Applied Intelligence (2022) 52:6723–6738
https://doi.org/10.1007/s10489-021-02623-9

BBW: a batch balance wrapper for training deep neural networks on extremely imbalanced datasets with few minority samples

Jingzhao Hu[1] · Hao Zhang[2] · Yang Liu[1] · Richard Sutcliffe[1] · Jun Feng[1]

Accepted: 15 June 2021 / Published online: 15 September 2021

Abstract

In recent years, Deep Neural Networks (DNNs) have achieved excellent performance on many tasks, but it is very difficult to train good models from imbalanced datasets. Creating balanced batches either by majority data down-sampling or by minority data up-sampling can solve the problem in certain cases. However, it may lead to learning process instability and overfitting. In this paper, we propose the Batch Balance Wrapper (BBW), a novel framework which can adapt a general DNN to be well trained from extremely imbalanced datasets with few minority samples. In BBW, two extra network layers are added to the start of a DNN. The layers prevent overfitting of minority samples and improve the expressiveness of the sample distribution of minority samples. Furthermore, Batch Balance (BB), a class-based sampling algorithm, is proposed to make sure the samples in each batch are always balanced during the learning process. We test BBW on three well-known extremely imbalanced datasets with few minority samples. The maximum imbalance ratio reaches 1167:1 with only 16 positive samples. Compared with existing approaches, BBW achieves better classification performance. In addition, BBW-wrapped DNNs are 16.39 times faster, relative to unwrapped DNNs. Moreover, BBW does not require data preprocessing or additional hyper-parameter tuning, operations that may require additional processing time. The experiments prove that BBW can be applied to common applications of extremely imbalanced data with few minority samples, such as the classification of EEG signals, medical images and so on.

Keywords Deep learning · Deep neural networks · Imbalanced dataset · Batch balance wrapper framework

1 Introduction

In machine learning, imbalanced data is a very common problem when training a classifier. A dataset is imbalanced if there is a difference between the number of training examples in one class and the number in the other class. This difference can lead to machine learning algorithms being biased towards predicting unseen samples as members of the majority class, a problem which also affects deep neural networks [1]. The degree of data imbalance can be measured by the *imbalance ratio*, defined to be the number of examples in the majority class divided by the number in the minority class. Even with the same imbalance ratio, the smaller the number of minority class samples, the harder it is for machine learning algorithms to learn effective distribution and classification bounds.

In traditional machine learning, imbalanced data has been well-studied [2, 3]. Approaches include oversampling [4], undersampling [3], cost-sensitive learning [5], and ensemble learning [6, 7]. Such methods have also been applied to deep learning [1]. For example, [8] report a GAN-based minority sample synthesis method following the paradigm of oversampling.

Recently, imbalanced dataset processing methods have been specifically designed for deep learning, for example those based on loss functions such as focal loss [9], gradient harmonizing [10], weighted softmax loss [11] and class imbalance loss [12]. Modified optimizer methods [13] are another technical solution, such as DM-SGD [14] and ABSGD [15]. The principle of all these methods is to adjust the gradient to allow the deep learning method to support

✉ Richard Sutcliffe
rsutcl@nwu.edu.cn

✉ Jun Feng
fengjun@nwu.edu.cn

[1] Department of Information Science and Technology, Northwest University, Xi'an 710127, China

[2] Graduate School, Shaanxi University of Chinese Medicine, Xianyang 712083, China

the imbalanced dataset. However, they all require data preprocessing or additional hyper-parameters. Additionally, most of them can only work in situations in which minority samples are not extremely scarce.

There is therefore a need for an end-to-end neural network method supporting extremely imbalanced datasets, that does not require data preprocessing or additional hyper-parameters, and that is also compatible with existing neural network systems. We propose the Batch Balance Wrapper framework (BBW) to meet this need: It is not sensitive to imbalance ratio, it learns well and it is extremely fast. In addition, it can handle situations in which there are only a few minority samples in the dataset.

In BBW, stratified sampling is first used to divide the training and test sets of an imbalanced dataset. Then, two new layers are added to the start of an existing neural network. The first layer performs an adaptive normalization of the input, improving the expressiveness of the minority sample distribution. The second layer diversifies the normalized samples, alleviating overfitting of the minority class. Finally, the neural network, complete with the two added layers, is trained using a Batch Balance (BB) algorithm which samples training data in such a way that data items in each batch are always balanced during the learning process. In this way, the learning method is not biased towards a certain class.

We also propose the evaluation metrics Valuable Convergence Ratio (VCR) and Average False-positive rate of Valuable Convergence (AFVC). These measures are intended to evaluate neural networks on an imbalanced dataset with few minority samples by using the 'leave-one-out' method for cross-validation. Here, the leave-one-out method is defined so that it only applies to minority samples.

We tested BBW on three imbalanced binary datasets with few minority samples, the CHB-MIT Scalp EEG Dataset (CHB-MIT) [16], the resampled University of Bonn EEG time series dataset (BonnEEG) [17], and the resampled First Affiliated Hospital of Xi'an Jiao Tong University Tuberculosis Chest Radiograph Dataset (FAHXJU) [18]. The maximum imbalance ratio reaches 1167:1 with only 16 positive samples, 200:1 with just 2 positive samples.

We carried out six experiments, all of which use the DenseNet121 architecture [19] as a baseline model. First, we demonstrated the feasibility of the proposed approach by training the baseline both with and without BBW. Second, we compared BBW to the baseline using the proposed modified leave-one-out method for cross-validation. Third, BBW was compared to six existing approaches (down sampling [3], oversampling [4], class weights [1], focal loss [9], weighted softmax loss (WSL) [11], and class imbalance loss (CIL) [12]) for training with unbalanced data. Fourth, we carried out an ablation study to show the contribution of different parts of the BBW framework. Fifth, we reran the second experiment using the normal definition of epoch. Sixth and finally, we measured the performance of the baseline, this time adopting the normal epoch definition and constraining learning counts to those used by BBW.

Experiment 1 illustrated the improved learning behavior of the BBW-wrapped neural network in comparison to the Baseline. Experiment 2 showed that BBW attained 14-40% higher VCR and 9-15% lower AFVC. Experiment 3 found that BBW was a better approach than downsampling, oversampling, class weights, focal loss, weighted softmax loss, and class imbalance loss. Experiment 4 demonstrated by ablation that each component of BBW improves the results, and that the overall BBW is better than its individual parts. Experiments 5 and 6 suggested that BBW is 16.39 times faster, while in addition the performance is better.

In this work, our primary contributions are:

- We propose the Batch Balance Wrapper Framework (BBW) for adapting general DNNs, allowing them to be well trained from extremely imbalanced datasets with few minority samples.
- We propose the input adaptive normalization layer, sample diversification layer, and batch balance strategy, which perform the mechanisms of trainable normalization, sample dynamic synthesis, and sample dynamic balance. They can improve the expressiveness of the sample distribution of minority samples and alleviate the overfitting of the minority class.
- BBW is not sensitive to imbalance ratio, and is extremely fast. It does not require data preprocessing, additional hyper-parameters, or even data normalization.
- We propose the evaluation measures Valuable Convergence Ratio (VCR) and Average False-positive-rate of Valuable Convergence (AFVC). Combined with the proposed minority-class-only leave-one-out cross validation, they can fairly evaluate neural networks on an imbalanced dataset with few minority samples.
- Using three different datasets, with balance ratio as high as 1167:1 (CHB-MIT) and as few as 2 positive examples (BonnEEG), we demonstrate that BBW achieves better performance compared to existing methods (downsampling, oversampling, class weights, focal loss, weighted softmax loss, and class imbalance loss).

2 Related work

Processing methods for imbalanced datasets have been well-studied in traditional machine learning. These methods can

be divided into two main groups, dataset preprocessing-based methods, and algorithm modification-based methods [20, 21]. The main idea of dataset preprocessing-based methods, such as oversampling and downsampling, is to try to preprocess the dataset to alleviate its imbalance. Zhang et al. [3] propose several multi-class imbalance learning methods. The oversampling method attempts to create more minority samples to alleviate the problem [4]. The simplest implementation is to randomly repeat minority samples until the dataset reaches an acceptable imbalance ratio. The downsampling method attempts to drop majority samples in order to balance the dataset. A more sophisticated approach is to delete majority samples which are far away from the classification boundary. An interesting development of the resample idea is the SMOTE method [22] and its variants [23, 24], which can create reasonable samples based on clustering theory and interpolation theory. Sun et al. [25] propose an imbalanced classification algorithm combining SMOTE and Support Vector Machines. This embeds SMOTE into the iteration of ADASVM-TW to synthesize samples during processing. Following the oversample paradigm, Zhou et al. [8] propose a sample synthesis method based on generative adversarial networks to augment minority samples of original imbalanced datasets.

Approaches based on algorithm modification attempt to improve the classification method to allow it to support imbalanced datasets, such as cost-sensitive learning [5] and the ensemble method [6, 7]. Cost-sensitive learning assigns different costs to the different misclassifications, thereby adjusting the classification results by minimizing the total cost. The ensemble method in imbalanced dataset processing divides the original dataset into a series of balanced subdatasets, then assembles all the sub-classifiers to boost the final result [26, 27]. Hayashi et al. [28] report an imbalanced learning algorithm focusing on the main class, based on a cluster-based zero-shot classifier. There are also some classification methods that are inherently insensitive to imbalance ratio, such as the decision tree method [29].

The above methods are designed for traditional machine learning, but they can also be used in deep learning. Buda et al. [1] report the effect of applying traditional imbalanced processing methods to deep neural networks, such as the paradigms of oversampling, downsampling, cost-sensitive learning, ensemble learning, etc. Taherkhani et al. [30] propose a transfer learning based multi-class imbalanced classification method by combining an adaptive boosting algorithm and neural networks. Pérez-Hernández et al. [31] describe binarization techniques on neural networks, which convert a multi-class task into several binary tasks to reduce multi-class imbalance problems. In recent years, some interesting imbalanced dataset processing methods specifically for deep learning have been developed. The methods can be categorized into two main groups, loss-based methods and optimizer-based methods. The principle of both categories is to adjust the gradient to let the deep learning method support imbalanced datasets. The loss-based method assigns different weights to each class/sample in order to adjust the loss. In recent work, this is the most widely followed method for neural networks supporting imbalanced data. The most successful instantiation of this idea is focal loss as proposed by [9], which can automatically decide the weights of each sample via predicted probability. Following this, Li et al. [10] propose the Gradient Harmonizing mechanism, which can adjust the gradient using gradient density. Jia et al. [11] propose weighted softmax loss, adaptively parameterized by maximum multi-class imbalance ratio. Zhang et al. [12] devise class imbalance loss to improve the cross-entropy loss on imbalanced datasets.

Optimizer-based methods modify neural networks in order to support imbalanced datasets. Zhang et al. [14] propose DM-SGD, supporting imbalanced datasets by actively selecting samples. Qi et al. [15] develop ABSGD, supporting imbalanced classification by weighting gradients based on an attention mechanism.

In conclusion, therefore, it should be noted that the above-mentioned methods require data preprocessing or additional hyper-parameters, which create the need for additional calculation or parameter tuning. There is a need for an end-to-end neural network method supporting imbalanced datasets, that does not require data preprocessing or additional hyper-parameters, and is compatible with all existing neural network systems. The method we propose can meet these needs, and will be described next.

3 Methodology

In this section, we will explain the proposed Batch Balance Wrapper framework (BBW). This allows deep neural networks to learn from extremely imbalanced datasets. BBW is designed to work with the Batch Balance algorithm (BB) and stabilize it. The framework is illustrated in Fig. 1.

As shown in Fig. 1, BBW has five parts that are organized in logical order. The first part of BBW is the stratified sampling method, which we use to divide the data into the training set and test set. After this, two new neural network layers are added to the start of the existing DNN. The first of these added neural network layers is the input adaptive normalization layer. The second is the sample diversity layer. After this comes the pre-existing classifier DNN whose training performance on imbalanced data we wish to improve. In our experiments, we use DenseNet121 [19] as an example, but it can be replaced with any DNN

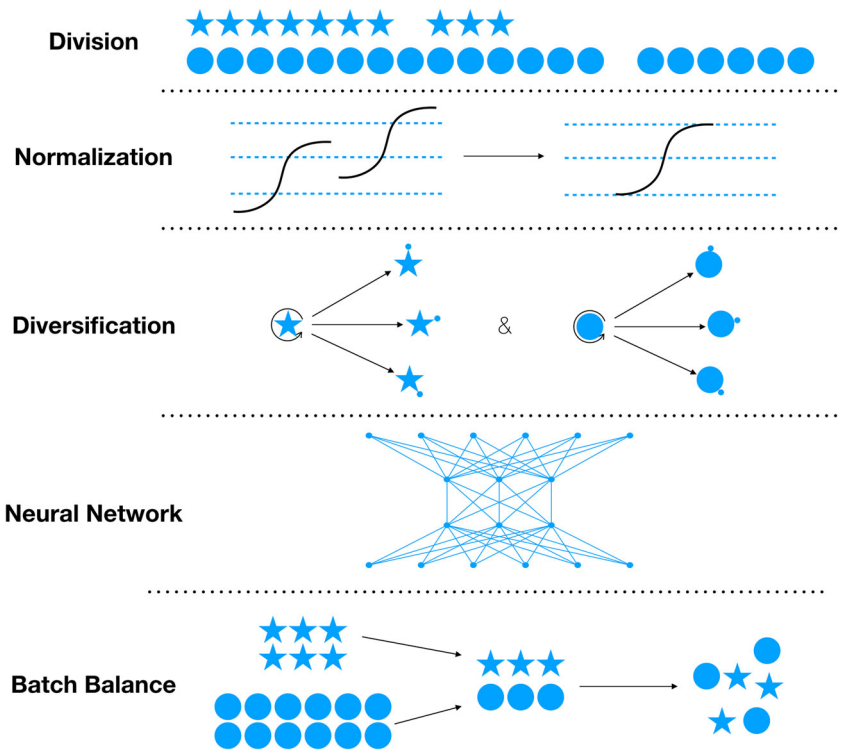

Fig. 1 Batch balance wrapper framework

which is a classifier. The final part of BBW is BB, which makes sure the samples in each batch are always balanced during the learning process. In the following subsections, we will show the details and implementation of each part of the framework.

3.1 Stratified sampling based dataset division

In a very imbalanced dataset that has a small number of minority samples, the random sampling method, which is normally used in a balanced dataset for dividing it into training and test sets, will cause the number of minority samples in the test set to be small, possibly even zero. This causes the evaluation metric applied to the test set to be invalid. Therefore, we use the stratified sampling method for division so that the test set can maintain the imbalance ratio of the original dataset. This allows the evaluation metric to be more credible, and avoids a situation in which it cannot be calculated from the test set.

In the implementation, we first shuffle the samples of each class. Then, we divide the shuffled samples into training subsets and test subsets using a fixed ratio, e.g. 7:3. Finally, we concatenate the samples of classes in the training subsets or test subsets to obtain a training set and a test set, and then shuffle them again.

It should be noted that if the number of minority samples is too few to divide, the leave-one-out method should be utilized to select the minority samples for the test set, and the imbalance ratio of the original dataset should be used to determine the number of majority samples in the test set. In this situation, majority samples should be randomly selected from the original dataset, and the rest should belong to the training set. These changes will ensure that the training set has enough minority samples for training. Additionally, this approach does not require too many models that need to be trained and evaluated, compared with the standard leave-one-out procedure. The details of this **modified leave-one-out method** are described in Algorithm 2 in Section 5.

3.2 Input adaptive normalization

In traditional input normalization, sufficient training data is always required, and it is assumed that this training data can correctly express the real distribution of test data. Therefore, the parameters of the normalization method can be determined from the training set, and those parameters are directly applied to the test samples. However, in an extremely imbalanced dataset, the number of minority samples may be too few to express the real distribution of the test data, even where the dataset is very large. If the parameters that we determined from the training set are directly used, the samples in the testing set will not be correctly normalized when the value of the testing set is too large, too small, or has a different statistical profile from the training set. In such cases, those inputs may produce an unrepresentative output.

Therefore, our intuition is that if the input normalization process can be trained together with the neural network,

the whole system will be more robust in cases where the minority samples cannot express the correct distribution. We refer to this as input adaptive normalization, because the neural network can simultaneously 'think about' the input normalization and feature extraction together. We believe that neural networks can better adapt to outliers in the testing set by using the normalization process and forward process together, even though the number of minority samples is too few to express the correct distribution.

To implement this process, a parameterizable normalization method [32] is used as the first layer of the neural network. Equations (1) to (3) show the principle of input adaptive normalization, where x_i denotes the input of the adaptive normalization layer, μ is the mean of the inputs, σ^2 is the variance, w and b are the trainable parameters of the linear transformation, and y_i denotes the output of the layer. The process can be described as normalizing x into the mean of zero and the variance of 1, then multiplying x by the linear trainable parameters. Therefore, we can use the input adaptive normalization layer to normalize the input of the neural network, and allow its trainable parameters to learn the distribution of the input data. Most importantly, those parameters will be trained together with the neural network.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{2}$$

$$y_i = w\frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + b \tag{3}$$

As an alternative to (1) to (3), other trainable normalization methods could also be considered. However, this approach works well with our implementation.

3.3 Sample diversification

In BB, the minority samples will be used many more times than the majority samples throughout the entire training process. This will cause the neural network to overfit the minority class before the majority class is properly learned. This could also cause the loss to be insensitive to the minority class, because the majority class, which is reused less times, can always provide a larger loss. Therefore, our idea is to apply a random transformation to the input data that are reused, to ensure that they are not exactly the same when they are fed into the neural network. Hence, we use a noise function as the second layer of the network to perform a random transformation of the adaptively normalized input data, which is the reason we refer to it as the sample diversification layer.

In the implementation, we select the Gaussian noise function as the sample diversification function, which adds zero-centered, one-variance Gaussian noise to the normalized input data. Equation (4) shows this principle, where X is the adaptively normalized input data, X' is the noised normalized input data, and R is a random tensor that obeys the Gaussian distribution.

$$X' \leftarrow X + R, \quad where \quad R \sim \mathcal{N}(0, 1) \tag{4}$$

It should be noted that if the noise is too complex, it will inhibit the neural network's ability to learn the patterns in the original data. Similarly, if the noise is too simple, such that it is easy for the neural network to find the patterns of the noise, the noise layer will lose the ability to diversify samples. We believe that the best choice could be for the noise to be changed over time, because it would then be hard for the neural network to find such patterns. However, in this work, the Gaussian noise is sufficient for our implementation.

3.4 Batch balance

We think that there are two problems that restrict the training of neural networks on extremely imbalanced datasets. The first problem is the invalid sampling problem which is the probability that there is no minority sample in a batched training set. The second problem is that the majority samples in the training set can always provide a much larger total loss than the minority samples, because majority samples are much larger in number than minority samples. Our solution is to keep the sample balanced in the batched training set, to keep the loss fair in different classes. BB achieves this, working after the side effects of forcing the data to be balanced are solved by the BBW framework. The batch, here, is defined as the sample set that is fed to a deep neural network in one gradient-updating iteration.

Algorithm 1 Batch balance algorithm in one Epoch.

Input: minorityTrainingSet, minorityTrainingSetLabel, majorityTrainingSet, majorityTrainingSetLabel, batchSize
Output: modelOfThisEpoch
for *number of batch in one epoch* **do**
  Randomly sample batchSize/2 samples from minorityTrainingSet without replacement ;
  Randomly sample batchSize/2 samples from majorityTrainingSet ;
  Concatenate those two subsets sampled by above steps to get batched training set ;
  Shuffle batched training set and find the corresponding label of each sample ;
  Train neural network using shuffled batched training set ;
end

Algorithm 1 shows the process of BB. The algorithm works in one 'epoch'. Normally, an epoch is defined as a stage of training in which all training samples are fed to a neural network. However, in BB, we define one epoch to be a stage of training in which all minority training samples are fed to the network; we use this definition because the number of majority samples of the training set is decided by the number of minority samples in Algorithm 1.

The meaning of the input and output variables in Algorithm 1 can be interpreted literally. The algorithm is defined on an epoch and returns a model trained on the current epoch. In one batch of an epoch, we first randomly sample $\frac{batchSize}{2}$ samples from the minority training set without replacement to obtain the batched minority training subset. Then, we randomly sample $\frac{batchSize}{2}$ samples from the majority training set to obtain the batched majority training subset. Next, we concatenate the batched minority training subset and batched majority training subset together to obtain the batched training set. Then, we shuffle the batched training set, and find the corresponding label of each sample. Finally, the shuffled batched training set is fed to the neural network to finish the training process of the current batch.

Something else of note in Algorithm 1 is that the number of minority samples in the training set should be divisible by $\frac{batchSize}{2}$. The algorithm works with a binary classification task; if we want BB to work with a multi-classification task, we just need to sample data from every class and change $\frac{batchSize}{2}$ to $\frac{batchSize}{numberOfClasses}$.

Finally, we will outline the BBW framework as a whole (Fig. 1). In BB, we can deduce that the minority samples will be reused many more times than the majority samples before the training processes converge. Therefore, BBW is designed to stabilize the training process. From the perspective of the minority class, BBW looks like an upsampling technique. However, from the perspective of the majority class, BBW is a downsampling one. The diversification layer is responsible for sample synthesis, but the BB process carries out class-based downsampling. The input adaptive normalization layer normalizes the input, and cooperates with the feature extracting process. BBW is thus a combination of components which are able to cooperate with each other. It is designed for an extremely imbalanced dataset.

4 Datasets

4.1 CHB-MIT scalp EEG dataset (CHB-MIT)

The CHB-MIT Scalp EEG Dataset [16] is an ElectroEncephaloGram dataset with the task of detecting the occurrence of epilepsy from an EEG signal. This dataset was chosen to evaluate BBW because the multichannel brainwave data is imbalanced, and in particular the minority samples are extremely lacking. Another benefit is that the patient specificity of EEG is significant, which can fully test BBW on various situations of learning patterns and let us deeply analyze the principles behind BBW. Figure 2 shows the schematic diagram of CHB-MIT. One blue wavy line represents the brainwave of one channel recorded against time, and there are 23 channels. The time interval marked in red represents the occurrence of epilepsy during this time. The $S_n^S - S_n^E$ pair in the figure represents the start and end points of the seizure.

There are 24 cases in this dataset, but cases 12 and 13 were excluded because they frequently changed their channel definition during the recording. In each case, there are dozens of hours of EEG data, but only a few minutes of seizure onset EEG data. The dataset is thus extremely imbalanced and has few minority samples, making it an ideal one on which to test BBW.

This seizure detection task is a typical classification problem, but the data is successive. To segment the data into fixed lengths, we used the non-overlapping sliding window method to cut continuous data, as depicted in Fig. 3. t_W is the sliding window size, and W_n represents the data fragment n denoted by window n. We labeled the data fragment $True$ if the data fragment contained any seizure in the time window, otherwise we marked the data fragment $False$.

In this work, we set t_W to 30 seconds. The statistical information of the fragments after selecting the data is shown in Table 1. There are thousands of samples, but few of them are positive samples. The imbalance ratio can be as high as 1167.31:1. There are as few as 8 positive samples. It can therefore be said that this dataset is extremely imbalanced, and has few minority samples.

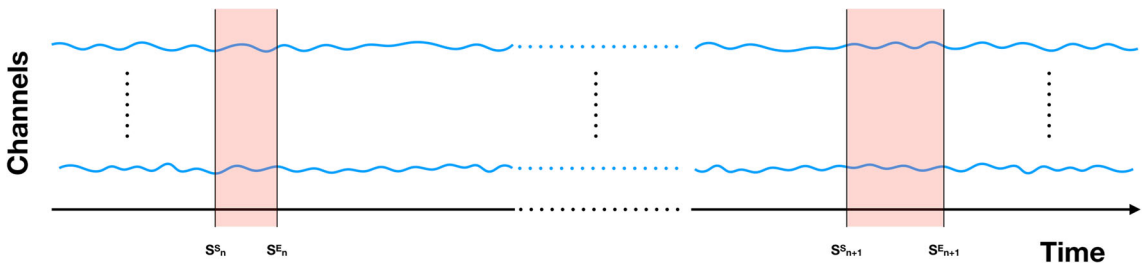

Fig. 2 CHB-MIT dataset schematic

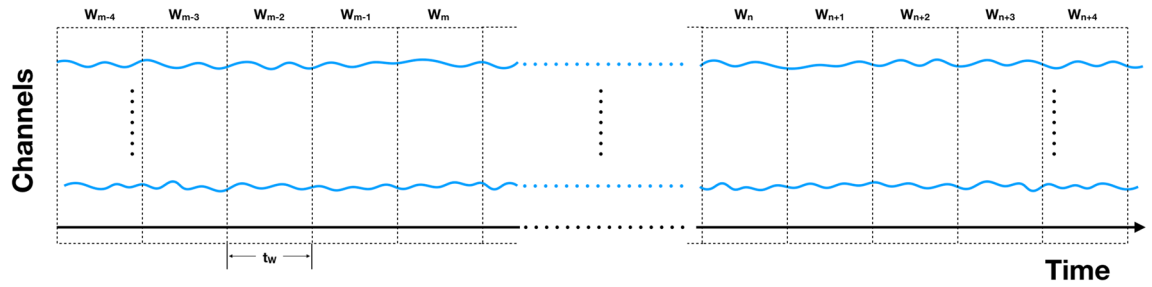

Fig. 3 Non-overlapping sliding window

BBW is compatible with all types of neural networks, and we use the convolutional neural network in this task. To feed the data fragments to the CNN, we must express them in matrix or tensor form. A data fragment is already a matrix with a time and channel axis, but it only shows the time and spatial domain information. To expose the frequency domain information that is hidden in the data fragment, we used the short-time Fourier transform (STFT) to calculate the spectrogram of each channel in the data fragment, as shown in Fig. 4. By stacking the spectrogram of each channel, we can convert the data fragment into a 3D tensor. The three dimensions of this 3D tensor are time, frequency, and channel. It can express the time domain, frequency domain, and spatial domain (channel) information at the same time. Now, we believe that the data is clear enough for CNN to learn its features.

Table 1 Data fragment statistical information for each case

Case No.	Total number of samples	Number of positive samples	Imbalance ratio
01	4825	23	208.78
02	4196	8	523.50
03	4523	18	250.28
04	18693	16	1167.31
05	4642	23	200.83
06	7992	17	469.12
07	8030	13	616.69
08	2381	35	67.03
09	8130	12	676.50
10	5980	21	283.76
11	4022	30	133.07
14	3094	14	220.00
15	4643	86	52.99
16	2023	9	223.78
17	2381	13	182.15
18	4121	16	256.56
19	3444	11	312.09
20	3286	15	218.07
21	3907	11	354.18
22	3690	10	368.00
23	3180	20	158.00
24	2534	26	96.46

4.2 Bonn EEG Dataset (BonnEEG)

The University of Bonn EEG time series dataset (BonnEEG) was released by [17] for EEG-based epilepsy detection. It contains 500 EEG samples, labeled by five related subject statuses. The EEG data were collected from five healthy human subjects and five human subjects with epilepsy. All EEG segments were single-channel signals with an acquisition duration of 23.6 seconds, with 4,097 sampling points and a sampling rate of 173.61 Hz. The epilepsy detection task is abstracted as a binary classification task. To emphasize the extreme imbalance and the few minority samples, the dataset is resampled to only 2 minority samples (epilepsy) with 400 majority samples (normal).

4.3 First affiliated hospital of Xi'an Jiao Tong University Tuberculosis chest radiograph dataset (FAHXJU)

The FAHXJU dataset was collected at the First Affiliated Hospital of Xi'an Jiao Tong University [18]. It contains 1,403 chest radiographs, labeled by the types of tuberculosis. The task is to classify the two types of tuberculosis from the chest radiograph, which is a binary classification problem. There are 1,345 majority samples (cavity) and only 58 minority samples (exudation) in the dataset.

5 Evaluation

In extremely imbalanced datasets with few minority samples, those samples are too few to evaluate the performance of the method. In this situation, the leave-one-out method is the only appropriate method to conduct the cross-validation. However, this method is very time consuming, especially in a situation in which the total

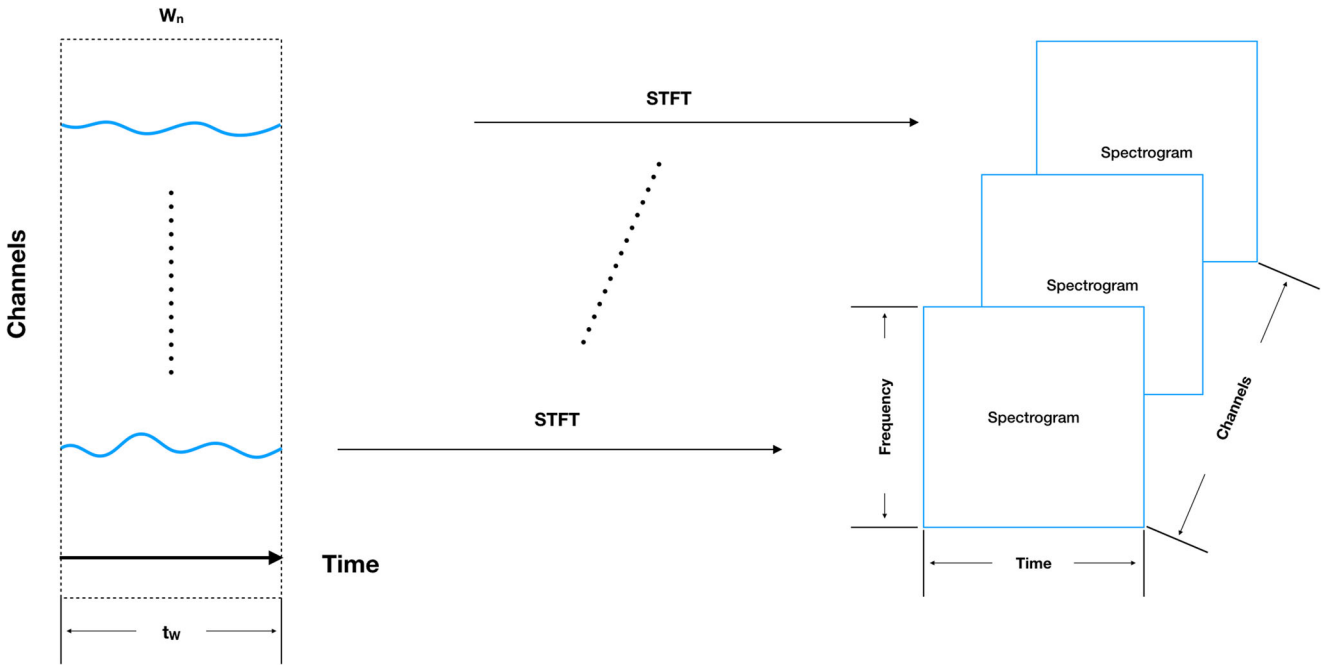

Fig. 4 Using a short-time Fourier transform to expose frequency domain information

number of samples is very large. In addition, it is hard to calculate evaluation metrics such as F-measure, P-R curve, ROC curve, and AUC for extremely imbalanced data when using the leave-one-out method.

Our solution is to only use the leave-one-out method for the minority class, and to randomly sample the majority samples according to the imbalance ratio. Then, the leave-one-out positive sample and randomly sampled negative sample set are concatenated to obtain the test set. Finally, we calculate the average evaluation score of all leave-one-out models, as in the normal leave-one-out process. The detailed core process is shown in Algorithm 2. $leaveOneOutPositiveSample$ is the minority sample which we 'left out'.

The evaluation measures we chose were True Positive Rate (TPR) and False Positive Rate (FPR). The calculations are shown in (5) and (6), where TP is the number of true positive examples, P is the total number of positive examples, FP is the number of false positive examples, and N is the total number of negative examples.

$$TPR = \frac{TP}{P} \tag{5}$$

$$FPR = \frac{FP}{N} \tag{6}$$

Algorithm 2 Minority-class-only leave one out method.

Input: leaveOneOutPositiveSample, imbalanceRatio, minoritySampleSet, majoritySampleSet
Output: TPR, FPR

```
begin
    Randomly sample ⌈imbalanceRatio⌉ samples from the majoritySampleSet, and leave the rest for the training set ;
    Concatenate the leaveOneOutPositiveSample and the subset sampled by the above step to get the test set ;
    Concatenate the minoritySampleSet without the leaveOneOutPositiveSample and the subset left by the above step to get the training set ;
    Train the neural network model using the training set ;
    Calculate the confusion matrix of the neural network model in this turn using the testing set ;
    Calculate the TPR and the FPR ;
end
```

Although it should be easy to calculate the average TPR and FPR of all leave-one-out models, a flaw concerning the stop criterion of this method is that TPR = 1 can always appear during the epochs of the training process, which may

lead to bias in our evaluation process. This is because it will improperly pull up the FPR if TPR = 1 only occurs at the earlier epochs. Actually, in this situation, instead of triggering the stop criterion, we prefer to consider that the model cannot correctly predict the leave-one-out sample, and believe that the neural network does not converge at all. Therefore, we defined a filter to handle this situation, which we refer to as Valuable Convergence. A valuable convergence must satisfy TPR = 1 and FPR <0.5. The condition TPR = 1 means the model must correctly predict the leave-one-out sample, which is the only one positive sample in the test set. FPR <0.5 means the predictive ability for negative samples in the test set is just better than random prediction.

Next, we use the Valuable Convergence Ratio (VCR) and Average FPR of Valuable Convergence (AFVC) to evaluate the performance of the learning method. The formulas for VCR and AFVC are shown in (7) and (8), where N_{vc} is the number of valuable convergence, N_{lm} is the number of leave-one-out models, N_{ms} is the number of minority samples in dataset, and each FPR_{vc} is the FPR value of a valuable convergence. This evaluation method is suitable for datasets that are extremely imbalanced and have only very few minority samples.

$$VCR = \frac{N_{vc}}{N_{lm}} = \frac{N_{vc}}{N_{ms}} \tag{7}$$

$$AFVC = \frac{\sum FPR_{vc}}{N_{vc}} \tag{8}$$

In general, the VCR and AFVC measures are defined to evaluate models fairly on such a dataset with few minority samples when using the leave-one-out method. Furthermore, the leave-one-out method is modified to only apply to minority samples.

6 Experiments

6.1 Outline

We carried out six experiments to demonstrate the effectiveness of our proposed BBW framework for learning effectively with extremely imbalanced data containing few minority samples. Experiment 1 compares the learning behavior of the DenseNet121 [19] neural network architecture when used with BBW and without. Experiment 2 follows the same setting, but the test set is created using the modified leave-one-out method proposed in Section 5. Experiment 3 measures the performance of six previous methods for dealing with imbalanced data containing few minority samples. Each is trained in an identical setting so that results may be compared directly with that of BBW in Experiment 2. Experiment 4 is an ablation study to show the effect on training of components within BBW. Experiment 5 repeats the setting of Experiment 2 but uses a different definition of epoch. Finally, Experiment 6 measures the performance of DenseNet121, once again using the standard definition of epoch, but this time restricted to the same learning counts as BBW in Experiment 2.

6.2 Experiment 1 - verification of BBW learning

The aim was to test the learning abilities of BBW by comparing it with a baseline implementation, in order to demonstrate the differences in learning behavior between them.

Only the CHB-MIT dataset was used for this experiment. We set both training and test sets to the whole dataset, then calculated the True Positive Rate (TPR) and False Positive Rate (FPR) of the test set to observe the learning ability of each learning method. We did not use the TPR and FPR that were calculated from the training set because there are differences in the behaviors of some neural network structures in the training process and test process, such as the noised layer and dropout layer, which are only activated in the training process. By setting the training set and test set to the same dataset, we can obtain more objective TPR and FPR to describe the learning ability of each learning method.

The basic architecture used for the experiment is a modified DenseNet121 [19], which is an important convolutional neural network backend for automatic feature extraction. This is then wrapped by the BBW framework. As a baseline for comparison, the DenseNet121 network is used alone, with no BBW wrapping. For the test to be fair, the baseline model used the same parameters of batch and epoch as the BBW version. We also normalized the input data with a min-max method for DenseNet121 alone, even though the wrapped version did not require this step.

The resulting learning ability just for Case 1 (the first case in dataset) is shown in Fig. 5. In the figure, we can observe that the TPR and FPR of both models fluctuate up and down at the early stage. However, from about the 40th epoch, the TPR and FPR of the baseline DenseNet121 both go down with occasional fluctuations, which indicates that the majority class has begun to overwhelm the minority class. In other words, the total loss of majority samples is much larger than the total loss of minority samples, which results in the neural network preferring the majority class in order to ensure that the total loss of each batch can be lower. By contrast, the TPR of the BBW-wrapped model gradually approaches 1, and FPR gradually approaches 0. This demonstrates the good learning ability of BBW.

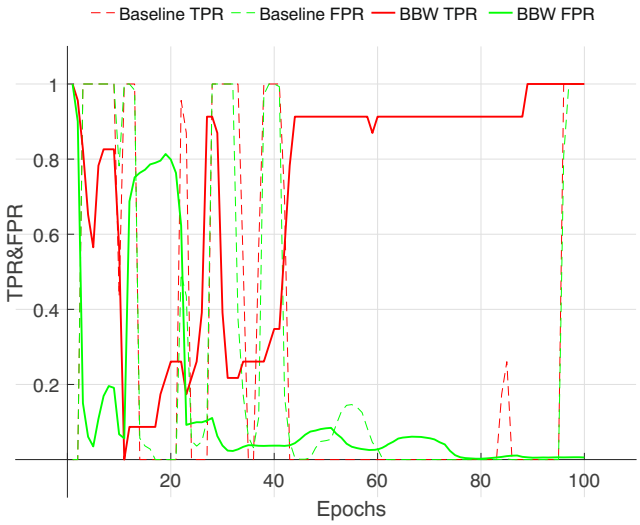

Fig. 5 Experiment 1: Learning ability (Case 1 from CHB-MIT dataset shown)

6.3 Experiment 2 - BBW performance using leave-one-out

The aim was to demonstrate the better performance of BBW when compared to the baseline. However, in this case the test set was created using the modified leave-one-out method discussed earlier.

All three datasets were used: CHB-MIT, BonnEEG and FAHXJU. This time, performance metrics were VCR, AFVC (both discussed earlier), and Time.

The general setup was very similar to that of Experiment 1, i.e. DenseNet121 wrapped with BBW was compared with DenseNet121 alone. However, the test set in this experiment was created by the modified leave-one-out method proposed in Section 5.

The results can be seen in Table 2. We can observe that the results are much better than those of the baseline on all three datasets. It is as expected, since Experiment 1 showed that a neural network wrapped by BBW has better learning ability in extremely imbalanced datasets with few minority samples.

More detailed results for CHB-MIT can be seen in Table 3, itemized for the 24 cases. Figures given are the average performances of the leave-one-out models in each case.

We can observe in the table that almost all of BBW's results are much better than those of the baseline, except for Case 19 in epoch 300. This might be because the pattern in Case 19 is simpler, leading to the over-fitting of our method. We know from Fig. 5 that the learning ability curve of controlled Experiment 2 is more unstable. The unpredictable fluctuations might have resulted in better results in Case 19 for the Experiment 2 baseline model. Overall, the average VCR of BBW is 14-40% higher than the baseline, and the average AFVC of our method is 9-15% lower.

We can also observe that the results in Table 3 of some cases in the baseline model are better and better with the increase of epoch. Indeed, the average performance of the baseline model becomes better when the epoch increases. This is because patterns in the data are not very difficult for a network to learn, and the performance is reliant on the imbalance ratio of the dataset. The larger the imbalance ratio, the more epochs are needed. If the patterns are not easy for the neural network part of the BBW framework, the performance will not become better, even if the number of epochs is increased. This can be observed in Cases 16, 17, etc. However, the performance of BBW is independent of the imbalance ratio. In other words, BBW is insensitive to the ratio. Additionally, BBW converges more quickly, and is more stable and more robust.

6.4 Experiment 3 - comparison of BBW with existing methods

The aim was to compare the performance of BBW with six popular imbalanced data processing methods, using the same datasets and the same underlying model: down

Table 2 Experiment 2: BBW and baseline DenseNet121 compared - All datasets

Dataset	Epoch = 100				Epoch = 200				Epoch = 300			
	BBW		Baseline		BBW		Baseline		BBW		Baseline	
	VCR	AFVC	VCR	AFVC	VCR	AFVC	VCR	AFVC	VCR	AFVC	VCR	AFVC
CHB-MIT	93.74	5.124	53.88	19.72	95.59	3.127	72.75	16.82	95.96	3.314	81.93	12.61
BonnEEG	100.0	00.50	50.00	21.00	100.0	00.75	50.00	42.50	100.0	00.75	100.0	35.00
FAHXJU	65.52	18.97	17.24	27.08	81.03	18.44	39.66	24.64	87.93	16.99	51.72	26.11
Average	86.42	08.20	40.37	22.60	92.21	07.49	54.14	27.99	94.63	07.02	77.88	24.57
Time	7.345h		7.493h		9.740h		10.93h		11.76h		13.97h	

Table 3 Experiment 2: BBW and baseline DenseNet121 compared - CHB-MIT dataset

Case No.	Epoch = 100				Epoch = 200				Epoch = 300			
	BBW		Baseline		BBW		Baseline		BBW		Baseline	
	VCR	AFVC	VCR	AFVC	VCR	AFVC	VCR	AFVC	VCR	AFVC	VCR	AFVC
01	95.65	01.39	73.91	21.11	95.65	01.13	91.30	12.37	91.30	00.11	95.65	07.29
02	100.0	04.44	75.00	30.15	100.0	03.72	62.50	19.77	100.0	01.34	75.00	17.24
03	100.0	01.22	50.00	14.62	100.0	00.09	94.44	18.73	100.0	00.20	100.0	22.69
04	75.00	06.20	68.75	28.99	87.50	04.98	68.75	22.98	81.25	03.00	68.75	13.96
05	95.65	01.52	73.91	10.56	100.0	02.92	95.65	09.41	100.0	02.49	100.0	04.24
06	100.0	20.13	29.41	22.94	94.12	06.65	58.82	23.81	100.0	10.58	58.82	22.15
07	92.31	05.00	23.08	26.20	100.0	03.27	53.85	27.53	100.0	03.57	61.54	21.88
08	94.29	01.07	22.86	17.10	97.14	01.30	48.57	10.47	94.29	01.11	65.71	08.76
09	91.67	00.36	41.67	33.77	91.67	00.15	41.67	21.30	91.67	00.16	41.67	12.88
10	100.0	02.45	61.90	13.84	95.24	00.44	95.24	03.54	100.0	00.70	95.24	02.01
11	96.67	00.90	80.00	04.82	96.67	00.03	100.0	06.04	96.67	00.08	93.33	01.68
14	85.71	12.73	35.71	24.45	100.0	07.34	71.43	19.82	92.86	04.34	92.86	13.71
15	90.70	02.18	65.12	16.41	91.86	01.70	84.88	09.67	94.19	01.12	90.70	08.49
16	100.0	23.61	55.56	28.13	88.89	11.16	22.22	17.86	100.0	18.80	66.67	21.43
17	100.0	03.03	61.54	14.00	100.0	02.56	84.62	17.14	100.0	01.43	76.92	10.71
18	100.0	10.38	37.50	14.07	100.0	03.21	62.50	23.23	100.0	08.85	81.25	17.63
19	72.73	00.16	54.55	04.70	81.82	01.03	81.82	13.64	72.73	00.20	90.91	12.08
20	93.33	01.89	26.67	24.32	100.0	04.26	66.67	26.53	100.0	02.98	73.33	09.38
21	100.0	06.43	36.36	41.62	100.0	08.07	63.64	25.03	100.0	04.23	81.82	23.54
22	90.00	04.23	70.00	11.26	90.00	00.51	80.00	17.49	100.0	01.60	100.0	09.27
23	100.0	00.98	65.00	16.65	100.0	00.79	95.00	12.69	100.0	00.57	100.0	06.65
24	88.46	02.42	76.92	14.02	92.37	03.48	76.92	10.93	96.15	05.44	92.31	09.75
A[a]:	93.74	5.124	53.88	19.72	95.59	3.127	72.75	16.82	95.96	3.314	81.93	12.61
T[b]:	6.4644h		6.6115h		8.4889h		9.7028h		10.2394h		12.3912h	

[a] The average of each column

[b] The total process time of each experiment

sampling [3], oversampling [4], class weights [1], focal loss [9], weighted softmax loss (WSL) [11], and class imbalance loss (CIL) [12].

All three datasets were used: CHB-MIT, BonnEEG and FAHXJU. The performance metrics were VCR, AFVC, Precision, and F1.

The six methods were implemented by adapting standard code and then trained using the DenseNet121 model. All settings were the same as Experiment 2.

Results across all three datasets are shown in Table 4 for Epoch 100. We will compare the results with those for BBW in Table 2. Table 4 indicates that downsampling is powerless in this situation; this is as expected, because there are too few samples for training after downsampling. Class weights, focal loss, WSL, and CIL even had the opposite effect compared with the baseline in Experiment 2 because they are not designed for this extremely imbalanced situation. Oversampling is a challenging competitor; it does improve the performance compared with the baseline in Experiment 2. However, there still exists a large performance gap compared with BBW in Experiment 2. The results do not prove that these popular methods do not work; they just tell us that they cannot work well on extremely imbalanced datasets with few minority samples because they are not designed for such an extreme situation. Our method works just because it is specially designed for such datasets.

6.5 Experiment 4 - Ablation study on BBW

The aim was to measure the performance of BBW when certain components of it are removed, in order to demonstrate their individual contributions.

The CHB-MIT dataset was used, and the performance metrics were VCR, AFVC, Precision, and F1.

The basic setup was the same as in Experiment 2. Four model configurations were trained – the baseline

Table 4 Experiment 3 - BBW and previous methods compared

	Precision*			F1*			VCR*			AFVC*		
	CHB-MIT	BonnEEG	FAHXJU	CHB-MIT	BonnEEG	FAHXJU	CHB-MIT	BonnEEG	FAHXJU	CHB-MIT	BonnEEG	FAHXJU
Downsampling [3]	79.36	-	80.00	19.66	-	06.61	11.22	00.00	03.45	-	-	25.00
Oversampling [4]	86.46	81.30	80.24	73.70	61.92	30.69	64.22	50.00	18.97	15.66	23.00	24.62
Class Weights [1]	84.40	-	77.42	54.84	-	30.47	40.61	00.00	18.97	18.48	-	29.17
Focal Loss [9]	82.58	96.62	77.73	58.86	65.90	23.42	45.73	50.00	13.79	21.09	03.50	28.65
WSL [11]	85.75	84.75	78.11	58.96	62.89	30.53	44.93	50.00	18.97	16.62	18.00	28.03
CIL [12]	81.49	70.42	81.45	41.48	58.48	39.26	27.82	50.00	25.86	22.71	42.00	22.78
BBW	95.13	99.50	84.05	94.43	99.75	73.64	93.74	100.0	65.52	5.124	00.50	18.97

* Average of all cases

DenseNet121 from Experiment 2, baseline with BB added, baseline with adaptive normalization & diversification, and finally the complete BBW system.

The results are shown in Table 5, which demonstrates the gain in performance caused by the components of BBW. The table also shows that the performance of the complete BBW is better than its individual components. Firstly, we observe that adding BB to the baseline DenseNet121 slightly improves the classification performance. This is because BB arbitrarily corrects the imbalance, but it inevitably leads to instability in the learning process and overfitting of the minority class. Adding just the input adaptive normalization layer and the sample diversification layer does not significantly improve the classification performance, since the training is still extremely imbalanced. However, we see a clear improvement when wrapping the model with the full BBW framework. The result of the ablation study meets our expectations.

6.6 Experiment 5 - Normal definition of epoch

The aim was to investigate how imbalance ratios and patterns of samples influence the loss to affect the final learning results. The normal definition of epoch was used. In all other respects, the model was the same as the baseline in Experiment 2. The CHB-MIT dataset was used, and the performance metrics were VCR, AFVC, and Time.

The results of this experiment when epoch = 100 are shown in Table 6. We can see that behaviour is much more unstable, but some typical scenarios can still be analyzed by comparing them with other experiments. By comparing Tables 6 and 3, it is apparent that more than $\frac{2}{3}$ of the VCR in Experiment 2 (BBW) is better than the VCR of Experiment 5. Additionally, almost all of the AFVC in Experiment 2 is lower than that of Experiment 5 in the additional $\frac{1}{3}$. There are only four cases in Experiment 5 in which the results are better than those in Experiment 2, if we relax the constraints on epoch. Even so, the total processing time of Experiment 5 is 16.39 times longer than the total processing time of Experiment 2, i.e. BBW is 16.39 times faster.

In Table 6, we can observe that some cases converge sufficiently (such as Case 22), but some cases cannot converge to an accepted result (such as Case 6). This phenomenon can be explained as follows: The learning method will learn the patterns of the majority class quickly if the patterns are easy. The total loss of majority samples will be less than the total loss of minority samples, which have not been well-learned. While the learning process is occurring, the discriminative ability of the minority class will catch up with the discriminative ability of the majority class. The learning method will then report an accepted

Table 5 Experiment 4: Ablation experiments on BBW

Configuration	Precision	F1	VCR	AFVC
Baseline[a]	83.53	65.51	53.88	19.72
Baseline[a] + BB[d]	85.40	71.49	61.48	17.09
Baseline[a] + IAN[b] + SD[c]	81.22	65.77	55.26	23.12
Baseline[a] + IAN[b] + SD[c] + BB[d] (i.e. BBW)	95.13	94.43	93.74	5.124

[a] DenseNet121
[b] Input Adaptive Normalization
[c] Simple Diversification
[d] Batch Balance

convergence at some epoch. If the patterns of samples are a little difficult for the learning method, the total loss of the majority class will be greater than the total loss of the minority class, even if the learning method has already learned the majority class preferentially. In fact, the learning method will always try to learn the majority class first, because it can always provide a larger total loss at the beginning of the learning process. Another situation that can lead to learning failure is when the patterns of the samples are too easy for the learning method, leading to the premature overfitting of the majority class. To solve these problems, BB is stabilized by the BBW framework, and attempts to balance 'by force' the number of samples in each class in one batch, and also to balance the total loss of each class in one batch to ensure the learning method does not bias towards some class.

6.7 Experiment 6 - Normal epoch definition and training counts limited

The aim was to measure the performance of the baseline in Experiment 2 when using the normal definition of epoch and also limiting sample learning times.

The CHB-MIT dataset was used. The performance metrics were VCR, AFVC, and Time.

After conducting the previous experiment, we were curious about the performance of a DenseNet121 model that has the normal epoch definition and the same total sample learning times as in Experiment 2 for BBW. We ensured that the total sample learning times in this experiment were the same as those of Experiment 2 by controlling the epoch. The epoch adjustment formula is calculated by (9) and (10), where N_s is the number of all samples in dataset, r_i is the imbalance ratio of the dataset, and r_a denotes the epoch adjustment ratio computed independently for programming convenience. The *epoch* in the formula is the epoch of Experiment 2, and $epoch_{new}$ is the epoch for this experiment, to ensure they have the same total sample learning times. The ceiling function used here is to prevent epoch = 0.

$$r_a = \frac{N_{ms} \times 2 - 2}{N_s - 1 - r_i} \tag{9}$$

$$epoch_{new} = \lceil r_a \times epoch \rceil \tag{10}$$

Table 6 Experiment 5: Baseline using normal epoch definition

Case No.	VCR	AFVC	Case No.	VCR	AFVC
01	100.0	07.55	15	90.70	05.88
02	100.0	05.18	16	66.67	05.51
03	94.44	06.54	17	100.0	12.57
04	93.75	05.98	18	100.0	08.05
05	91.30	01.42	19	90.91	15.61
06	64.71	08.88	20	100.0	04.35
07	100.0	13.49	21	81.82	08.48
08	68.57	09.38	22	100.0	00.30
09	83.33	11.82	23	100.0	06.80
10	95.24	00.79	24	100.0	13.40
11	96.67	01.42	Average	90.76	07.76
14	78.57	17.31	Time	105.9h	

Table 7 Experiment 6: DenseNet121 using normal epoch definition and same total sample learning times as for BBW in Experiment 2

Case No.	E *=[a] 100		E *=[a] 200		E *=[a] 300	
	VCR	AFVC	VCR	AFVC	VCR	AFVC
01	13.04	18.02	21.74	27.37	04.35	26.79
02	00.00	–	00.00	–	25.00	23.28
03	16.67	17.20	11.11	46.40	11.11	34.20
04	00.00	–	06.25	17.81	00.00	–
05	00.00	–	26.09	21.31	17.39	07.71
06	05.88	20.64	00.00	–	00.00	–
07	00.00	–	23.08	22.37	07.69	41.00
08	00.00	–	00.00	-	11.43	29.41
09	00.00	–	16.67	24.74	00.00	–
10	04.76	03.17	14.29	16.90	47.62	04.08
11	36.67	19.27	56.67	15.58	70.00	19.15
14	07.14	03.18	00.00	–	14.29	30.68
15	12.79	22.64	34.88	09.87	58.14	20.45
16	00.00	–	11.11	20.54	00.00	–
17	15.38	06.28	15.38	24.04	38.46	19.78
18	06.25	04.67	06.25	36.58	25.00	31.23
19	09.09	22.44	18.18	14.58	18.18	07.85
20	06.67	42.92	00.00	–	06.67	43.84
21	00.00	–	00.00	–	09.09	13.80
22	10.00	36.96	00.00	–	30.00	33.51
23	35.00	09.40	20.00	23.10	35.00	18.44
24	07.69	42.27	30.77	21.65	34.62	17.18
Average	08.50	–	14.20	–	21.09	–
Time	6.032h		7.188h		8.422h	

[a] Total sample learning times are equal to the corresponding epoch

The results are shown in Table 7. We can observe that the system did not conduct valuable learning at all. However, the performance improved with the increase in the epoch. This indicates that the learning method is still in the early stage of the learning process. It again proves the efficiency of BBW. The total processing time here is slightly less than that for Experiment 2 because it has fewer cross-validation steps.

Lastly, It should be noted that almost all traditional machine learning and deep learning methods that support imbalanced datasets require either data preprocessing or additional hyper-parameters. Our method is an end-to-end method that supports imbalanced datasets, and does not require additional data preprocessing or hyper-parameters. Additionally, BBW displayed the best performance in the extreme situation. The experiments designed in this work prove the ability of the proposed BBW framework to perform well on extremely imbalanced datasets with few minority samples, and verify its core principles.

7 Conclusion

In summary, the proposed BBW framework can adapt general DNNs to be trained better on extremely imbalanced datasets with few minority samples. In essence, BBW performs downsampling of majority samples, and oversampling of minority samples. In addition, it carries out sample synthesis within the sample diversification layer. The input adaptive normalization layer in BBW allows DNNs to perform the normalization process automatically and natively. Moreover, BBW does not require data preprocessing or

additional hyper-parameters, not even data normalization. Experimental results in this paper demonstrate the performance and efficiency of BBW.

In our early studies, we found that the ability to discriminate the minority class is very sensitive to the dropout method. This phenomenon needs further evidence and study. We will also attempt in future work to enrich the BBW framework and to combine it with other methods.

Acknowledgements This work was supported by the National Key Research and Development Program of China under grant 2017YFB1002504; by the National Natural Science Foundation of China (NSFC Grant No. 62073260).

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Buda M, Maki A, Mazurowski MA (2017) A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks 106:S0893608018302107–
2. Mathews L, Hari S (2018) Learning from imbalanced data
3. Zhang C, Bi J, Xu S, Ramentol E, Fan G, Qiao B, Fujita H (2019) Multi-imbalance: An open-source software for multi-class imbalance learning. Knowl.-Based Syst. 174(JUN.15):137–143
4. Sharma S, Bellinger C, Krawczyk B, Zaiane O, Japkowicz N (2018) Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In: 2018 IEEE International Conference on Data Mining (ICDM), pp 447–456. https://doi.org/10.1109/ICDM.2018.00060
5. Zheng W, Zhao H (2020) Cost-sensitive hierarchical classification for imbalance classes. Appl Intell 1–11
6. Zhou ZH, Liu XY (2006) Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans Knowl Data Eng 18(1):63–77
7. Sun Y, Kamel MS, Wong AKC, Wang Y (2007) Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn. 40(12):3358–3378
8. Zhou F, Yang S, Fujita H, Chen D, Wen C (2020) Deep learning fault diagnosis method based on global optimization gan for unbalanced data. Knowle-Based Sys 187(Jan.):104837.1–104837.19
9. Lin TY, Goyal P, Girshick R, He K, Dollar P (2017) Focal loss for dense object detection. IEEE Trans Pattern Anal Mach Intell PP(99):2999–3007
10. Li B, Liu Y, Wang X (2018) Gradient harmonized single-stage detector. arXiv preprint arXiv:181105181
11. Jia F, Lei Y, Lu N, Xing S (2018) Deep normalized convolutional neural network for imbalanced fault classification of machinery and its understanding via visualization. Mech. Syst. Signal Process. 110:349–367
12. Zhang L, Zhang C, Xiao H, Quan S, Liu L (2020) A class imbalance loss for imbalanced object recognition. IEEE J Sel Top Appl Earth Obs Remote Sens PP(99):1–1
13. Valova I, Harris C, Mai T, Gueorguieva N (2020) Optimization of convolutional neural networks for imbalanced set classification. Procedia Computer Science 176:660–669
14. Zhang C, Kjellstrom H, Mandt S (2017) Determinantal point processes for mini-batch diversification. arXiv preprint arXiv:170500607
15. Qi Q, Xu Y, Jin R, Yin W, Yang T (2020) Attentional biased stochastic gradient for imbalanced classification. arXiv preprint arXiv:201206951
16. Shoeb AH (2009) Application of machine learning to epileptic seizure onset detection and treatment. Massachusetts Institute of Technology
17. Andrzejak RG, Lehnertz K, Mormann F, Rieke C, David P, Elger CE (2001) Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. Phys Rev E Stat Nonlin Soft Matter Phys 64(6):061907
18. Xie Y, Wu Z, Han X, Wang H, Wu Y, Cui L, Feng J, Zhu Z, Chen Z (2020) Computer-aided system for the detection of multicategory pulmonary tuberculosis in radiographs. Journal of Healthcare Engineering
19. Huang G, Liu Z, Pleiss G, Van Der Maaten L, Weinberger K (2019) Convolutional networks with dense connectivity. IEEE Trans Pattern Anal Mach Intell
20. Sun Y, Wong AKC, Kamel MS (2011) Classification of imbalanced data: A review. Int J Pattern Recognit Artif Intell 23(04):687–719
21. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
22. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: Synthetic minority over-sampling technique. J. Artif. Intell. Res. 16(1):321–357
23. Han H, Wang WY, Mao BH (2005) Borderline-smote: A new over-sampling method in imbalanced data sets learning. In: International conference on advances in intelligent computing
24. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2009) Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In: Pacific-asia conference on advances in knowledge discovery & data mining
25. Sun J, Li H, Fujita H, Fu B, Ai W (2020) Class-imbalanced dynamic financial distress prediction based on adaboost-svm ensemble combined with smote and time weighting. Information Fusion 54:128–144
26. Xu-Ying L, Jianxin W, Zhi-Hua Z (2009) Exploratory undersampling for class-imbalance learning. IEEE Trans Sys Man & Cybern Part B 39(2):539–550

27. Lopez-Garcia P, Masegosa AD, Osaba E, Onieva E, Perallos A (2019) Ensemble classification for imbalanced data based on feature space partitioning and hybrid metaheuristics. Appl. Intell. 49(8):2807–2822
28. Hayashi T, Ambai K, Fujita H (2020) Applying cluster-based zero-shot classifier to data imbalance problems
29. Lee JS (2019) Auc4.5: Auc-based c4.5 decision tree algorithm for imbalanced data classification. IEEE Access 7:106034–106042
30. Taherkhani A, Cosma G, McGinnity T (2020) Adaboost-cnn: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. Neurocomputing 404:351–366. https://doi.org/10.1016/j.neucom.2020.03.064
31. Pérez-Hernández F, Tabik S, Lamas A, Olmos R, Herrera F (2020) Object detection binary classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance. Knowledge-Based Systems, p 105590
32. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift, pp 448–456

Jingzhao Hu received a B.S. degree in computer science and technology from Northwest University in 2016. Now he is a Ph.D. Candidate in Northwest University. His main research interests include deep learning, brain-computer interface, and artificial intelligence.

Hao Zhang, born in 1989, ph. D. candidate, lecturer, his research in deep learning, intelligent education, and natural language processing.

Yang Liu received the B.S. degree in communication engineering from Northwest university, Xi'an, China, in 2018, received the M.S. degree in communication and information system from Northwest university, Xi'an, China, in 2021. His research interests include machine learning and brain-computer interface.

Richard Sutcliffe received a Ph.D. from University of Essex in 1989. He is an Associate Professor at Northwest University China. His research interests are in the areas of Information Retrieval, Music Information Retrieval and Natural Language Processing. Recent projects have included persuasive conversational agents, public sector message classification, analysis of classical music texts, and personality and translation ability. He has reviewed for Artificial Intelligence Review, Computational Linguistics, Computers and the Humanities, Information Processing and Management, Information Retrieval Journal, Journal of Natural Language Engineering, Journal Traitement Automatique des Langues. Conferences he has reviewed for include ACL, CIKM, COLING, IJCNLP, LREC, NAACL-HLT, and SIGIR. He is the co-author of 108 articles and is co-editor of three books and ten conference proceeedings.

Jun Feng received a Ph.D. from City University of Hong Kong in 2006. She is a Professor in the School of Information Science and Technology at Northwest University. Her research areas include pattern recognition and machine learning, especially in the fields of brain-computer interface, medical imaging analysis and intelligent education. Recent projects have included intelligent education based on AI and Brain-Human Interaction, and medical image analysis with deep learning. She has reviewed for many journals, including TSP, JIVP, MTAP, JDIM, CJC, JCAD, OPE, and INFPHY. Conferences she has reviewed for include IEEE-VR,MICCAI, SIGCSE, IWCSE, and CompEd. She is a member of IEEE and ACM, and is co-author of 132 articles and co-editor of three books.