm that can improve the accuracy of hard-to-learn examples. It can adjust the distribution weights on the training examples iteratively depending on the training error of the previous classifier. During this process, the examples that are easily misclassified will obtain larger distribution weights than other examples. This forces the following classifier to concentrate on those hard-to-learn examples. To enhance the learning of such examples, different distribution weights are assigned to different examples in the error function based on the idea of AdaBoost.

\[
E_k = \sum_{i=1}^{N_k} w(X_i) l(d^{(i)}, y^{(i)})
\]  

(15)

where \( w(X_i) \) is the weight of example \( X_i \). In contrast to AdaBoost, this algorithm updates the distribution weight of each example according to the G-mean value of the previous net. The initial weight of each example in the training set is set uniformly to \( 1/N \). The updating process is shown as the following equation.

\[
W_{e+1}(X_i) = \frac{W_e(X_i) \exp\left(-at(d^{(i)}, y^{(i)})\right)}{Z_e}
\]  

(16)
where $\alpha = 0.5 \ln(G\text{-mean}/(1-G\text{-mean}))$ is the weight updating parameter. $Z_k = \sum (W_k (X_k) \exp(-\alpha d_k))$ is a normalisation factor. $1(d, y)$ is an indicator function.

$$1(d, y) = \begin{cases} 1 & \text{if } d = y \\ 0 & \text{else} \end{cases} \quad (17)$$

As the Equation (16) shows, the weights of correctly classified examples are reduced to the original $\exp(-\alpha)$ times while the weights of misclassified examples are unchanged. Note that the distribution weights are only updated when $G\text{-mean} > 0.5$.

In this approach, the minority classes that are often misclassified will obtain larger class-dependent costs than the majority classes. Moreover, the hard-to-learn examples are assigned larger distribution weights than other examples. This forces the classifier to focus on those minority classes and the hard-to-learn examples of each class during the next training process.

Since the global error is just the weighted sum of errors of all classes, the gradient computation of a single example at each neuron is the same as in the original case. The details of the computational process are as follows.

(4) MES loss:

For the MES loss, the gradient at each output neuron is calculated by

$$\frac{\partial l(d, y)}{\partial o_k} = -(d_k - y_k) \frac{\exp(-o_k)}{(1 + \exp(-o_k))^2}. \quad (18)$$

The $y_k$ value is given by Equation (4).

$$y_k = \frac{1}{1 + \exp(-o_k)}$$

Therefore, the gradient in Equation (18) can be expressed as

$$\frac{\partial l(d, y)}{\partial o_k} = -(d_k - y_k) y_k (1 - y_k) \quad (19)$$

(5) CE loss

For the CE loss, the gradient at each output neuron is calculated by

$$\frac{\partial l(d, y)}{\partial o_j} = -\sum_{k=1}^{K} d_k \frac{\partial y_k}{o_j} \quad (20)$$

The partial derivative in Equation (20) is solved in two cases: $k \neq j$ and $k = j$. When $k \neq j$,

$$\frac{\partial y_k}{o_j} = -\frac{\exp(y_k + y_j)}{\left(\sum_{l=1}^{K} \exp(o_l)\right)^2} = -y_k y_j, \quad \text{s.t. : } k \neq j.$$  

When $k = j$,

$$\frac{\partial y_k}{o_j} = \frac{\exp(o_j) \sum_{l=1}^{K} \exp(o_l) - \exp(2o_j)}{\left(\sum_{l=1}^{K} \exp(o_l)\right)^2}$$

$$= y_j (1 - y_j), \quad \text{s.t. : } k = j.$$  

Therefore, the gradient in Equation (20) can be expressed as

$$\frac{\partial l(d, y)}{\partial o_j} = \left(\sum_{k \neq j} d_k y_j\right) - d_j (1 - y_j)$$

$$= \left(\sum_{k} d_k y_j\right) - d_j. \quad (21)$$

Since $\sum_{k} d_k = 1$, Equation (21) is equivalent to the following equation.

$$\frac{\partial l(d, y)}{\partial o_j} = y_j - d_j \quad (22)$$

(6) SVM hinge loss

For the SVM hinge loss, the gradient at each output neuron is calculated by

$$\frac{\partial l(d, y)}{\partial o_k} = -(2d_k - 1) \frac{\partial y_k}{\partial o_k} 1[y_k (2d_k - 1) < 1]$$

$$= -(2d_k - 1) 1[y_k (2d_k - 1) < 1]. \quad (23)$$

where $1(\cdot)$ is an indicator function.

4 | EXPERIMENTAL RESULTS

4.1 | Network architecture

Experiments were conducted on three networks: VGG-19, Resnet-18 and Densenet ($L = 40$, $k = 12$). Their network architectures follow the works in [33–35], respectively. Their error functions are modified by inserting cost-sensitive
strategy. The modified architectures are identical to the baseline CNNs except the error functions in back-propagation process. All the networks are trained using stochastic gradient descent (SGD). Furthermore, all the experiments are conducted with CE loss function, as it has been proved that CE loss performs better than MES loss and SVM hinge loss [36]. In the experiments, the weight decay is $10^{-4}$, the momentum is 0.9, the batch size is 200 and the epochs are 163. A dropout layer is added after each convolutional layer (except the first one) and the dropout rate is set to 0.5. The initial learning rate is 0.01 and is decreased by a factor of 0.1 after 50-th epoch and then 80-th epoch. For image preprocessing, we normalise the data using the channel means and standard deviations. To facilitate the comparison with the data-level methods, the experiments use VGG as the baseline network.

### 4.2 Datasets

In this article The experiments are conducted on two balanced datasets: CIFAR10 and MNIST and two imbalanced datasets: Edinburgh dermofit image library (DIL) and moorea labelled corals (MLC). CIFAR10 is a well-known image classification dataset which contains 10 classes such as animals, ship and car, etc. MNIST is a handwritten digit dataset which contains 10 classes of digits from 0 to 9. For these two datasets, the training and test split follow the source websites. One of the most important uses of inserting cost-sensitive strategy into classifiers is to improve the accuracy of the minority classes in imbalanced datasets. Nevertheless, the two datasets CIFAR10 and MNIST are somewhat balanced. To evaluate the algorithm on imbalanced datasets, the construction of the corresponding imbalanced datasets based on these two balanced datasets is needed. For each dataset, an imbalanced dataset is constructed by removing 70% or 90% of the examples that belong to the even-numbered classes. For convenience, these imbalanced datasets are named as CIFAR10$_{imb70}$, MNIST$_{imb70}$, CIFAR10$_{imb90}$ and MNIST$_{imb90}$, respectively. DIL is an imbalanced dataset used for melanoma detection, which contains 960 skin lesion images belonging to five classes (45 actinic keratosis, 239 basal cell carcinoma, 331 melanocytic nevus/mole, 88 squamous cell carcinoma, 257 seborrheic keratosis).

We split the dataset to build the training and testing sets according to [37]. MLC is a subset of the MCR LTER packaged for computer vision research, which contains 2055 images from three habitats from 2008 to 2010. Each image contains roughly 200 points belonging to nine classes. The number of examples in each class varies approximately from 2622 to 196910. We use the data from year 2008 as training set and that from year 2009 as testing set. The number of examples in these training and testing sets are shown in Table 1.

#### 4.3 Experimental results

Both the CIFAR10 dataset and its two imbalanced variants, were experimented to compare our Alcs-CNN with the baseline CNN (VGG-19) [33]. Table 2 shows the accuracy and G-mean results of the comparisons and the best result in each experiment is highlighted in bold. From Table 2, it is observed that all the accuracy and G-mean values decrease with the increase of imbalanced degree. This method has slightly higher accuracy than baseline CNN. Meanwhile, this method achieves a G-mean gain of 3% and 5% over baseline CNN on CIFAR10$_{imb70}$ and CIFAR10$_{imb90}$, respectively. Figure 1a,b shows the confusion matrices of the baseline CNN and Alcs-CNN on CIFAR$_{imb90}$, respectively. From the figure, a boost in performance for the even-numbered classes is observed, for example, in aeroplane, bird, deer, frog and ship.

For the MNIST dataset and its two imbalanced variants, experiments following the works on the CIFAR10 dataset are conducted. Note that MNIST is a binary dataset, and the

| Datasets | Original | $X_{imb70}$ | $X_{imb90}$ |
|----------|----------|------------|------------|
|          | Training set | Testing set | Training set | Testing set | Training set | Testing set |
| CIFAR10  | 50,000   | 10,000     | 32,500     | 10,000     | 27,500     | 10,000     |
| MNIST    | 55,000   | 10,000     | 34,527     | 10,000     | 29,527     | 10,000     |
| DIL      | 640      | 320        | ---        | ---        | ---        | ---        |
| MLC      | 268,400  | 278,000    | ---        | ---        | ---        | ---        |

Abbreviations: DIL, dermofit image library; MLC, moorea labeled corals.

| Datasets | Accuracy | G-Mean |
|----------|----------|--------|
|          | Baseline-CNN | Alcs-CNN | Baseline-CNN | Alcs-CNN |
| CIFAR10  | 0.9309   | 0.9309  | 0.9301     | 0.9304   |
| CIFAR10$_{imb70}$ | 0.8792 | 0.8938 | 0.8623 | 0.8901 |
| CIFAR10$_{imb90}$ | 0.7666 | 0.8082 | 0.7534 | 0.8025 |

Abbreviation: Alcs-CNN, adaptive learning cost-sensitive convolutional neural network.
colour channel of the first convolution layer was changed to one in the network architectures. Table 3 shows the experimental results. From Table 3, it is observed that this method achieves higher accuracy and G-mean values than baseline CNN on the imbalanced MNIST datasets. Their confusion matrices are shown in Figure 1c,d.

We also conduct experiments on two real-world imbalanced datasets: DIL and MLC. For the dataset DIL, threefold cross validation was performed on the five classes similar to the work of Ballerini et al. [37]. For the dataset MLC, experiments were conducted following the work of Khan et al. [21]. Table 4 shows the experimental results. From Table 4, it is observed that our method achieves higher accuracy and G-mean values than baseline CNN on these imbalanced datasets. Figure 1e,f,g,h are the confusion matrices on DIL and MLC. From these confusion matrices, a boost in
performance for the least frequent classes is observed. For example, actinic keratosis, squamous cell carcinoma, turf, macro and monti.

This method is also compared with two data-level methods (Rus [4], SMOTE [6]) and three cost-sensitive deep learning methods (CSNN [19], Sosr-CNN [20] and Cosen-CNN [21]) on these imbalanced datasets. Besides, it is also compared with other two methods used for object detection: FL [25] and GHM-C [23]. For fairness, the experimental process should be conducted as close as possible to the proposed Alcs-CNN. For the two data-level methods, a 2048 dimensional feature extracted from the pre-trained deep CNN [33] was used as the input feature. Note that, this 2048 dimensional feature is the input of the first fully connected layer in the pre-trained deep CNN. Furthermore, a three-layered neural network, which is identical to the last three fully connected layers in VGG19, was used for classification with these data-level methods. For CSNN, one-vs-all method was used to make it applicable to multi-class classification. For Sosr-CNN, their proposed smooth one-sided regression (SOSR) loss was incorporated as the last layer of the baseline CNN in these experiments. The fixed cost matrix of Sosr-CNN was generated according to the approach in [38]. For Cosen-CNN, two extra fully connected layers were deleted before the output layer in the experiments.

For FL, the hyper-parameters are set as $\gamma = 2$ and $\alpha = 0.25$. For GHM-C, the hyper-parameters are set as $\alpha = 0.75$ and $M = 30$. Table 5 shows the experimental G-mean results of the comparisons. From Table 5, it is observed that the proposed method has an improvement of G-mean over all of these methods.

In Alcs-CNN, the class-dependent costs are updated according to the errors of different classes in each epoch. To validate the effectiveness of the proposed method, the class-dependent costs in two cases were fixed and experimented with them. In the first case, the cost of each class was set to $1/K$. In the second case, the cost of each class was set according to its own proportion in the training set (PT). For convenience, the former is named as Alcs-CNN$_{1/K}$ and the latter as Alcs-CNN$_{PT}$. The experimental G-mean results are shown in Table 6. The results show that this cost-sensitive strategy achieves better performance than these two fixed cost strategies.

Moreover, to further validate the effectiveness of the proposed cost-sensitive strategy, Resnet [34] and Densenet [35] were also used as the baseline algorithms for experiments. Their cost-sensitive variants were named as Alcs-Resnet and Alcs-Densenet, respectively. In the experiments, Resnet and its variant have 18 layers while Densenet and its variant have the configurations of $L = 40$, $K = 12$. Table 7 shows the experimental G-mean results of the comparisons. From Table 7, it is observed that the proposed cost-sensitive strategy can improve the performance of both Resnet and Densenet.

Finally, the training and validation loss curves were shown in Figure 2. As the figure shown, both the two loss curves tend to be flat as the number of epoch increases.

5 | CONCLUSION

Imbalance classification is an important task in many domains. In this article, we proposed a novel cost-sensitive convolutional neural network to solve the imbalance problem. The proposed method replaced the global error function of deep CNN with GMSES so that the minority classes were being more concerned. Meanwhile, this method has enhanced the learning of hard-to-learn examples in each class by assigning a distribution weight to each example. Experiments were conducted on four image classification datasets to evaluate this method.
TABLE 6  Comparisons of this method with two fixed-costs methods on G-mean (%)

| Datasets   | Alcs-CNN1/K | Alcs-CNNPT | Alcs-CNN  |
|------------|-------------|------------|-----------|
| CIFAR10    | 0.9282      | 0.9223     | 0.9304    |
| CIFAR10imb/70 | 0.8839    | 0.8642     | 0.8901    |
| CIFAR10imb/90 | 0.7830    | 0.6363     | 0.8025    |
| MNIST      | 0.9938      | 0.9938     | 0.9943    |
| MNISTimb/70 | 0.9914      | 0.9854     | 0.9932    |
| MNISTimb/90 | 0.9841      | 0.9719     | 0.9883    |
| DIL        | 0.8129      | 0.7951     | 0.8249    |
| MLC        | 0.7286      | 0.7028     | 0.7342    |

Abbreviations: Alcs-CNN, adaptive learning cost-sensitive convolutional neural network; DIL, dermofit image library; MLC, moorea labeled corsals.

TABLE 7  Results of the other two baseline algorithms on G-mean (%)

| Datasets   | Resnet | Alcs-Resnet | Densenet | Alcs-Densenet |
|------------|--------|-------------|----------|---------------|
| CIFAR10    | 0.8866 | 0.8964      | 0.9996   | 0.9263        |
| CIFAR10imb/70 | 0.8245  | 0.8436      | 0.8566   | 0.8900        |
| CIFAR10imb/90 | 0.7567  | 0.7758      | 0.7743   | 0.8412        |
| MNIST      | 0.9954 | 0.9962      | 0.9924   | 0.9942        |
| MNISTimb/70 | 0.9874  | 0.9935      | 0.9792   | 0.9886        |
| MNISTimb/90 | 0.9829  | 0.9907      | 0.9760   | 0.9813        |
| DIL        | 0.7903 | 0.8083      | 0.8057   | 0.8125        |
| MLC        | 0.7034 | 0.7206      | 0.6902   | 0.7146        |

Abbreviations: DIL, dermofit image library; MLC, moorea labeled corsals.

FIGURE 2  The training/validation loss curves of adaptive learning cost-sensitive-convolutional neural network on CIFAR_9imb/90

Experimental results, it can be concluded that this method can improve the performance of the classifier on an imbalanced dataset. In comparison with other class-imbalance methods, this cost-sensitive strategy not only takes into account the imbalance between classes (class-class imbalance), but also the imbalance between examples (easy/hard examples imbalance). Besides, the proposed strategy can also avoid setting the hyperparameters artificially. However, for the training set containing noisy labels, this strategy will continuously increase the distribution weights of noisy examples in the following epochs, which may degrade the performance of the classifiers. This issue will be delved in the future work.

ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation Youth Fund under Grant 51,705,436 and the Sichuan Province Science and Technology Support Programme Project under Grant 2016GZ0194 and Grant 2018GZ0361.

REFERENCES

1. Zhou, L.: Performance of corporate bankruptcy prediction models on imbalanced dataset: the effect of sampling methods. Knowl. Base. Syst. 41(1), 16–25 (2013)
2. Krawczyk, B., et al.: Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. Appl. Soft Comput. 38, 714–726 (2016)
3. Bermejo, P., et al.: Improving the performance of naive bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets. Expert Syst. Appl. 38(3), 2072–2080 (2011)
4. Seiffert, C., et al.: Rusboost: a hybrid approach to alleviating class imbalance. IEEE Trans. Syst. Man Cybern. A Syst. Hum. 40(1), 185–197 (2010)
5. Liu, X.Y., et al.: Exploratory undersampling for class-imbalance learning. IEEE Trans. Syst. Man Cybern. 39(2), 539–550 (2009)
6. Chawla, N.V., et al.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16(1), 321–357 (2002)
7. Bunkhumpoompat, C., et al.: Safe-level-smot: safe-level-synthetic minority over-sampling technique for handling the class imbalance problem. In: 13th Pacific-Asia Conference on Knowledge Discovery and data mining, pp. 475–482. Springer, Heidelberg (2019)
8. Ramentol, E., et al.: Smote-nb: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced datasets using smote and rough sets theory. Knowl. Inf. Syst. 33(2), 245–265 (2012)
9. Han, H., et al.: Borderline-SMOTE: A new oversampling method in imbalanced data sets learning. In: International Conference on Intelligent Computing, pp. 878–887. Springer, Heidelberg (2005)
10. Maciejewski, T., Stefanowski, J.: Local neighbourhood extension of smote for mining imbalanced data. In: IEEE Symposium on Computational Intelligence and Data Mining, pp. 104–111. IEEE, Paris (2011)
11. Kukar, M., Kononenko, I.: Cost-sensitive learning with neural networks. In: 13th European Conference on Artificial Intelligence, pp. 445–449. John Wiley & Sons (1998)
12. Wang, B.X., Japkowicz, N.: Boosting support vector machines for imbalanced data sets. Knowl. Inf. Syst. 25(1), 1–20 (2010)
13. Arya, I., et al.: Cost-sensitive support vector machines. Neurocomputing. 343, 50–64 (2019)
14. Sun, Y., et al.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn. 40(12), 3358–3378 (2007)
15. Zhang, L., Zhang, D.: Evolutionary cost-sensitive extreme learning machine. IEEE Trans. Neur. Net. Lear. Syst. 28(12), 3045–3059 (2017)
16. Ting, K.M.: An instance-weighting method to induce cost-sensitive tree. IEEE Trans. Knowl. Data Eng. 14(3), 659–665 (2002)
17. Zhang, C., et al.: A cost-sensitive deep belief network for imbalanced classification. IEEE Trans. Neur. Net. Lear. Syst. 30(1), 109–122 (2019)
18. Wang, S., et al.: Training deep neural networks on imbalanced data sets. In: Proceedings of the International Joint Conference on Neural Networks, pp. 4368–4374. IEEE, Vancouver (2016)
19. Raj, V., et al.: Towards effective classification of imbalanced data with convolutional neural networks. In: IAPR Workshop on Artificial Neural Networks in Pattern Recognition, pp. 150–162. Springer, Magdeburg (2016)

20. Chung, Y.A., et al.: Cost-aware pre-training for multiclass cost-sensitive deep learning. In: 25th International Joint Conference on Artificial Intelligence, pp. 1411–1417. New York (2015)

21. Khan, S.H., et al.: Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Trans. Neur. Net. Lear. Syst. 29(8), 3573–3587 (2018)

22. Shrivastava, A., et al.: Training region-based object detectors with online hard example mining. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 761–769. Las Vegas (2016)

23. Li, B.Y., et al.: Gradient harmonised single-stage detector. In: 33rd AAAI Conference on Artificial Intelligence, pp. 8577–8584. Hawaii (2019)

24. Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. J. Big Data 6(1), 6–27 (2019)

25. Lin, T.Y., et al.: Focal loss for dense object detection. IEEE Trans. Pattern Anal. Mach. Intell. 42(2), 318–327 (2018)

26. Qian, Q., et al.: DR loss: improving object detection by distributional ranking. In: IEEE International Conference on Computer Vision, pp. 12161–12169. Seattle (2019)

27. Han, H.M., et al.: Convolutional neural network learning for generic data classification. Inf. Sci. 477, 448–465 (2019)

28. Liu, Q., et al.: A review of image recognition with deep convolutional neural network. In: International Conference on Intelligent Computing, pp. 69–80. Liverpool (2017)

29. Marta, B.C., Susana, K.L.Y.: AdaResU-Net: multiobjective adaptive convolutional neural network for medical image segmentation. Neurocomputing 392, 325–340 (2020)

30. Li, K., et al.: Boosting weighted elm for imbalanced learning. Neurocomputing, 128, 15–21 (2014)

31. Castro, C.L., Braga, A.P: Artificial neural networks learning in ROC space. In: International Joint Conference on Computational Intelligence, pp. 484–489. Madeira (2009)

32. Schapire, R: The boosting approach to machine learning: an overview. In: MSRI Workshop on Nonlinear estimation and classification, pp. 149–172. New York (2003)

33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, pp.1–14. San Diego (2014)

34. He, K., et al.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–12. Las Vegas (2016)

35. Huang, G., et al.: Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2261–2269. IEEE, Honolulu (2017)

36. Lee, C.Y., et al.: Deeply-supervised nets. In: International Conference on Artificial Intelligence and Statistics, pp. 562–570, San Diego (2015)

37. Ballerini, L., et al.: A colour and texture based hierarchical K-NN approach to the classification of non-melanoma skin lesions. In: Color Medical Image Analysis, pp. 63–86. Springer, Dordrecht (2013)

38. Abe, N., et al.: An iterative method for multi-class cost-sensitive learning. In: Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 3–11. Seattle (2004)

How to cite this article: Hou, Y., et al.: Adaptive learning cost-sensitive convolutional neural network. IET Comput. Vis. 15(5), 346–355 (2021). https://doi.org/10.1049/cvi2.12027