Hindawi
Complexity
Volume 2021, Article ID 8667868, 10 pages
https://doi.org/10.1155/2021/8667868

Research Article

Classification of Long-Tailed Data Based on Bilateral-Branch Generative Network with Time-Supervised Strategy

Yalin Huang, Yan-Hui Zhu, Zeng Zhigao, Yangkang Ou, and Lingwei Kong

School of Computer Science, Hunan University of Technology, Zhuzhou, Hunan 412008, China

Correspondence should be addressed to Yan-Hui Zhu; swayhzhu@163.com

Received 4 September 2021; Accepted 20 October 2021; Published 2 November 2021

Academic Editor: Hou-Sheng Su

In the face of the long-tailed data distribution that widely exists in real-world datasets, this paper proposes a bilateral-branch generative network model. The data of the second branch is constructed by resampling the generative network training method to improve the data quality. A bilateral-branch network model is used to curb the risk of gradient explosion and to avoid over-fitting and under-fitting with the combined effect of different data branches. Meanwhile, Time-supervised strategy is introduced to improve the model's operational efficiency and ability to cope with extreme conditions by supervising and collaboratively controlling of the bilateral-branch generative network with time-invariant parameters. Time supervised strategy could ensure the accuracy of the model while reducing the number of iterations. Experimental results on two publicly available datasets, CIFAR10 and CIFAR100, show that the proposed method effectively improves the performance of long-tail data classification.

1. Introduction

With the rapid development of convolution neural network algorithms in recent years, there has been a very impressive improvement in the performance of image classification. Undoubtedly, the success is inextricably linked to the available high-quality large-scale datasets, such as ImageNet ILSVRC 2012 [1], MS COCO [2] and Places Database [3]. However, compared to these high-quality datasets, real-world datasets are always biased and it is difficult to ensure a uniform distribution of data, and more often than not, certain classes of data are very abundant while certain remaining classes are very scarce, which leads to a long-tail distribution of data [4, 5] and affects the performance of image classification. From the reviewed related materials, class re-balancing strategies are currently often used when faced with uneven data distribution. Class re-balancing strategies are further divided into two categories, namely re-sampling strategies [6–13] and re-weighting strategies [14–17]. The re-sampling strategy means that the source data is resampled according to the desired frequency distribution to obtain a new datasets by copying the minority data [6, 8, 12, 18] and reduce the majority data [7, 12, 13]. This strategy is able to reduce error rate caused by unbalanced data during training. Although resampling can show better results, this strategy still has a negative effect on the model. For example, the SMOTE algorithm [11] merely repeats and abandons the original data in the process of resampling. Although it changes the data distribution, it cannot bring more classified information to the deep learning model. Therefore classical resampling tends to make the tail data more prone to over-fitting conditions, while the head data is also tend to under-fitting conditions.

And this can be effectively avoided by a re-weighting method that adds a regularization term to the loss. Where the loss of the regularization parameter can often be expressed in

$$\text{Loss} = \text{loss}_1 + \lambda * \text{loss}_2 \quad (0 \le \lambda \le 1). \tag{1}$$

The over-fitting and under-fitting of the model is suppressed by introducing a new loss function as a constraint. However, the regularization method also has its drawbacks. Due to the introduction of the regularization parameter as a restriction in the loss, it sometimes makes the model parameters fail to converge, and in extreme cases, it even results in gradient explosion. Therefore some scholars proposed many

ways to prevent the over-fiting and under-fiting with data-dependent regularizer [19–22]. For example, the algorithms proposed by Zhou et al. [23] based on bilateral-branch network model has good accuracy in the classification problem of long-tailed data. The algorithm obtained the first place in iNaturalist2019 with an error rate of 30.38% on the iNaturalist2018 public datasets. The model of Boyan Zhou re-weights the loss function by a data-driven approach. This data-driven approach not only effectively curbs the risk of gradient explosion, the model will converge more easily under the combined effect of different data branches while avoiding the over-fitting phenomenon on tail data [4]. However, this method requires very strict data quality. Since the regularization term itself is data-driven approach, if the data quality of the second branch is poor, it will lead to loss_2 adverse effect on the direction of the overall model learning. Therefore how to construct high quality second branch with different distributions becomes a new problem.

Many works focus on minority samples, the re-balance the data by augmentation of minority samples [11, 24, 25]. while some other works provide a Major-to-minor translation to re-balance the data distribution [26–28]. the M2m algorithm proposed by Kim et al. [29], which is different from the above methods and gives a solution from the perspective of resampling. M2m algorithm is a method of resampling data through data generation by generation against network [26, 30, 31]. This method solves the problem of unbalanced data distribution by constructing fewer classes of data through multiple classes of data, while bringing more classification information to the data and improving the data quality. This method can effectively increase the number of minority class samples while greatly reducing the over-fitting of the minority class data. However, this method often requires the use of an already pre-trained network, and generating data during training still requires a large number of iterations, which will reduce the efficiency of model learning.

The main contributions of this paper are as follows.

(1) Incorporating the respective advantages of the Bilateral-branch model and the generative network model, this paper proposes a bilateral-branch generative network model. The data of the second branch is constructed by resampling and generating by the generative network training method to improve the data quality. The bilateral-branch network model is used to curb the risk of gradient explosion, and the model is made to avoid over-fitting and under-fitting phenomena under the combined effect of different data branches to improve the effect of long-tail data classification.

(2) Since the generative network model adds a large number of iterations in the process of generating data, which affects the efficiency of the model, this paper introduces a time-supervised strategy, which supervises and limits the number of iterations of the generative network through time-variant parameters, improves the operational efficiency of the model and its ability to cope with extreme conditions, and ensures the accuracy of the model while reducing the number of iterations.

(3) The accuracy of the algorithm was tested under two publicly available datasets, CIFAR10 and CIFAR100 [32], for different distributions, and the method used in this paper, was higher than both Boyan Zhou' algorithm and M2m algorithm.

2. Construction of a Bilateral-Branch Generative Network Model

The overall framework of the bilateral-branch generative network model proposed in this paper is shown in Figure 1, and the model is abbreviated as BBGN (Bilateral-Branch Generative Network).

As shown in the figure the BBGN model first constructs the inverse distribution data branches by data resampling. We refer to the source data branch as the first branch data, and the inverse distribution data constructed by the resampling method as the second branch. After the data of the second branch is generated by the re-sampling method, the first branch data is used as the data source to generate new high-quality data for replacing the data of the second branch by the generating network (GN) module to perform data augmentation on the data of the second branch. In the GN module and later in the network model we introduce respectively the time-supervised parameters $(1 - \alpha)$ with α for coordinate and control these two functional modules.

After the process of data augmentation is completed, the original two data branches are feature extracted using a pyramid-shaped multiple layer feature perceptron to form separate sets of features. Through the time-supervised parameterα of control, the two feature sets are fused to form a one-class feature set. The fused features are averaged pooled (GAP layer in the figure) and pushed into the fully connected classifier for classification. The model loss is calculated and the model parameter weights are updated backwards based on the classification results and the loss function.

Through the coordination and control of the time-supervision strategy, in the early stage of BBGN model learning, the BBN network is influenced by the time-supervision parameter, which makes the data of the second branch play almost no role in the learning of the model, while the GN network is also subject to the synergy of this parameter and does almost no data enhancement to the data of the second branch. At this point the data of the second branch plays a weak role in the learning of the model only by resampling the data, and the learning process of the model is dominated by the source data of the first branch. Over time, under the influence of the time-supervised strategy, the influence of the BBN model's bilateral-branch data on model learning gradually changes. The role of the influence of the data of the second branch gradually increases. At the same time the GN network is gradually activated by the coordination of the temporally supervised parameters, and the resampled data of the second branch starts to play its role in the model learning process after being enhanced by the GN

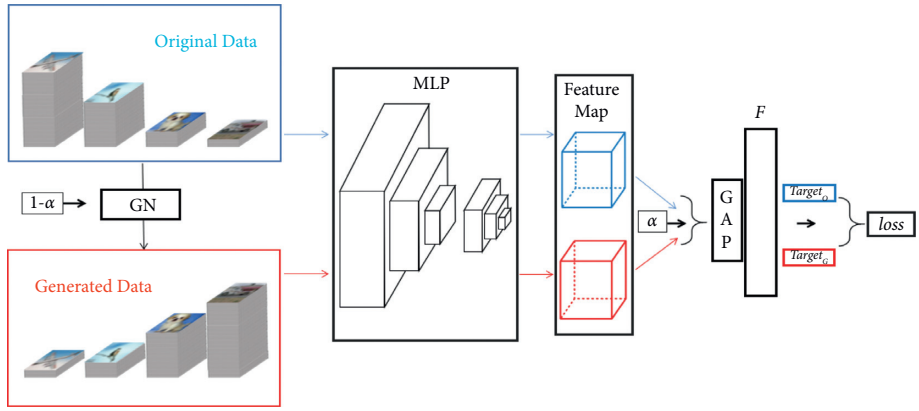

Figure 1: Overall framework of the bilateral-branch generative network model.

module. The introduction of the time-supervised strategy enables our algorithm to maintain accuracy while reducing the number of iterations and greatly improving the computational efficiency.

2.1. Bilateral-Branch Network (BBN) Model. The BBN model is data-driven and uses a loss function weighting method to avoid over- and under-fitting of the data.

The BBN model starts from a regularized weighting perspective, but unlike the usual method of constructing regularized expressions, this model achieves regularized weighting by using branches of data from different distributions to learn and fuse the losses generated by multiple branches. The pseudocode of the BBN model algorithm is as follows.

At the row 14 of Algorithm 1, we get the $\text{Target}_{\text{pred}}$ of model by mixing the feature_1 and the feature_2. This combiner function canbe expressed by the following equation (6). And then, at the row 18 of Algorithm 1 the BBN model Combiner2 function can be expressed by the following equation (2).

$$\text{Loss} = \alpha * \text{loss}_1 + (1 - \alpha) * \text{loss}_2 \quad (0 \le \alpha \le 1), \tag{2}$$

where loss_1 is the error generated in the learning process for the first branch data in the BBN model, and loss_2 is the error generated by the data of the second branch in the learning process. At the row 16 and 17 of Algorithm 1, the $\mathscr{L}$ function can be expressed by the following equation (8), and Loss is the total error generated by the BBN model in the learning process. Loss affects the learning process of the model through the synthesis of the error components of the two data.

2.2. Generating Network Models. The pseudocode of our proposed algorithm for generating models in BBGN is shown below.

At the first row of the Algorithm 2, the Bernoulli distribution function used to select the class to be generated is shown in .

$$P_G = p^{N_{\text{Target}_G}/N_0} * (1 - p)^{1-\left(N_{\text{Target}_G}/N0\right)}, \tag{3}$$

where Target_G denotes the class to be generated, and N_{Target_G} denotes the total number of samples of the class to be generated, and N_0 denotes the total sample size of the class corresponding to the class with the highest frequency, and P_G denotes the probability that such class is selected as the class to be generated.

At the row 3 fo Algorithm 2.

At the row 5 of Algorithm 2, the generation source class Target_O are selected using the distribution function as shown in .

$$P_O = 1 - \beta^{N_{\text{Target}_O} - N_{\text{Target}_G}}, \tag{4}$$

where $\beta \in (0, 1)$ is a fixed parameter, and N_{Target_O} denotes the total number of samples of the generated source class, and N_{Target_G} denotes the total number of samples of the class to be generated, and P_O denotes the probability that such class is selected as the generating source class.

And then we need to select the generation source based on the class to be generated. Since this data augmentation algorithm constructs minority class samples by extracting class-independent features from the majority class and class-related features from the minority class. Therefore the method has certain requirements on the data quality of the generation source data, and the class-irrelevant feature classes contained in the data should be as rich as possible, so as to improve the generalization ability of the model for minority class learning.

In this Algorithm 2, the class to be generated Target_G and the generated source class Target_O are selected by resampling the class label with the probability distribution. This approach can meet our requirements on the quality of the generated source data. At the same time, the possibility of generating fewer classes of data with fewer classes of data is preserved, and the choice of generating source class data is expanded.

The framework diagram of the generative network used for Algorithm 2 is shown in Figure 2.

For the selected image I, input the to-be--trained model F to obtain the I's classification result Label_F, and compute

Inputs: training datasets I_1, $\lambda, \gamma, \eta, L > 0, \beta \in [0, 1), \alpha$
Output: BBN network model F (input; weight)

$I_2 \leftarrow I_1$
for $k = 2$ to K do
  $\Delta \leftarrow (N_1 - N_k)$
  for $i = 1$ to Δ do
    $x \leftarrow$ A random sample of class k in I_2
    $I_2 = I_2 \cup x$
  end for
end for
$I_2 =$ Algorithm 2 $(I1, I2, \lambda, \gamma, \eta, L > 0, \beta \in [0, 1), \alpha)$
for $i = 0$ to N_{I_1} do
  $j \leftarrow$ A random sample of I_2
  $\text{feature}_1 = \digamma(I_{1i}, W^T)$
  $\text{feature}_2 = \digamma(I_{2i}, W^T)$
  $\text{Target}_{\text{pred}} = \text{Combiner1}(\text{feature}_1, \text{feature}_2, \alpha)$
  $\text{loss}_1 = \mathscr{L}(\text{Target}_{1i}, \text{Target}_{\text{pred}})$
  $\text{loss}_1 = \mathscr{L}(\text{Target}_{2j}, \text{Target}_{\text{pred}})$
  $\text{Loss} = \text{Combiner2}(\text{loss}_1, \text{loss}_2, \alpha)$
  $W \leftarrow W - \text{Loss}$
end for
$F \leftarrow W$

Algorithm 1: BBN algorithm.

Input: An unbalanced data branch I_1. A resampled data branch I_2. A network f. A pre-trained generative network G. $\lambda, \gamma, \eta, L > 0, \beta \in [0, 1)$
Output: A augmented balanced data branch O

$\text{Target}_G \leftarrow$ A random sample of class K in I_2 with P_G
for $k =$ in Target_G do
  $\Delta \leftarrow (N_1 - N_k) * T(t, 1 - \alpha)$
  for $i = 1$ to Δ do
    $\text{Target}_O \leftarrow$ A random sample of class k in I_1 with P_O
    $I \leftarrow$ A random sample of class Target_O in I_1
    Add some noise σ to the I
    for $t = 1$ to L do
    $\text{Loss}_G = \mathscr{L}(G; I; k)$
    $\text{Loss}_O = f_{\text{Target}_O}(I)$
    $\text{loss}_{GN} \leftarrow \nabla_R[\text{Loss}_G + \lambda * \text{Loss}_O]$
    $I \leftarrow I - \eta * (\text{loss}_{GN}/\text{loss}_{GN2})$
    end for
    $P \sim \text{Bernoulli}(\beta^{(N_{k_0} - N_k)^+})$
    If $\mathscr{L}(g; I; k) > \gamma$ or $P = 1$ **then**
    $I_2^* \leftarrow$ A random sample of class k in I_2
    end if
    $I_2^* \leftarrow I$
  end for
end for

Algorithm 2: Data generation algorithm.

this result with Label_F and the generated source class label Target_O. The distance between this result and the generated source class label Target_O is used as the regularization term loss_O in loss function. Classify I using the feature information provided by the pre-trained model G to obtain the result Label_G. Use the loss function to calculate the loss_g by Label_G and the target to be generated Target_G. calculate the total loss loss_{GN} using equation (5), and backward update the pixel of selected image I based on the total error.

$$\text{loss}_{GN} = \text{Loss}_G + \lambda * \text{Loss}_O. \tag{5}$$

In the updating process, the goal of loss_g is to cause the selected image I to be classified as the class Target_G with the pre-trained network G. Since the pre-trained network G

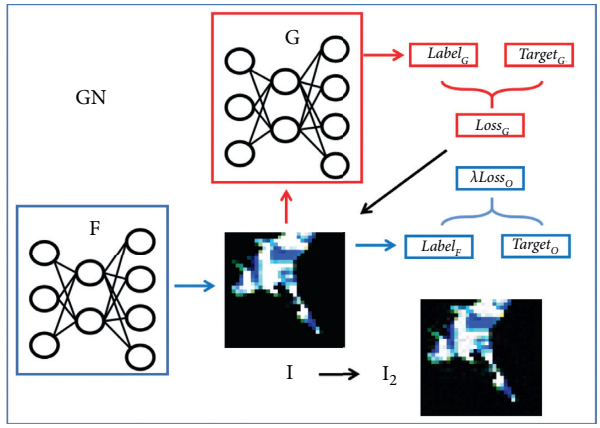

Figure 2: Framework diagram of the generation network.

contains the class-related feature information of each class, the $loss_g$ guided by the class-related feature information of the network G, could gradually eliminate the class-related features of the class $Target_O$ in the image I and add the class-related features of the class $Target_G$. The result image generated by this generative network algorithm consists of the class-independent features of the $classTarget_O$ and the class-irrelevant features of the class $Target_G$. By this way we could generated the new image of the class $Target_G$.

In the unbalanced datasets, the majority class has a very rich set of class-irrelevant features in addition to such class-relevant features. While the minority class has only class-related features with a small number of class-irrelevant features. Therefore the class irrelevant features carried in the majority classes are combined with the class relevant features carried in the minority classes in the pre-trained network G through the generative network to generate a large number of images of the minority classes. New classification information is introduced while balancing the data distribution.

Also, in order to prevent the situation that the gradient disappears during the training process and the target image cannot be generated, the regularization term can be set to eliminate this situation. The above $loss_O$ represents the prediction result of image I in the network F to be trained $Label_F$ with the image to be generated in the source $classTarget_O$ of the difference. Before the classification is performed, the $Label_F$ is in the form of a 0-1 vector, and the respective values on the vector indicate the probability of the image being of this class. By setting this difference as a regularization term, the gradient disappearance can be effectively prevented, allowing the input image I to be generated more easily as an image of other classes.

3. Time Supervision Strategy

The variation of the time-supervised parameters with training time is shown in Figure 3.

The temporal supervision strategy proposed in this paper acts on both the BBN model and the data generation module. The time-supervised strategy coordinates and controls the overall model learning process by setting the time-supervised parameters α.

3.1. Time-Supervision Strategies for BBN Models. In our BBGN model, the degree of influence of different branches on the learning process of the model can be adjusted by introducing time-supervised parameters in the computation process of the fusion and loss of the two branches. The first branch data is the original real data, while the data of the second branch enhanced by the generative network is the non-real data generated by data features. Therefore, in the early stage of model training, the real data should be the main focus, and the real features in the data should be learned as much as possible. And when the minority class gradually starts to over-fit, then gradually start to increase the degree of influence of the non-real data generated by the features on the model learning to prevent the model from over-fitting on the minority class, and at the same time enhance the generalization ability of the model on the minority class to improve the accuracy of the model.

The fused feature is obtained during the fusion of model features by .

$$\text{Feature} = \alpha * (\text{feature}_1) \cup (1 - \alpha) * (\text{feature}_2). \tag{6}$$

It is important to note that when we use the results obtained from fused feature with classifier when calculating the loss function, it needs to be compared the result label to the two objectives $Target_O$ with the $Target_G$ together to calculate the loss. Since the feature fusion is performed with the time-supervised parameter α, the loss should also be calculated with the time-supervised param-eter α to determine the final loss. The loss calculation in this paper is shown in equation (7).

$$\text{Loss} = \alpha * \mathscr{L}(\text{Label}, \text{Target}_O) + (1 - \alpha) * \mathscr{L}(\text{Label}, \text{Target}_G), \tag{7}$$

where the loss function $\mathscr{L}(L, T)$ is calculated as shown in .

$$\mathscr{L}[L, T] = -\sum_i T_i * \ln(L_i), \tag{8}$$

L_i denotes the ith vector in Label, representing the probability that the input is of class i. T_i denotes the ith vector in Target, where only one vector is 1 indicating that the input is of class i and the rest of the vectors are 0.

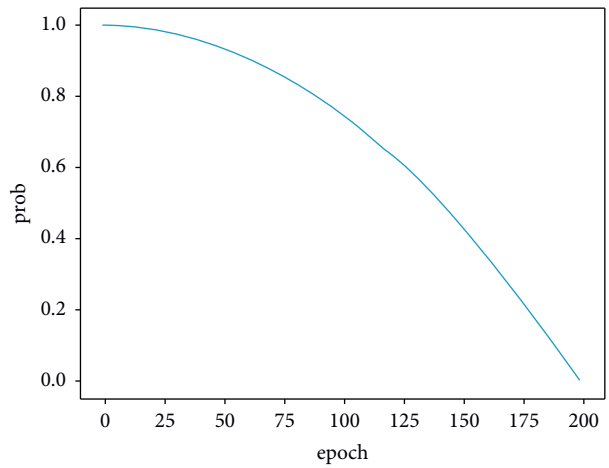

Figure 3: Curve of the change in the parameters of time supervision.

3.2. Time-Supervision Strategies in Generative Network Models. The generative network model improves the performance of the model by improving the quality of the data of the second branch. However in the early stages, the data of the second branch has little impact on the learning process of model learning, and the large amount of generated data does not contribute to the model learning process. This would generate a waste of resources. So we use equation (9) and introduce a time-supervised strategy to limit the size of generated data in the early stage.

$$L = N_{\text{Target}_G} * (1 - \alpha), \tag{9}$$

where L is the number of samples to be generated, and N_{Target_G} is the total number of classes to be generated, and α is the time supervision parameter.

By the introduction of the time-supervised strategy, the amount of generation of the generative network is limited in the early stage of model training. And with the increase of the number of iterations, the size of generated data L of the generative network is gradually activated by the time-supervised strategy, when the main component of the data of the second branch is changed from resampled data to generated data, and the data quality is gradually improved. In this way, the number of iterations can be greatly reduced and the learning efficiency can be improved while ensuring the accuracy of the model.

4. Experimentation and Analysis

4.1. Experimental Data Set. In this paper, we use online public datasets CIFAR10 and CIFAR100 [32], both of which contain 60,000 RGB color images of $32 * 32$ size. There are 50,000 images in these images for training and 10,000 images for testing. CIFAR10 and CIFAR100 have 10 classes and 100 classes of data respectively. The source datasets is a uniform datasets. And in this paper, we resample the long-tail datasets with class imbalance of CIFAR10 and CIFAR100 by setting the imbalance ratio parameter Ratio during the experiment. Among Ratio = $N_{\max}/N_{\min}$, in this paper, Ratio contains three values of 10,50,100 respectively.

4.2. Introduction of Experimentation. Our experimentation are running in the ubuntu20.04 operation system. Traning the model need about 16G RAM. The machine language of the experimentation is python. In the experimentation, at first we will train our model with pretrained model and unbalanced trainning datasets. Then we test the result of our model in the testing dataset. Finally we visualize our result and analyze it.

4.3. Analysis of Experimental Results. In this paper, the accuracy of the optimal solution on the test set is used to compare the classification ability of different methods by comparing the experimental results. Also, we track the accuracy of the classification results of different models on the test set during the training process and plot it as a line graph, which can visualize the difference of different models during the training process.

4.3.1. Results of the CIFAR10 Experiment. From Table 1 and Figure 4, we can find that the accuracy of BBN algorithm and our algorithm decreases when the number of training is between 50 and 75, while M2m algorithm can quickly improve to a higher level and stabilize at that level in the early stage. However, as the number of training gradually increases, the improvement of M2m accuracy gradually becomes slower, while the accuracy of BBN algorithm and our algorithm improves rapidly. Considering that both the BBN algorithm and our algorithm contain a bilateral-branch structure, the decrease in the accuracy of the algorithm is related to the gradual increase in the influence of the second branch. At the beginning, the accuracy drops significantly due to the small influence of the data in the second branch of the model, but as the number of training sessions increases, the accuracy of the BBN algorithm and our algorithm always steadily exceeds that of the M2m algorithm with the effect of the data in the second branch. This indicates that the pure data generation model can achieve a high accuracy rate within a smaller number of training sessions, but the final accuracy ceiling of the model is not high. Compared to the BBN algorithm, the lead is weaker on the CIFAR10 datasets

Table 1: The accuracy of resnet-32 on long-tailed cifar10.

Methods	CIFAR100		
	$N1/N100 = 10$	$N1/N100 = 50$	$N1/N100 = 100$
M2M	57.55	45.63	42.16
BBN	59.47	45.90	42.47
Methods (ours)	60.14	47.06	42.64

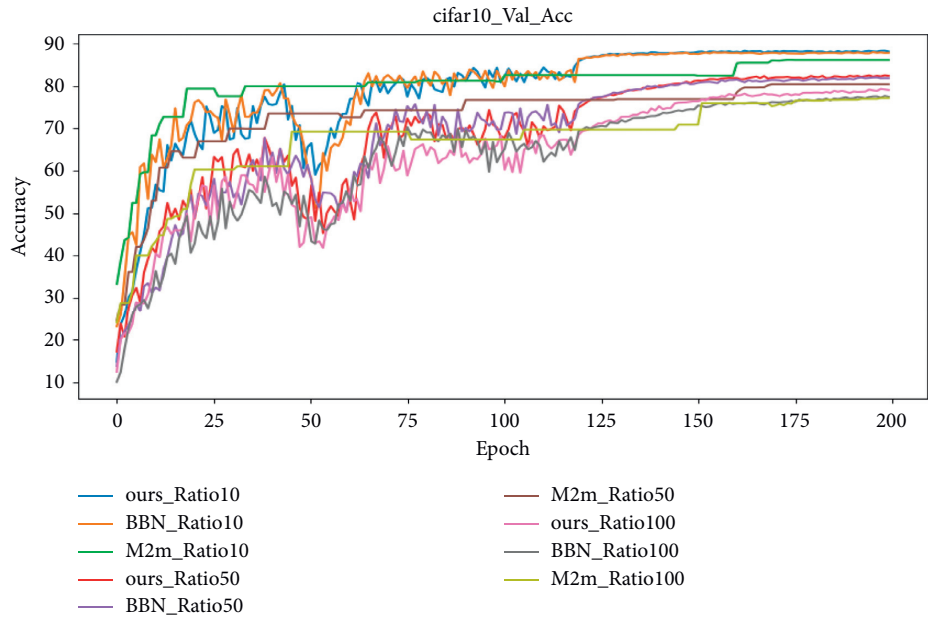

Figure 4: Accuracy variation curve of cifar10 long-tail datasets.

when the imbalance ratio is 10 versus 50, although our method outperforms the BBN algorithm. But when faced with an extreme imbalance situation, i.e., when the imbalance ratio is 100, the advantage of our algorithm is significantly improved. This shows that when the realistic environment is not complex enough, the data augmentation by resampling alone can achieve better results, but in the extreme environment, the data quality of the second branch will be very poor, and thus better results can be obtained by introducing data generation to improve the data quality.

4.3.2. Results of the CIFAR100 Experiment. The results of the CIFAR100 experiments are shown in Table 2 and Figure 5. It can be found that the curves are roughly the same as cifar10, the M2m accuracy stabilizes to a high value very quickly, while the BBN algorithm with our algorithm takes the lead in decreasing and then improving as the number of training increases. The CIFAR100 datasets is 10 times more classified and has less datasets in each class than CIFAR10, so obtaining a higher accuracy is very difficult. In this extreme condition, our algorithm can take advantage of the data enhancement and lead the BBN and M2m algorithms. However, surprisingly, in the most extreme case of Ratio = 100 for CIFAR100, the accuracy rates of all three are comparable and lower, and the accuracy curves are almost completely indistinguishable. This is perhaps due to the fact that in the too extreme case, the ResNet32 network leads to this result due to the limitations of its network structure.

To test this conjecture, we did a set of comparison experiments on the CIFAR100 datasets with Ratio = 100, using the ResNet50 network.

4.3.3. Results of the ResNet50 Experiment. The experimental results are shown in Figure 6. From the accuracy curves, we can find that when the model has more parameters and the network structure is more complex, the accuracy results of our method and the BBN method produce very obvious differences. Benefiting from the role of the data generation network, the BBGN network model constructed by the generation method with the time-supervision strategy can also significantly improve the accuracy of the model in the case of very extreme data distribution.

Table 2: The accuracy of resnet-32 on long-tailed cifar100.

Methods	CIFAR100		
	$N1/N100 = 10$	$N1/N100 = 50$	$N1/N100 = 100$
M2M	57.55	45.63	42.16
BBN	59.47	45.90	42.47
Methods (ours)	60.14	47.06	42.64

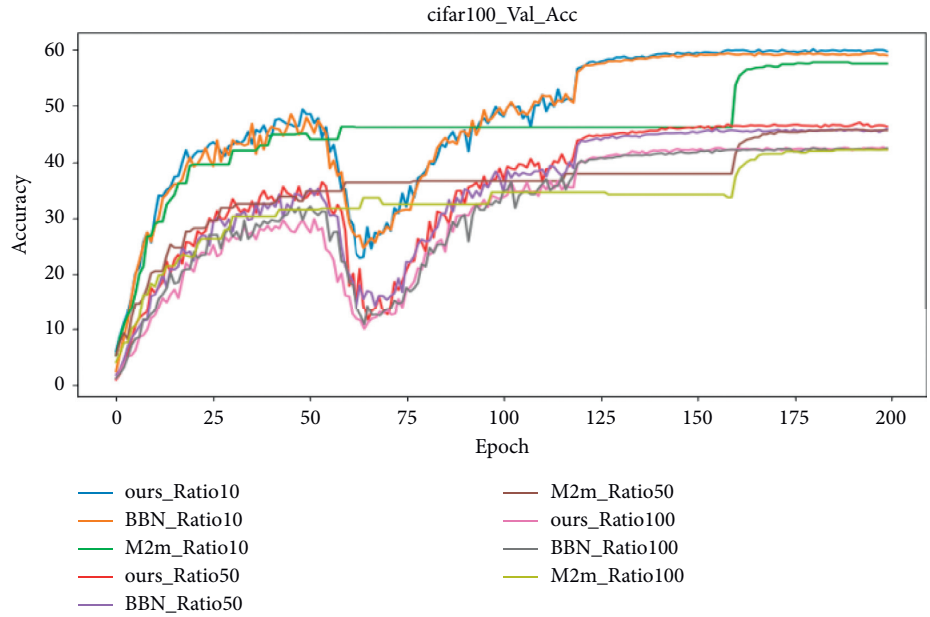

Figure 5: Accuracy variation curve of cifar100 long-tail datasets.

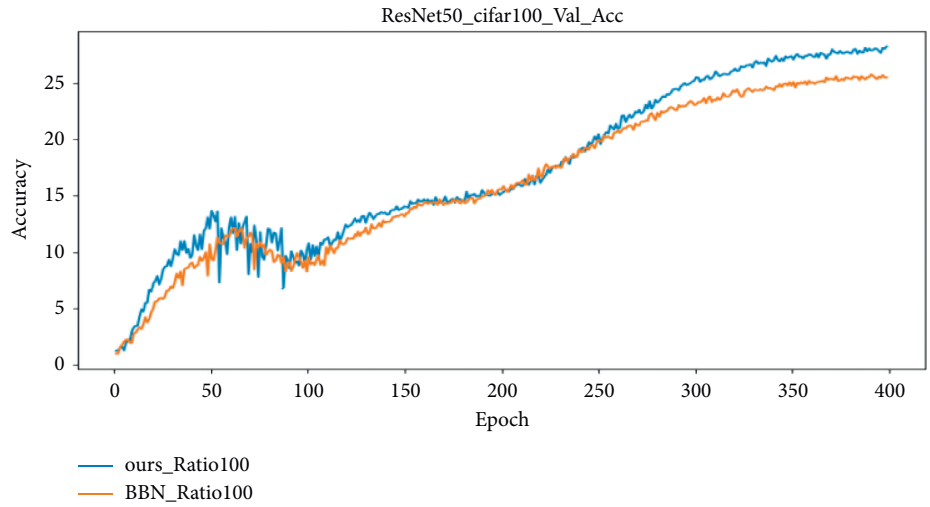

Figure 6: Accuracy curve of resnet50 network structure on a long-tailed datasets with cifar100 imbalance ratio of 100.

5. Summary and Outlook

In this paper, two classical solutions are introduced to address the characteristics of unbalanced data distribution of long-tailed data sets: resampling methods and weighting methods. Two types of algorithms for processing long-tailed data, BBN and M2m. we studied and analyzed these two methods and proposed bilateral-branch generative network model based on them. This model improves the accuracy of classifying long-tailed data by data-driven re-weighting

methods successfully and data enhancement methods based on generative networks. Also in this paper, a time-supervised strategy is introduced in this model to coordinate and control GN and BBN modules to reduce the number of algorithm iterations while maintaining a high accuracy rate. Compared with BBN and M2m algorithm, this algorithm can obtain higher accuracy rate stably.

The following shortcomings still exist in this paper: the time-supervised strategy proposed in this paper, although it reduces a large number of iterations and improves the operation efficiency, making the model run significantly more efficiently than the M2m algorithm, there are still differences compared to the BBN algorithm.

Data Availability

We used CIFAR10 and CIFAR100 public datasets to support the findings of the research. All data used in the study can be download at http://www.cs.toronto.edu/~kriz/cifar.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of the study.

Acknowledgments

This paper was partially supported by National Key Research and Development Program of China Grant (2018AAA0100400), the Natural Science Foundation of Hunan Province (2020JJ6089) and the Key Project of the Department of Education in Hunan Province (19A133).

References

[1] O. Russakovsky, J. Deng, H. Su et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.

[3] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: a 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2018.

[4] G. Van Horn and P. Perona, "The devil is in the tails: fine-grained classification in the wild," 2017, https://arxiv.org/abs/1709.01450.

[5] M. G. Kendall, A. Stuart, J. Keith Ord, S. F. Arnold, O. 'H. Anthony, and J. Forster, *Kendall's Advanced Theory of Statistics*, Vol. 1, Wiley, New York, NY, USA, 1987.

[6] L. Shen, Z. Lin, and Q. Huang, "Relay backpropagation for effective learning of deep convolutional neural networks," in *Proceedings of the Computer Vision - ECCV 2016*, Amsterdam, The Netherlands, October 2016.

[7] E. A. Haibo He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[8] J. Byrd and Z. Lipton, "What is the effect of importance weighting in deep learning?" in *Proceedings of the 36th International Conference on Machine Learning, PMLR*, vol. 97, pp. 872–881, Long Beach, CA, USA, 2019.

[9] C. Drummond and R. C. Holte, "C4.5, class imbalance, and cost sensitivity: why under-sampling beats oversampling," *Workshop on Learning From Imbalanced Datasets II*, vol. 11, pp. 1–8, 2003.

[10] A. More, "Survey of resampling techniques for improving classification performance in unbalanced datasets," 2016, https://arxiv.org/abs/1608.06048.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Philip, and S. M. O. T. E. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[12] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[13] N. Japkowicz and S. Stephen, "The class imbalance problem: a systematic study," *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[14] C. Huang, Y. Li, C. C. Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5375–5384, CVPR, Las Vegas, NV, USA, June 2016.

[15] Y.-X. Wang, D. Ramanan, and M. Hebert, "Learning to model the tail," in *Proceedings of the 2016 NeurIPS*, pp. 7029–7039, Barcelona, Spain, December 2017.

[16] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9268–9277, CVPR, Long Beach, CA, USA, June 2019.

[17] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," 2018, https://arxiv.org/abs/1803.09050.

[18] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie, "Large scale fine-grained categorization and domain-specific transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018.

[19] S. Khan, M. Hayat, S. W. Zamir, J. Shen, and L. Shao, "Striking the right balance with uncertainty," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019.

[20] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with labeldistribution-aware margin loss," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2019.

[21] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, "Range loss for deep face recognition with long tailed training data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Venice, Italy, October 2017.

[22] D. Qi, S. Gong, and X. Zhu, "Imbalanced deep learning by minority class incremental rectification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 6, pp. 1367–1381, 2018.

[23] B. Zhou, Q. Cui, X. S. Wei, and Z. M. Chen, "BBN: bilateral-branch network with cumulative learning for long-tailed visual recognition," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, December 2019.

[24] S. S. Mullick, S. Datta, and S. Das, "Generative adversarial minority oversampling," in *Proceedings of the IEEE*

International Conference on Computer Vision (ICCV), Seoul, South Korea, October 2019.

[25] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and X. Yu Stella, "Large-scale long-tailed recognition in an open world," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019.

[26] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycleconsistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, vol. 2, pp. 2223–2232, Venice, Italy, 2017.

[27] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 8789–8797, Salt Lake City, UT, USA, June 2018.

[28] S. Mo, M. Cho, and J. Shin, "InstaGAN: instance-aware image-to-image translation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LO, USA, 2019.

[29] J. Kim, J. Jeong, and J. Shin, "M2m: imbalanced classification via major-to-minor translation," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[30] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Computing Surveys*, vol. 49, no. 2, p. 31, 2016.

[31] S. Mo, M. Cho, and J. Shin, "InstaGAN: instance-aware image-to-image translation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[32] A. Krizhevsky, "Learning multiple layers of features from tiny images," Technical report, Department of Computer Science, University of Toronto, Toronto, Canada, 2009.