Diffusion-Noise-Based Augmentation for Long-Tailed Remote Sensing Image Classification

Qianqian Wang, Haibo Ye, Dong Liang, and Sheng-Jun Huang

***Abstract*— Remote sensing image classification refers to the task that using algorithms to categorize satellite or aerial imagery into different land cover types. In the real world, long-tailed data distribution is commonly present in remote sensing image classification tasks, causing models to excessively favor sufficient head classes during training and degrading prediction accuracy for scarce tail classes. Although existing methods such as resampling and reweighting can alleviate the issue of data imbalance to a certain extent, they struggle to sufficiently enhance the diversity of tail class samples. In recent years, some works have begun to use diffusion models to generate diverse samples to balance the class distribution. However, these approaches often overlook the distribution inconsistency between generated and real images, which will hinder the improvement of model performance. To tackle these challenges, this article proposes a novel diffusion-noise-based augmentation (DONA) method with a two-stage training process. Before training, our specially designed conditional prompts are used together with the original training set to guide the diffusion model in image generation. Furthermore, we propose two strategies to effectively leverage the generated images, which are applied, respectively, at the end of the first training stage and during the second training stage. First, we design DiffCam-Mix to fuse the background of the generated data with the foreground of the original data, preserving the essential information of the original real images while incorporating the diversity of the generated ones. Second, we use cosine similarity to minimize the differences between the mixed data and their corresponding original data, further calibrating the distribution of different samples. Extensive experiments on three public datasets—SIRI-WHU-LT, PatternNet-LT, and RSI-CB256-LT—demonstrate the effectiveness of the proposed method.**

***Index Terms*— Diffusion model, long-tailed distribution, remote sensing image classification, visual recognition.**

I. Introduction

REMOTE sensing image classification is mainly used to identify and classify land cover types in images obtained from satellites or aerial photography. It has significant implications for fields, such as environmental monitoring [1], [2], [3], urban planning [4], [5], [6], [7], [8], agriculture [9], and other areas [10], [11], [12]. In recent years, deep convolutional neural networks have been widely applied to remote sensing image classification tasks [13], [14], [15], [16], [17]. These networks are capable of automatically learning and extracting complex features of remote sensing images, thereby greatly improving the performance of image classification tasks.

Although traditional convolutional neural networks have shown great superiority in remote sensing image classification, the distribution of remote sensing images in the real world often shows imbalance, i.e., the long-tailed distribution of data: a few classes have a large number of samples, namely, the head class; while the vast majority of classes have a small number of samples, namely, the tail class. Fig. 1 illustrates this data distribution. In this case, traditional deep convolutional neural networks may learn biased representations toward the head class, resulting in poor recognition of the tail class. Therefore, how to improve the performance of long-tailed remote sensing image classification tasks has gradually attracted widespread attention. There are three main solutions to address long-tailed problems: class rebalancing, information augmentation, and module improvement [18]. Class rebalancing mainly includes methods, such as resampling [19], [20] and reweighting [21], [22], which aims to balance the feature space and class boundaries between the head class and the tail class. Although these methods can enhance the performance of the model to some extent, they do not fundamentally address the issue of data scarcity, particularly when the data in the tail class are extremely limited. Information augmentation mainly includes methods, such as transfer learning [23], [24] and data augmentation [25], [26], which seeks to improve the model performance by introducing additional information. These methods introduce limited additional information and thus have limited contribution to enhancing the model. Module improvement mainly includes decoupled training [27], [28], ensemble learning [29], [30], and other methods [31], [32], [33], which strives to improve the performance of feature extraction and classifiers. Such methods have certain advantages in improving feature extraction and classifier performance, but at the same time, they also increase the complexity of the model.

Recently, a few works dedicated to generating augmented samples by traditional data augmentation [34], [35], [36]. However, these methods experience obstacles in generating

Received 14 November 2024; revised 21 February 2025 and 16 April 2025; accepted 22 April 2025. Date of publication 12 May 2025; date of current version 12 June 2025. This work was supported in part by the National Key Research and Development Program under Grant 2024YFB3311401 and in part by NSFC under Grant 62272229 and Grant 62222605. *(Corresponding author: Haibo Ye.)*

The authors are with the MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, Collaborative Innovation Center of Novel Software Technology and Industrialization, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: wangqianqian0026@nuaa.edu.cn; yhb@nuaa.edu.cn; liangdong@nuaa.edu.cn; huangsj@nuaa.edu.cn).

Digital Object Identifier 10.1109/TGRS.2025.3568812

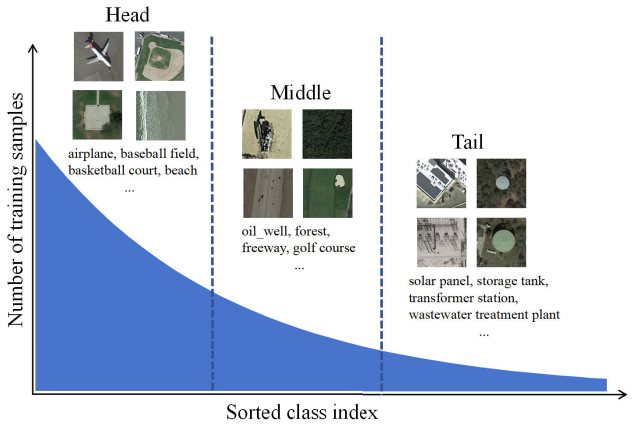


Fig. 1. Long-tailed distribution of data in the real world. The data can be divided into three parts according to the increasing class index: head class, middle class, and tail class.

diverse and detailed-rich images, which impacts the generalization of models. Traditional data augmentation methods primarily expand datasets by applying geometric transformations, color adjustments, noise addition, and other operations to the original images. Although these methods increase sample diversity to a certain extent, the additional information they provide is mainly concentrated on superficial changes in image features, lacking the extension of deeper information. Furthermore, when dealing with extremely long-tailed distributions, these methods struggle to significantly enhance the diversity and quality of samples in tail classes. Reference [37] as a prominent generative model has shown outstanding performance in image generation tasks in recent years. Currently, some researchers have attempted to use images generated by diffusion models directly for data augmentation [38], [39]. Nonetheless, when faced with remote sensing data, these approaches exhibit limited efficiency, which might be attributed to the distribution shift between generated images and real images. Therefore, we resort to both utilizing diffusion models to generate new samples, and considering combining these new samples with the original ones to improve their generalization ability.

An effective strategy is to mix the generated images with the original images and use the resulting mixed samples to train the model. This strategy can not only leverage the diversity of generated samples to expand the feature space of the tail class but also introduce the information of the original real images to reduce the risk of using poorly generated images to affect the overall training. Classic data mixing methods include Mixup [40], CutMix [40], and GuidedMixup [41]. These methods usually use interpolation to mix single or multiple images, which can enrich the sample space of the tail class to a certain extent. However, at the same time, it also brings about the omission of salient regions of the image and label ambiguity [42].

To address the above challenges, we propose a diffusion-noise-based augmentation (DONA) method that both effectively expands the diversity of the original dataset and avoids biases caused by training the model on generated data. First, we input the specially designed conditional prompts and the original images into a conditional diffusion model to generate images and filter them by calculating cosine similarity based on the CLIP model. The prompts incorporate various weather conditions and seasons, significantly enhancing data diversity and aligning with real-world conditions. However, training models on synthetic images often overemphasize spurious qualities and biases caused by imperfect generative models. Therefore, we design DiffCam-Mix to counter these challenges. Inspired by the work on feature visualization [43], we use class activation maps (CAMs) to extract salient regions of images, i.e., calculate background masks and foreground masks after the first stage of training. The background mask is used to extract the nonkey part (background) of the generated image, and the foreground mask is used to extract the key part (foreground) of the original image corresponding to the generated image. Finally, the foreground of the original image and the background of the generated image are mixed to obtain mixed data for the second stage of training. In this way, the crucial information of the original image can be retained to solve the deviation problem of directly using the generated data. In addition, we introduce SimSiam contrastive learning, which maximizes the cosine similarity between the mixed data and the corresponding original data in the feature space, bringing them closer together and further correcting this bias.

In general, our main contributions are as follows.

1) We introduce a novel DONA method for long-tailed remote sensing image classification. This method leverages the diffusion model controlled by specialized prompts to generate new images and employs the CLIP model to filter low-quality generated samples. It can generate diverse and high-quality samples to expand the original dataset and is suitable for long-tailed remote sensing image classification tasks.
2) We propose two strategies to effectively utilize generated images. The first strategy is our designed DiffCam-Mix module to generate mixed data for two-stage training. The second strategy is to maximize the similarity between the mixed samples and the corresponding original samples. These strategies can effectively reduce the bias caused by directly using generated images to train the model and improve the overall performance.
3) We conducted extensive experiments on three long-tailed remote sensing datasets and found that our method achieved significant improvements compared with other methods, demonstrating the effectiveness and superiority of our proposed approach.

II. Related Works

In this section, we will mainly introduce the related work in the following four parts: remote sensing image classification, diffusion models for data augmentation, data mixing, and contrastive learning.

A. Remote Sensing Image Classification

Data scarcity and long-tailed distributions are common issues in the real world, especially in remote sensing image

classification. This is because different regions have varying landforms and land use patterns, leading to certain classes frequently appearing in specific areas while being rare in others. This has resulted in two distinct tasks: few-shot remote sensing image classification and long-tailed remote sensing image classification. Regarding few-shot remote sensing image classification, Liu et al. [44] proposed a multiform ensemble self-supervised learning (MES^{2}L) framework, which effectively enhances the performance and efficiency of few-shot remote sensing scene classification through global–local contrastive learning and ensemble enhancement methods. Li et al. [45] introduced a collaborative self-supervised evolution framework (CSENet), which automatically optimizes self-supervised task weights to resolve task conflicts in few-shot remote sensing scene classification, thereby improving classification performance. For the long-tailed remote sensing image classification task, various methods have been proposed from different perspectives. Zhao et al. [46] proposed a novel hierarchical distillation framework (HDF) to solve the problem of long-tailed object recognition in aerial images. Miao et al. [47] proposed a multigranularity decoupling network (MGDNet), which consists of three parts: a multigranularity complementary feature representation (MGCFR) method, a class-imbalanced pseudolabel selection (CIPS) approach, and the diversity component feature (DCF) loss function. MGCFR and CIPS are used for multigranularity feature learning and high-confidence pseudolabel measurement, and DCF loss can make local features more discriminative. Liu et al. [48] proposed a method based on loss reweighting, using the cumulative classification score instead of the number of samples in each class to construct the class weight to improve the accuracy of the tail class. Bai et al. [49] used energy-based discriminators (EDors) to divide the data into head and tail classes, and designed different experts for classification of the head and tail classes. Xie et al. [50] introduced dynamic dissociation (DD) and a multiobjective optimization framework (MOOF) to separate feature learning and classifier learning into two stages, while using learnable feature centroids (LFCs) and masked world knowledge learning (WKL) to improve the learned features. Although these methods can enhance the ability to handle the imbalance between head and tail classes, they suffer from high complexity, poor adaptability to different scenarios, and lack of effective improvement for rare classes.

B. Diffusion Models for Data Augmentation

The diffusion model was first proposed by Sohl-Dickstein et al. [51], but it did not attract much attention at the time. It was not until the proposal of DDPM [37] in 2020 that many subsequent image generation works began to turn to research on diffusion models. In recent years, many studies have begun to focus on how to use diffusion models for data augmentation. Trabucco et al. [38] explored how to effectively augment data by using text-to-image diffusion models (DA-Fusion) to generate images. Azizi et al. [39] demonstrated that using samples generated by fine-tuned diffusion models for training can significantly improve classification accuracy on ImageNet. Feng et al. [52] proposed a new test-time prompt tuning (TPT) method, called DiffTPT, which leverages pretrained diffusion models to generate diverse and informative augmented data. Islam et al. [42] proposed DIFFUSEMIX, a method that utilizes diffusion models to reshape training images and blends generated images, real images, and fractal images for better augmentation. These methods do not take into account the contextual relationships among different land cover types in remote sensing images, making them unsuitable for remote sensing image classification tasks that require a precise understanding of spatial structures.

C. Data Mixing

Data mixing is a classic data augmentation method that can significantly enhance the robustness and generalization capability of deep convolutional neural networks. A well-known approach is Mixup [40], which linearly interpolates pairs of samples and their labels to create new synthetic samples. CutMix [53] improves upon this by cutting and pasting regions from one image to another while adjusting the corresponding labels. Manifold Mixup [54] extends the concept of Mixup to the hidden layers of neural networks, performing linear interpolation at intermediate layers to produce richer feature representations. GuidedMixup [41], based on saliency maps, controls the mixing ratio of each pixel by smoothly interpolating between paired images, thereby better preserving salient regions. Recently, Zheng et al. [55] introduced a data mixing method called CamMix, which achieves stricter class region localization to minimize redundant areas. Data mixing can increase the number of training samples, helping to alleviate the issue of data scarcity in long-tailed distribution. Compared with the above methods, our approach can integrate key information from both original and generated images, preserving the critical semantics of the original images as well as the diversity obtained from the diffusion model.

D. Contrastive Learning

Contrastive learning includes self-supervised contrastive learning and supervised contrastive learning. This method trains by bringing similar samples (positive pairs) closer together and pushing dissimilar samples (negative pairs) further apart. Self-supervised contrastive learning does not rely on label information to construct positive and negative pairs, whereas supervised contrastive learning does the opposite. Common self-supervised contrastive learning methods include SimCLR, MoCo, BYOL, and SimSiam. SimCLR [56] generates positive pairs from different views by using data augmentation and trains the model using a nonlinear projection head and InfoNCE loss. MoCo [57] introduces a momentum-updated encoder queue to store feature representations of negative samples, making contrastive learning more effective on large-scale datasets. BYOL [58] relies only on positive pairs for self-supervised learning of representations, avoiding the negative sample selection issues common in contrastive learning. SimSiam [59] achieves effective representation learning through a simple Siamese network structure and a design of predictor head. In our method, we use self-supervised contrastive learning to calibrate the distribution in the feature space, reducing the inconsistency in sample distribution.

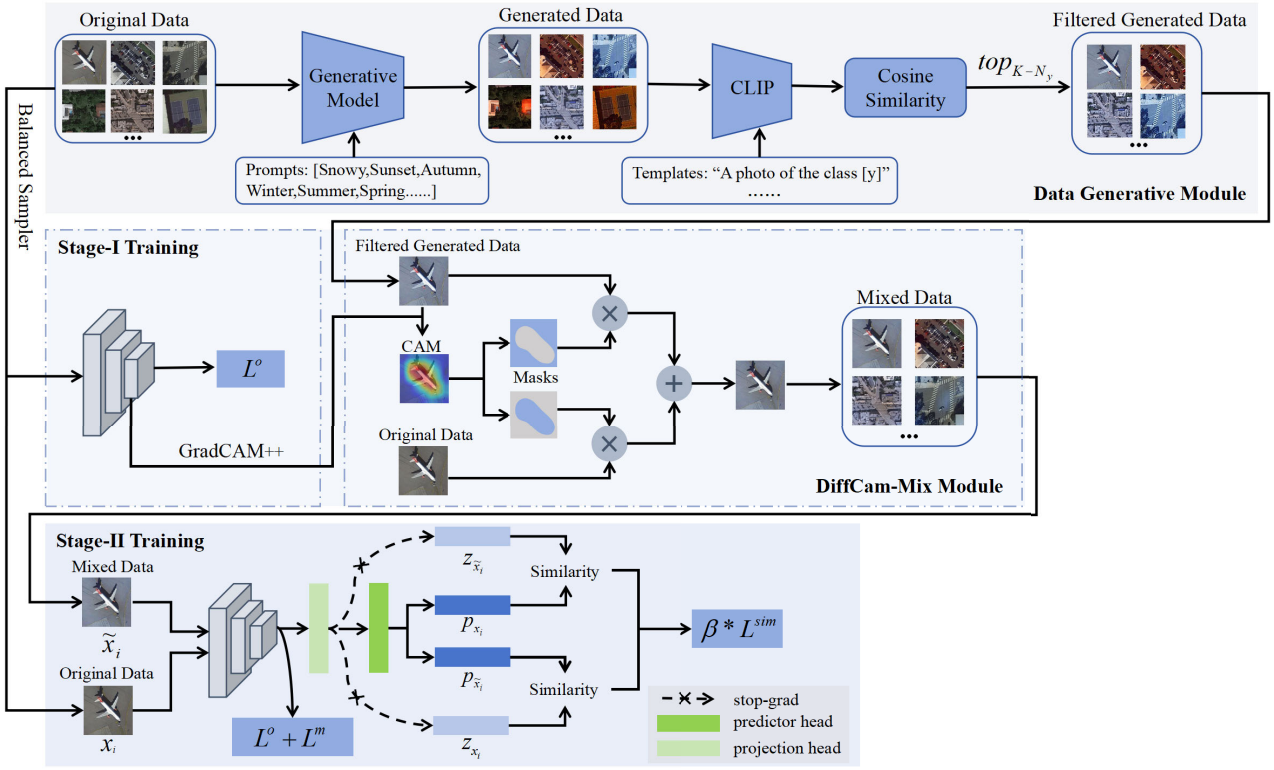


Fig. 2. Framework of our method. Before the model starts training, the data generative module generates new data for subsequent operations. The model training process is divided into two stages. Stage I uses the original data to train the basic feature extractor and classifier. At the end of Stage I, the DiffCam-Mix module is applied to generate mixed data, which will be used together with the original data for Stage II training. During Stage II training, we use the cosine similarity to pull the distance between the mixed data and the corresponding original data in the feature space.

III. Method

A. Preliminaries

In this section, we will introduce the background knowledge of diffusion models and self-supervised contrastive learning.

1) Diffusion Models: The diffusion model works by corrupting the training data by continuously adding Gaussian noise in the forward diffusion process and then learn to recover the data by reversing this process (i.e., the reverse diffusion process). For a given real image sample $x_0 \sim q(x)$, the forward process adds Gaussian noise to it through T accumulations. Finally, we get T noise images $x_1, x_2, \ldots, x_T$. The size of each step is controlled by a series of hyperparameters $\{\gamma_t \in (0,1)\}_{t=1}^{T}$ of the Gaussian distribution variance. Since each moment t in the forward process is only related to the moment $t-1$, it can also be regarded as a Markov process

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\gamma_t}x_{t-1}, \gamma_t \mathbf{I}\right) \tag{1}$$

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}). \tag{2}$$

In this process, as t increases, x_t becomes closer and closer to pure noise. When $T \to \infty$, x_T is completely Gaussian noise.

Contrary to the forward diffusion process, the reverse diffusion process is the denoising process. If we can reverse the above process and sample from $q(x_{t-1} \mid x_t)$, we can restore the original image distribution $x_0 \sim q(x)$ from the Gaussian noise $x_T \sim \mathcal{N}(0, \mathbf{I})$. However, we cannot simply infer $q(x_{t-1} \mid x_t)$, so we need to use a deep learning model to predict such an inverse distribution. The current method mainly uses the U-Net + attention structure.

2) Self-Supervised Contrastive Learning: Self-supervised contrastive learning learns useful feature representations by maximizing the similarity between positive pairs and minimizing the similarity between negative pairs. It usually uses cosine similarity to evaluate the similarity between samples. In a mini-batch, the number of samples is N. Assuming that there is a sample x_i and its positive sample x_p (different augmented view of the same image), and other negative samples x_n (from different samples), the common self-supervised contrastive learning loss is defined as follows:

$$L = -\sum_{i=1}^{2N} \log \frac{\exp\left(\text{sim}\left(z_i, z_p\right)/\tau\right)}{\sum_{n=1}^{2N-1} \exp(\text{sim}(z_i, z_n)/\tau)} \tag{3}$$

where z_i, z_p, and z_n are the feature representations of samples x_i, x_p, and x_n, $\text{sim}(z_i, z_p)$ is the cosine similarity between z_i and z_p, and τ is the temperature parameter.

As self-supervised contrastive learning, the core idea of SimSiam is to learn representations through a pair of identical networks, where the goal of the model is to maximize the similarity between two different augmented views of the same input image. Unlike other self-supervised contrastive learning methods such as SimCLR or MoCo, SimSiam does not require a large number of negative sample pairs or batch sizes.

B. Overview of the Proposed Method

The overall framework of our method is shown in Fig. 2, which mainly includes four parts: data generative module,

Stage I training, DiffCam-Mix module, and Stage II training. Before the feature extractor and classifier start training, the original images and the corresponding prompts are first input into the generative model InstructPix2Pix [60] to generate new images. Considering that the generated images may be of low quality, the CLIP model is subsequently used to filter them in order to improve the performance.

During the training phase, we used a two-stage training method. In the first stage, the original images were first class-balanced resampled for training the feature extractor and classifier. When the first stage of training is completed, the network is now able to basically extract the features of the images, so that they can be effectively identified and classified. After that, the DiffCam-Mix module uses the class activation mapping method Grad-CAM++ to extract the key parts of the image and generate the foreground mask and background mask, respectively. The mask information are used to mix the foreground of the original image and the background of the generated image to generate new mixed data.

In the second stage of training, the original data and the mixed data processed by the DiffCam-Mix module are input into the network, and the network is fine-tuned to further improve the generalization ability. At the same time, SimSiam contrastive learning is used to minimize the negative cosine similarity of the mixed data and its corresponding original data. The purpose is to reduce the distance between the mixed data and the original data in the feature space, thereby calibrating the distribution of the mixed data and further reducing the negative effects of using the generated data for training the model.

C. Data Generative Module

The data generative module uses the pretrained diffusion model InstructPix2Pix as the generation model, which can achieve refined image generation by combining text instructions and diffusion processes. InstructPix2Pix incorporates the knowledge of two large pretrained models during training: a language model (GPT-3) and a text-to-image model (stable diffusion), enabling quick image editing guided by language instructions. We designed a prompt set $P = \{$ Snowy, Sunset, Autumn, Rainy, Winter, Sunny, Summer, Cloudy, Spring , Foggy$\}$ to guide the diffusion model to generate images of different styles. These prompts are related to the weather and seasons and will not cause fundamental changes in the semantic information and spatial structure of the original image, but can enrich the diversity of the original data to a certain extent. We define the prompt set as $P = \{P_1, P_2, \ldots, P_m\}$, and the original image set is defined as I_o. The original images and the prompts will be input into the diffusion model together. For each input image $x_i \in I_o$, the diffusion model generates an image x'_{ij} of the corresponding style based on the prompt $P_j \in P$. x_i is the original image corresponding to the generated image x'_{ij}. Repeat the above generation process, and finally, we get a generated image set I_g. The labels of the generated samples are consistent with the labels of their corresponding original samples.

There are often some low-quality samples in the generated images. These low-quality samples may include blurriness, unnatural textures, or other visual anomalies that deviate from the characteristics of real images. If they are directly used for subsequent training, the performance of the model may be damaged. Therefore, we use the pretrained CLIP model to filter the generated image set I_g. The core idea of CLIP is to train an image encoder and a text encoder via contrastive learning so that related image and text pairs are close to each other in the feature space. Given a dataset containing C classes, define the maximum number of samples for each class as K, the number of original samples of class y ($y \in [0, C-1]$) as N_y, and the template T_y of class y as "a photograph of the class $[y]$." $[y]$ here represents the name of the class y. Using CLIP's image encoder and text encoder, we obtain the image embedding of the generated image x'_{ij} and the text embedding of its corresponding class template T_y, and then calculate the cosine similarity score $S_{x'_{ij}}$ between them

$$S_{x'_{ij}} = \text{Encoder}_{\text{image}}\left(x'_{ij}\right) \cdot \text{Encoder}_{\text{text}}\left(T_y\right). \tag{4}$$

For class y, we select $K - N_y$ samples with the highest similarity score and retain them as high-quality generated images, filtering the remaining samples. Finally, we get the filtered high-quality generated image set I'_g.

D. Stage I Training

The training of the model is mainly divided into two stages. The first stage is the first n epochs, in which the original data are used for feature extractor and classifier training through the class-balanced sampler. The loss function L^o applied in this stage is cross-entropy loss. The resulting trained network is then utilized for subsequent CAM calculations.

E. DiffCam-Mix Module

Using Stage I trained feature extractor and classifier, we perform data mixing operations on the generated data and the corresponding original data. Class activation mapping is a visualization technique used to identify image regions that are important for class-specific judgment in convolutional neural networks. Our DiffCam-Mix module uses Grad-CAM++ to extract the key part (foreground) of the original image and mix it with the nonkey part (background) of the generated image, which can preserve both the key information from the original data and the diversity information from the generated data. Compared with other class activation mapping techniques, such as CAM [61] and Grad-CAM [62], Grad-CAM++ can capture multiple relevant areas in the image.

Specifically, we first calculate the CAM based on the feature activation map and weight information and adjust it to generate foreground masks and background masks. For each image, after the forward and backward propagation, we obtain the gradient of the target class c with respect to the feature map, denoted by $(\partial Y^c / \partial A^k)$, where Y^c is the output score of the target class c and A^k is the activation feature map of the target layer. For each activation map A^k, the gradient weights α_{mn}^{kc} is calculated as follows:

$$\alpha_{mn}^{kc} = \frac{\frac{\partial^2 Y^c}{\left(\partial A_{mn}^k\right)^2}}{2\frac{\partial^2 Y^c}{\left(\partial A_{mn}^k\right)^2} + \sum_a \sum_b A_{ab}^k \left\{\frac{\partial^3 Y^c}{\left(\partial A_{mn}^k\right)^3}\right\}} \tag{5}$$

where (m, n) and (a, b) are the same iterators over the entire activation map A^k. $(\partial^2 Y^c/(\partial A^k_{mn})^2)$ and $(\partial^3 Y^c/(\partial A^k_{mn})^3)$ represent the second-order partial derivative and the third-order partial derivative. For the convenience of calculation, in practice, the square and cube of the first-order gradient are used to replace the second-order gradient and the third-order gradient, respectively. Multiply α^{kc}_{mn} by the gradient after the Rectified Linear Unit (ReLU) activation and sum it over all positions (m, n) to get the final weight w^c_k

$$w^c_k = \sum_m \sum_n \alpha^{kc}_{mn} \cdot \text{ReLU}\left(\frac{\partial Y^c}{\partial A^k_{mn}}\right). \tag{6}$$

The calculated weights are multiplied and summed with the feature map, and then passed through the ReLU layer to obtain CAM

$$\text{CAM} = \text{ReLU}\left(\sum_k w^c_k A^k\right). \tag{7}$$

Then, normalization and resizing are applied for further processing to match the size of the input image. In this way, we get the final CAM $\text{CAM}_{\text{resized}}$. According to the following formula:

$$\boldsymbol{M} = (1 - \text{CAM}_{\text{resized}})^2. \tag{8}$$

We can get the background mask of each generated image, where $\boldsymbol{M} \in \{0, 1\}^{W \times H}$. We define the mixing operation as follows:

$$\tilde{x}_i = \lambda \boldsymbol{M}_{\boldsymbol{x}'_i} \odot x'_i + \left(\mathbf{1} - \lambda \boldsymbol{M}_{\boldsymbol{x}'_i}\right) \odot x_i. \tag{9}$$

The above operations can be used to obtain the mixed data. Here, x_i $(x_i \in I_o)$ is the original image sample and x'_i $(x'_i \in I'_g)$ is the corresponding generated image sample. $\tilde{x}_i$ is the mixed data generated based on the above x_i and x'_i. λ is a hyperparameter, ranging from 0 to 1, which is used to control the mixing ratio of the foreground and the background. $\lambda M_{x'_i}$ is used to calculate the background part of x'_i, $1-\lambda M_{x'_i}$ is used to calculate the foreground part of x_i, and $\odot$ is the elementwise multiplication. Since our operation is performed between two samples of the same class, the labels of the mixed data will not be changed.

F. Stage II Training

After using the DiffCam-Mix module to obtain mixed data, we use these samples and the original samples for the second stage of training to fine-tune the network. At the same time, according to the idea of SimSiam, we regard the original sample x_i and the corresponding mixed sample $\tilde{x}_i$ as two different augmented views and take them as a positive sample pair. Next, these two views are processed by the encoder network consisting of a feature extractor (from Stage I) and a projection head, and the encoder shares weights between the two views. In addition, a predictor head is included to transform the output of one view and match it with the other view. Both the projection head and the predictor head consist of MLPs, and the predictor head is placed after the projection head. In SimSiam, stop gradient is also a crucial mechanism to prevent the network from collapsing into trivial solutions during training. It ensures that only one branch (the predictor branch) can get updated during training.

After extracting features from the sample pair, we send them to the projection head to obtain z_{x_i} and $z_{\tilde{x}_i}$, respectively, and send them to the predictor's head to obtain p_{x_i} and $p_{\tilde{x}_i}$, respectively. Our goal is to minimize the negative cosine similarity between the positive sample pair, which can be expressed as follows:

$$L^{\text{sim}}_i = -\left(\frac{p_{x_i}}{\|p_{x_i}\|_2} \cdot \frac{z_{\tilde{x}_i}}{\|z_{\tilde{x}_i}\|_2} + \frac{p_{\tilde{x}_i}}{\|p_{\tilde{x}_i}\|_2} \cdot \frac{z_{x_i}}{\|z_{x_i}\|_2}\right) \tag{10}$$

where $\|\cdot\|_2$ is the l_2-norm. After the above operations, we can shorten the distance between the mixed sample and its corresponding original sample in the feature space. The final loss function of Stage II training process is as follows:

$$\begin{aligned} L_{\text{total}} &= \beta L^{\text{sim}} + L^o + L^m \\ &= \beta\left(\frac{1}{N}\sum_{i=1}^{N} L^{\text{sim}}_i\right) + \frac{1}{N}\sum_{i=1}^{N} L^o_i + \frac{1}{N}\sum_{i=1}^{N} L^m_i \end{aligned} \tag{11}$$

where β is a hyperparameter used to control the contrastive learning loss. L^o and L^m correspond to the cross-entropy loss of original data and mixed data training, respectively. We believe that the original samples and mixed samples are equally important in the second stage of training, so we assign them the same loss weight.

IV. Experiments

A. Datasets

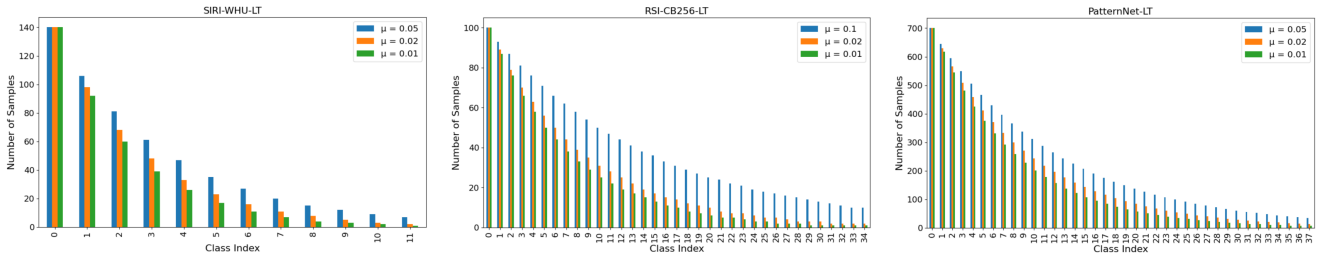
Our experiments are conducted on three long-tailed datasets: SIRI-WHU-LT, PatternNet-LT, and RSI-CB256-LT. We follow the following formula to construct the above three original datasets into long-tailed datasets:

$$N_y = N_{\max} \mu^{\frac{y}{C-1}} \tag{12}$$

where y is the class index, N_y is the number of samples contained in class y, $N_{\max}$ is the number of samples of the class with the most samples, μ is the imbalance factor, and C is the total number of classes. As the class index increases, the number of corresponding samples decreases. Fig. 3 shows the distribution of the three long-tailed datasets mentioned above. The basic information of the three datasets are as follows.

1) SIRI-WHU-LT: SIRI-WHU [63] is a small remote sensing dataset containing 12 scene classes and 200 images per class. Each image is 200 × 200 pixels in size and has a spatial resolution of 2 m. Following [48], we select 60 images from each class as the test set and retain 140 images per class to construct it as a long-tailed dataset with $N_{\max} = 140$. The imbalance factors are set to 0.05, 0.02, and 0.01, respectively. The value of K is set to 200.

2) PatternNet-LT: PatternNet [64] is a large-scale high-resolution remote sensing dataset collected for remote sensing image retrieval. There are 38 classes and each class has 800 images of size 256 × 256 pixels. Following [49], we select 100 images from each class as the testing set, and the remaining images generate a long-tailed dataset, $N_{\max} = 700$. The imbalance factors are set to 0.05, 0.02, and 0.01, respectively.



(a) (b) (c)

Fig. 3. Data distribution of three long-tailed remote sensing datasets. (a) SIRI-WHU-LT dataset. (b) RSI-CB256-LT dataset. (c) PatternNet-LT dataset.

The value of K is set to 800. We categorize the classes with indexes 0–12 as head classes, indexes 13–25 as middle classes, and indexes 26–37 as tail classes.

3) RSI-CB256-LT: RSI-CB256 [65] contains 35 classes and about 24 000 images, each with a resolution of 256 × 256 pixels. Since its distribution is not strictly long tailed, we select 50 samples from each class as the testing set, and the remaining images generate a long-tailed dataset, $N_{\max} = 100$. The imbalance factors are set to 0.1, 0.02, and 0.01, respectively. The value of K is set to 300. We categorize the classes with indexes 0–10 as head classes, indexes 11–21 as middle classes, and indexes 22–34 as tail classes.

B. Implementation Details

For data augment, random cropping, random flipping, brightness, and contrast adjustment are applied to all three datasets. We train our model on three NVIDIA 3090 GPUs. Other implementation details are as follows.

1) Implementation Details on SIRI-WHU-LT: ResNet32 is used as the backbone. The SGD optimizer is used with a momentum of 0.9. The weight decay is $2e^{-4}$, and the initial learning rate is 0.1. Cosine annealing is used to adjust the learning rate. The model is trained for a total of 200 epochs, the first 120 epochs are the first stage of training, and the last 80 epochs are the second stage of training. For the setting of hyperparameters, the mixing ratio λ is set to 0.5, and the contrastive learning loss weight β is set to 1.

2) Implementation Details on PatternNet-LT: ResNet50 is used as the backbone. We use the SGD optimizer with a momentum of 0.9 and the weight decay of $5e^{-3}$. The initial learning rate is 0.01, and the learning rate is adjusted using cosine annealing. The model is trained for a total of 100 epochs, with the first 40 epochs being the first stage of training, and the second stage of training starting from the 40th epoch. In terms of the setting of hyperparameters, the mixing ratio λ is set to 0.5, and the contrastive learning loss weight β is set to 10.

3) Implementation Details on RSI-CB256-LT: ResNet50 is used as the backbone. We use the SGD optimizer with a momentum of 0.9 and a weight decay of $5e^{-3}$. The initial learning rate is 0.01, and the learning rate is adjusted using cosine annealing. The model is trained for a total of 200 epochs, with the first 120 epochs being the first stage of training, and the second stage of training starting at the 120th epoch. The mixing ratio λ is set to 0.5, and the contrastive learning loss weight β is set to 10.

C. Comparison Results

We use top-1 accuracy to evaluate the performance of different models and use t-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize the test data. Different types of long-tailed methods are compared, including LDAM [19], CB [20], BBN [29], BKD [66], BCL [67], AREA [68], CCSMLW [48], CSA [69], GLMC [70], DECOR [50], Mixup [40], Cutmix [53], and DIFFUSEMIX [42].

1) Results on PatternNet-LT: As shown in Table I, compared with other methods, our method improves the performance by 0.44%–2.77%, 1.00%–8.46%, and 1.59%–11.74% when μ is 0.05, 0.02, and 0.01, respectively. When the imbalance factor is 0.05, the sample distribution in the dataset is relatively balanced, so all methods achieve good results. In this case, our method achieves the best performance in the tail class and overall, while also maintaining high accuracy in the head and middle classes. As the imbalance factor decreases to 0.02, the degree of imbalance increases. Although some methods, such as LDAM, DECOR, and Mixup, still perform well in the head and middle classes, their performance in the tail class significantly declines. In comparison, DONA continues to perform well in the tail class, demonstrating its strong ability to handle data imbalance issues. When the imbalance rate drop to 0.01, PatternNet-LT exhibits a high degree of imbalance. While some methods like BCL and GLMC achieve relatively high accuracy in the tail class, DONA performs significantly better in the head and middle classes. Most other methods, such as LDAM, Mixup, Cutmix, and DECOR, are biased toward the head and middle classes, which leads to a decrease in tail class accuracy. In contrast, even under the most extreme imbalance, DONA still demonstrates superiority, achieving a more balanced result.

Fig. 4 displays the samples generated by our method on the PatternNet-LT dataset, as well as the mixed samples. It can be seen that the mixed samples obtained using our method introduce certain transformations based on the original images, while also retaining key semantic information without any omissions. Fig. 5(a)–(c) shows the t-SNE of AREA, CSA, and our method on the PatternNet dataset with the imbalance factor of 0.01. We use points of different colors to represent samples of different classes, and the distribution of points shows the clustering of these classes in the feature space. As can be seen from these figures, all three methods are able to distinguish samples of different classes in the feature space to a certain extent. However, compared with AREA and CSA,

TABLE I
TOP-1 ACCURACY (%) ON PATTERNNET-LT

Method	PatternNet-LT ($\mu = 0.05$)				PatternNet-LT ($\mu = 0.02$)				PatternNet-LT ($\mu = 0.01$)			
	Head	Middle	Tail	All	Head	Middle	Tail	All	Head	Middle	Tail	All
LDAM [19]	99.05±0.24	98.47±0.40	94.51±0.37	97.33±0.45	**99.14±0.08**	97.45±0.17	86.27±0.15	93.78±0.11	99.00±0.07	96.61±0.08	72.67±2.35	88.52±0.60
CB [20]	98.55±0.04	98.92±0.18	96.91±0.19	97.76±0.19	98.11±0.10	97.75±0.14	89.52±1.00	94.69±0.37	96.47±0.69	96.05±0.50	82.00±1.30	90.98±0.36
BBN [29]	97.61±0.21	98.45±0.21	93.27±0.80	95.76±0.60	93.72±1.16	96.33±0.31	85.88±1.26	90.58±0.53	92.81±0.55	95.03±0.47	80.30±2.27	87.64±1.01
BKD [66]	98.78±0.55	98.14±0.69	95.24±0.93	97.26±0.48	98.28±0.69	97.64±0.76	89.33±0.83	94.58±0.17	97.06±0.27	95.61±0.33	86.09±0.39	92.33±0.44
BCL [67]	98.37±0.25	98.10±0.20	97.18±0.07	97.90±0.16	98.22±0.17	97.50±0.41	91.78±0.53	95.74±0.08	96.15±1.17	95.40±0.80	87.42±1.05	92.89±0.53
AREA [68]	99.03±0.14	98.61±0.14	96.91±0.32	98.05±0.07	97.78±0.50	97.75±0.20	91.55±0.71	95.03±0.16	97.66±0.12	97.50±0.18	85.36±0.59	92.90±0.18
CSA [69]	99.36±0.08	98.14±0.50	94.94±0.49	97.49±0.34	99.00±0.42	97.61±0.75	87.27±0.71	94.44±0.25	**99.34±0.12**	97.83±0.41	78.39±0.41	91.92±0.34
GLMC [70]	95.75±0.92	98.75±0.08	97.14±0.05	97.06±0.32	94.05±0.34	97.78±0.43	92.21±0.24	94.17±0.25	91.50±0.38	95.61±0.39	87.58±0.70	90.80±0.46
DECOR [50]	**99.79±0.04**	99.04±0.29	95.50±0.50	98.06±0.14	99.09±0.59	97.38±1.05	80.77±1.32	92.37±0.42	98.46±0.21	93.25±1.08	69.87±0.87	86.86±0.17
Mixup [40]	98.98±0.10	98.51±0.19	91.19±0.51	96.36±0.26	97.90±0.24	96.30±0.11	69.17±0.45	88.28±0.18	97.92±0.23	93.44±0.40	55.22±0.22	82.91±0.21
Cutmix [53]	99.21±0.10	99.15±0.23	94.89±0.43	97.83±0.18	99.02±0.15	98.23±0.12	83.33±0.65	93.80±0.19	98.51±0.13	97.03±0.19	67.97±0.24	88.36±0.11
DIFFUSEMIX [42]	99.20±0.12	99.12±0.12	95.82±0.27	98.09±0.17	98.92±0.25	98.54±0.29	89.50±0.68	95.70±0.20	98.79±0.04	**98.21±0.21**	81.59±1.23	93.06±0.41
DONA (Ours)	99.09±0.31	**99.19±0.43**	**97.58±0.19**	**98.53±0.13**	98.78±0.17	**98.89±0.10**	**93.33±0.16**	**96.74±0.08**	98.61±0.42	98.19±0.61	**87.94±0.18**	**94.65±0.07**

TABLE II
TOP-1 ACCURACY (%) ON RSI-CB256-LT

Method	RSI-CB256-LT ($\mu = 0.1$)				RSI-CB256-LT ($\mu = 0.02$)				RSI-CB256-LT ($\mu = 0.01$)			
	Head	Middle	Tail	All	Head	Middle	Tail	All	Head	Middle	Tail	All
LDAM [19]	94.60±0.28	95.13±0.41	88.28±0.52	91.73±0.27	**94.00±0.16**	**93.27±0.25**	58.28±0.08	79.53±0.10	**94.53±0.09**	85.07±0.09	35.67±0.27	68.97±0.12
CB [20]	92.60±0.33	95.45±0.68	91.50±0.41	92.68±0.33	78.33±0.82	86.40±0.28	67.39±0.42	75.71±0.04	72.30±0.30	81.40±0.80	45.59±1.09	64.63±0.23
BBN [29]	86.47±0.50	91.73±0.19	90.44±0.34	89.27±0.27	74.53±0.66	83.00±0.33	61.39±0.90	70.69±0.38	72.60±1.28	73.20±3.05	45.72±0.67	61.14±0.89
BKD [66]	90.87±0.09	94.56±0.60	90.87±1.24	91.75±1.03	88.42±0.69	87.35±0.48	70.29±0.14	79.90±0.30	89.50±0.50	82.59±0.41	47.78±0.55	71.03±0.68
BCL [67]	92.60±0.57	**95.87±0.09**	91.22±0.21	92.80±0.19	86.79±0.44	91.13±0.25	69.26±0.73	80.65±0.48	83.74±0.10	85.00±0.99	50.02±0.39	71.09±0.45
AREA [68]	92.40±0.33	94.00±0.16	90.39±0.34	91.73±0.16	88.93±0.57	89.80±0.28	69.67±0.36	80.67±0.16	84.40±0.20	83.10±0.90	51.42±0.43	70.31±0.77
CSA [69]	95.00±0.28	95.47±0.74	90.00±0.27	92.88±0.19	93.00±0.42	90.59±1.46	64.27±0.65	80.44±0.61	92.80±0.16	87.53±0.09	43.39±0.16	71.92±0.15
GLMC [70]	86.30±1.30	95.70±0.70	91.75±0.08	90.97±0.29	76.30±1.10	85.80±1.80	69.31±0.69	76.15±1.05	61.07±0.75	74.61±1.43	50.16±1.35	60.89±0.47
DECOR [50]	**95.60±0.20**	94.12±0.67	87.52±2.02	92.09±0.77	93.73±0.09	88.80±0.28	56.00±1.36	77.05±0.45	93.07±0.34	82.47±0.38	41.92±1.08	68.67±0.77
Mixup [40]	89.39±0.17	90.97±0.17	67.38±0.88	81.71±0.29	87.58±0.23	72.85±0.45	23.95±0.48	59.31±0.12	86.73±0.65	61.64±1.51	11.23±0.12	50.80±0.72
Cutmix [53]	94.00±0.54	95.46±0.26	90.31±0.13	93.09±0.28	91.88±0.45	88.42±0.23	54.26±0.32	76.82±0.30	90.54±0.39	80.73±0.39	37.33±0.65	67.70±0.40
DIFFUSEMIX [42]	94.60±0.20	92.80±0.60	85.09±0.59	90.11±0.40	90.10±0.90	85.50±1.10	61.92±0.25	78.11±0.40	92.40±0.20	84.70±0.50	48.66±1.16	72.72±0.38
DONA (Ours)	94.93±0.25	95.60±0.57	**94.56±0.28**	**94.34±0.19**	92.65±0.46	92.70±0.64	**71.91±0.43**	**83.81±0.24**	89.60±1.84	**88.33±0.66**	**52.72±1.52**	**74.49±0.22**

Fig. 4. Different images generated by the proposed method on the PatternNet-LT dataset.

our method can generate a more compact representation, and the separation between classes is more obvious. In summary, on the PatternNet-LT dataset, our method has significantly improved the classification performance, especially for the tail class. It can greatly improve the impact caused by data imbalance and enhance the generalization ability of the model.

2) Results on RSI-CB256-LT: As shown in Table II, compared with other methods, our method improves the performance by 1.25%–12.63%, 3.14%–24.50%, and 1.77%–23.69% when μ is 0.1, 0.02, and 0.01, respectively. When the imbalance factor is 0.1, DECOR achieves the highest accuracy in the head class, while BCL performs best in the middle class. When the imbalance factor is 0.02, LDAM achieves the highest accuracy in both the head and middle classes. With the imbalance factor of 0.01, LDAM achieves the best performance in the head class. Overall, DECOR and LDAM tend to favor the head class as data imbalance increases, with their tail class accuracy being significantly lower than other compared methods. CB, GLMC, and BBN sacrifice head class accuracy to improve performance in the middle and tail classes, and their performance on the head class is significantly lower than other methods. BKD, BCL, AREA, and CSA perform well in both the head and tail classes, but their performance in the tail class is not as stable as our method. As the degree of imbalance in the dataset increases, data mixing-based methods like Mixup and Cutmix show a significant decline in accuracy in the tail class. DIFFUSEMIX performs well, but overall, it is still inferior to our method. In contrast, our method stands out in the tail class, particularly when μ is smaller, showing better handling of long-tailed distribution while maintaining high performance in both the head and middle classes.

Fig. 5(d)–(f) shows the t-SNE of AREA, CSA and our method on the RSI-CB256-LT dataset when $\mu = 0.01$. Due to the extreme imbalance of samples in the tail class, there is a certain overlap in features between classes in both methods. In this case, compared with AREA and CSA, the features learned by our method are more compact within the same class, and the distribution of sample points between classes shows a certain degree of separation, with the overlap between

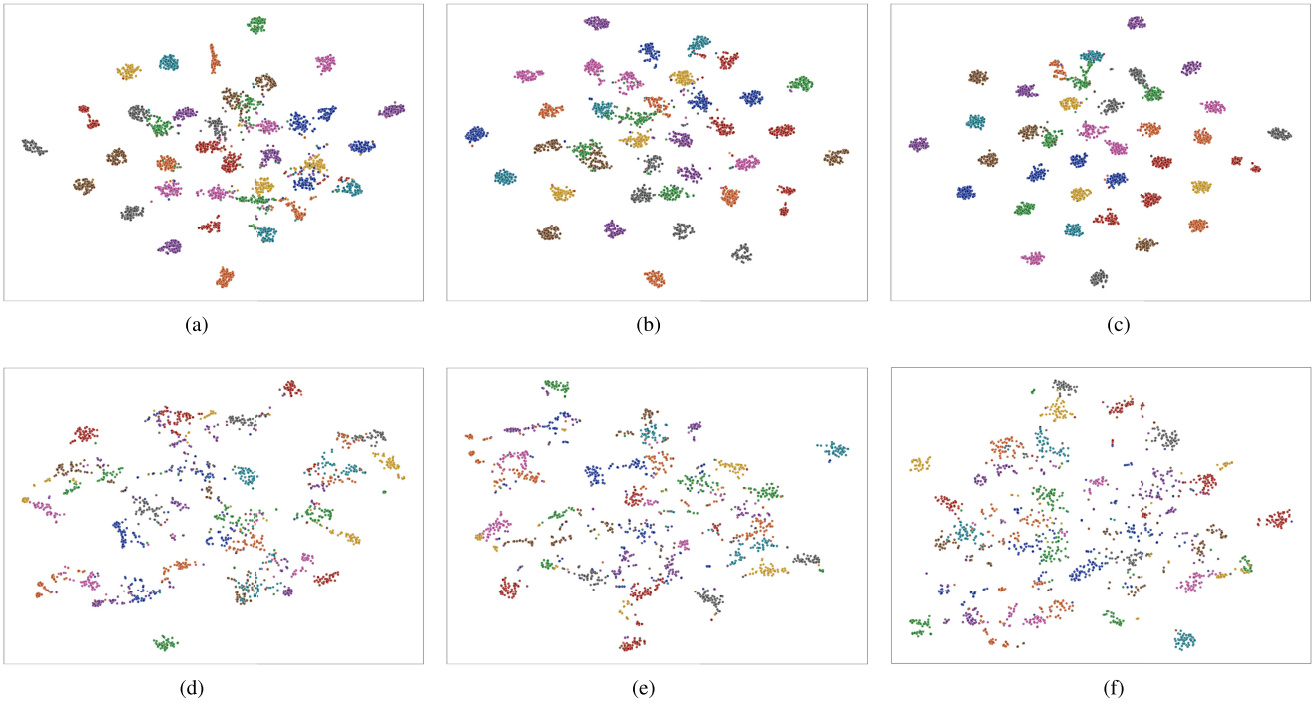


Fig. 5. t-SNE for PatternNet-LT ($\mu = 0.01$) and RSI-CB256-LT ($\mu = 0.01$). (Left to right) Results of AREA, CSA, and our method on these two datasets. (a) AREA (PatternNet-LT $\mu = 0.01$). (b) CSA (PatternNet-LT $\mu = 0.01$). (c) Ours (PatternNet-LT $\mu = 0.01$). (d) AREA (RSI-CB256-LT $\mu = 0.01$). (e) CSA (RSI-CB256-LT $\mu = 0.01$). (f) Ours (RSI-CB256-LT $\mu = 0.01$).

classes also being alleviated. In a word, on the RSI-CB256-LT dataset, our method has a stronger generalization ability under imbalanced data distribution.

3) Results on SIRI-WHU-LT: As shown in Table III, compared with other methods, our method improves by 4.35%–27.50%, 3.10%–36.29%, and 5.15%–30.77% when the imbalance factor is 0.05, 0.02, and 0.01, respectively. Most methods, such as CSA, AREA, GLMC, and Cutmix, can achieve high accuracy on datasets with lower levels of imbalance. However, as the degree of imbalance increases, their performance also decreases. Mixup consistently performs worse than other methods across all datasets with three imbalance factors. DIFFUSEMIX shows relatively good performance when the data are extremely imbalanced, but it still lags behind DONA. Our method achieves the best results on datasets with three imbalance factors, which fully demonstrates its effectiveness. Even though the data is extremely imbalanced, our method can also remain stable.

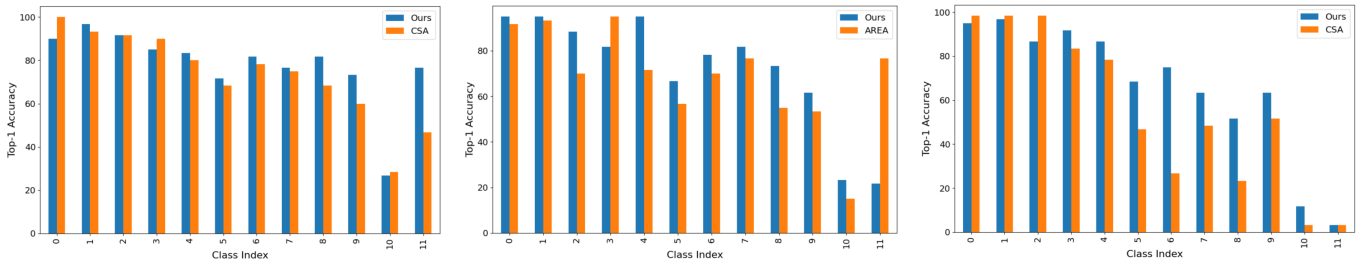
Fig. 6 shows the top-1 accuracy of CSA, AREA, and our proposed method for samples of each class in the SIRI-WHU-LT dataset, with imbalance factors of 0.05, 0.02, and 0.01, respectively. Compared with CSA and AREA, our method can better identify tail class samples under different degrees of imbalance. When the imbalance factor is 0.01, there are very few tail class samples in the dataset, but our method can still achieve relatively high accuracy, which shows that our method has strong generalization ability in the case of long-tailed distribution. Fig. 7 displays the confusion matrix of our DONA method trained on the SIRI-WHU-LT dataset with an imbalance factor of 0.05. The dark diagonal elements demonstrate the model's accurate classification for most samples. However, some tail classes (e.g., Class 10 River and Class 11 Water) show misclassification with head classes (e.g., Class 2 Harbor), which may be attributed to their similar feature representations.

TABLE III
TOP-1 ACCURACY (%) ON SIRI-WHU-LT

Dataset	SIRI-WHU-LT		
Imbalance Factor	0.05	0.02	0.01
LDAM [19]	59.84±0.43	35.56±0.41	43.77±0.88
CB [20]	67.81±0.52	54.09±0.83	49.98±1.24
BBN [29]	68.75±0.41	54.86±0.52	51.80±0.60
BKD [66]	70.51±0.98	65.72±0.40	46.33±0.44
BCL [67]	68.19±0.11	54.81±0.17	42.73±0.31
CSA [69]	73.33±0.34	62.08±0.34	55.21±0.45
AREA [68]	72.59±0.35	68.75±0.51	54.49±0.57
CCSMLW [48]	67.35±0.36	54.91±0.72	50.71±1.07
GLMC [70]	72.45±0.29	56.43±0.47	45.07±0.43
Mixup [40]	50.18±0.57	47.27±0.34	34.65±0.35
Cutmix [53]	69.03±0.50	53.19±0.52	44.81±0.24
DIFFUSEMIX [42]	68.68±0.49	64.23±0.20	60.27±0.55
DONA (Ours)	**77.68±0.47**	**71.85±0.17**	**65.42±0.49**

D. Ablation Studies

In this section, we conduct ablation experiments to demonstrate the effectiveness of the data generative module, DiffCam-Mix module, and contrastive learning. We conducted experiments on PatternNet-LT ($\mu = 0.01$) and RSI-CB256-LT ($\mu = 0.1$). Generated data mean that the model is trained by using generated images without any other operations. CLIP means that the generated images are filtered by using CLIP. DiffCam-Mix represents the DiffCam-Mix module we proposed, and L^{sim} means that contrastive learning is used in the second stage of training. ✓ means adding the module.



(a) (b) (c)

Fig. 6. Top-1 accuracy of CSA, AREA, and our method for each class of samples are compared on the SIRI-WHU-LT dataset when the imbalance factor is 0.05, 0.02, and 0.01. respectively. (a) $\mu = 0.05$. (b) $\mu = 0.02$. (c) $\mu = 0.01$.

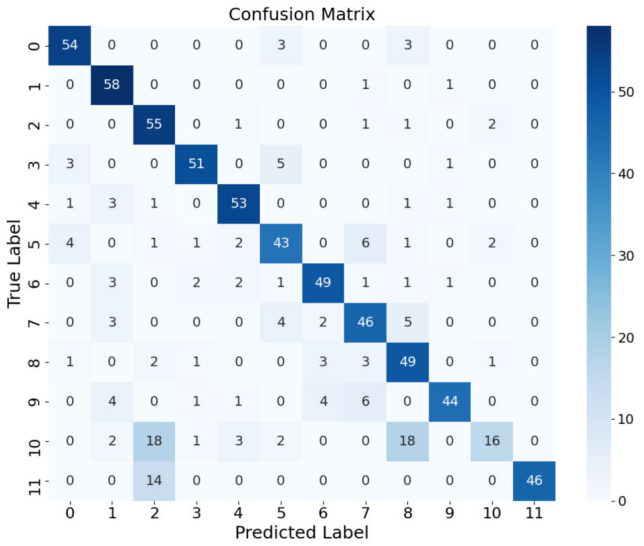


Fig. 7. Confusion matrix of the model trained on the SIRI-WHU-LT dataset with an imbalance factor of 0.05.

The first row of Tables IV and V represents the models trained on the original dataset using only class-balanced resampling.

1) Data Generative Module: Observing the first and second rows of Tables IV and V, we can see that after training the model with images generated by the prompt we designed, the overall accuracy improved by 2.12% and 4.20%, respectively, especially on the RSI-CB256-LT dataset, where both the middle and tail classes showed significant improvement. This indicates that using images generated by the diffusion model for data augmentation can improve the model's performance to some extent. However, for the PatternNet-LT dataset, the accuracy improvement is mainly concentrated in the head and middle classes, with improvements of 5.54% and 3.80%, respectively. In contrast, the accuracy of the tail class decreased by 4.75%. This may be due to the poor quality of the generated tail class images in the PatternNet-LT dataset. Low-quality generated images may introduce unnecessary noise into the model, especially when there are fewer tail class samples, and the impact of noise becomes more significant. As a result, it becomes more difficult for the model to learn the tail class, leading to a decrease in accuracy. Therefore, we used the CLIP model to filter the generated images and retain those with higher quality. It can be observed that after introducing CLIP, the accuracy of the tail class improved by 2.96% and 0.67% on the two datasets, respectively. In addition to further balancing the model's performance, the overall accuracy increased by 0.98% and 0.97%, which proves the necessity of using CLIP for filtering generated images.

2) DiffCam-Mix Module: There is a domain gap between generated samples and real samples, which may aggravate the impact of the long-tailed effect on the testing set [71]. Our DiffCam-Mix module can further alleviate the class imbalance. As can be seen from Tables IV and V, after applying the DiffCam-Mix module to generate mixed data for training, the accuracy of both datasets has improved. In comparison, the tail class on PatternNet-LT increased by 3.45%, and the middle and tail classes on RSI-CB256-LT increased by 0.40% and 4.00%, respectively. The overall accuracy of the two datasets increased by 0.83% and 1.54%. From the above results, it can be seen that DiffCam-Mix can effectively use the generated data and the original data to balance the feature space, thereby improving the performance of the model.

3) Contrastive Learning: After using the strategy of SimSiam contrastive learning, the overall accuracy on the PatternNet-LT dataset was further improved by 0.59%, and the accuracy of the tail class was further improved by 3.03%. The overall accuracy on the RSI-CB256-LT dataset increased by 0.86%, the head class increased by 0.63%, the middle class increased by 0.60%, and the tail class increased by 1.56%. The more balanced distribution of samples from each class in the feature space led to an overall performance boost, thereby demonstrating the effectiveness of this strategy. Through contrastive learning, the distance between the mixed sample and its corresponding original sample in the feature space can be shortened, so that the model can more effectively capture the useful information in the mixed sample and make it closer to the distribution of real data. In addition, this method enables the model to learn the overall representation of the class during training, thereby improving the generalization ability of the model.

Each part of the above modules plays a vital role, and they work together to ensure that the model can learn a balanced feature space during training.

E. Influence of Hyperparameters and Runtime Analysis

1) Influence of the Hyperparameter β: β is used to control the contrastive learning loss in the second stage of training. Following the setting of [70], its value is 1 or 10. If the value of

TABLE IV
ABLATION STUDY ON PATTERNNET-LT (%)

Module				PatternNet-LT ($\mu = 0.01$)			
Generated Data	CLIP	DiffCam-Mix	L^{sim}	Head	Middle	Tail	All
				93.80±1.04	94.86±1.36	83.25±1.31	90.13±1.43
✓				**99.34±0.09**	98.66±0.09	78.50±0.41	92.25±0.22
✓	✓			99.14±0.04	**98.67±0.07**	81.46±1.05	93.23±0.18
✓	✓	✓		99.04±0.04	98.58±0.25	84.91±0.55	94.06±0.20
✓	✓	✓	✓	98.61±0.42	98.19±0.61	**87.94±0.18**	**94.65±0.07**

TABLE V
ABLATION STUDY ON RSI-CB256-LT (%)

Module				RSI-CB256-LT ($\mu = 0.1$)			
Generated Data	CLIP	DiffCam-Mix	L^{sim}	Head	Middle	Tail	All
				94.20±0.20	81.50±1.10	85.16±1.84	86.77±0.20
✓				93.90±0.30	92.10±1.30	88.33±0.50	90.97±0.63
✓	✓			94.50±0.70	94.60±0.80	89.00±0.17	91.94±0.40
✓	✓	✓		94.30±0.50	95.00±0.20	93.00±0.67	93.48±0.22
✓	✓	✓	✓	**94.93±0.25**	**95.60±0.57**	**94.56±0.28**	**94.34±0.19**

TABLE VI
COMPARISON OF TOP-1 ACCURACY (%) BASED ON DIFFERENT β ON THE DATASET PATTERNNET-LT ($\mu = 0.01$)

β	Head	Middle	Tail	All
β = 1	**99.22±0.08**	**98.67±0.18**	84.97±0.70	94.11±0.31
β = 10	98.61±0.42	98.19±0.61	**87.94±0.18**	**94.65±0.07**

β is relatively large, it indicates that the model focuses more on contrastive learning during training, which means that it encourages the mixed data to be closer to the original data in the feature space, making the mixed data and the original data more similar in the feature space. If the value of β is small, the model relies more on the standard classification task and reduces the impact of contrastive learning. 1 and 10 represent the two different strategies and weight adjustment methods mentioned above. When β is 1, it means that the impact of contrastive learning on the loss function is smaller during training, leading to a more balanced focus between the classification task and contrastive learning. On the other hand, when β is 10, it indicates that the weight of the contrastive loss is significantly increased, and the model will pay more attention to the distinction between samples in the feature space, with the mixed data becoming closer to its corresponding original data in the feature space. Table VI shows the top-1 accuracy on the PatternNet-LT ($\mu = 0.01$) dataset when β is 1 and 10. We can see that in this case, when β is 10, the performance of the model is more balanced. The choice of β depends on the model's performance on different datasets. In most cases, we set it to 10 to effectively reduce the discrepancy between the original data and the mixed data.

2) Influence of the Hyperparameter λ: In the DiffCam-Mix module, λ is used to control the mixing ratio of the background of the generated image and the foreground of its corresponding original image, which ranges from 0 to 1. We tested the top-1 accuracy under different λ on the SIRI-WHU-LT dataset, and the results are shown in Table VII. We can see that on the datasets with three imbalance factors, the accuracy reaches a peak when the value of λ is 0.5. When λ approaches 1, the proportion of generated images is higher, and the accuracy on the three datasets decreases. This shows that relying entirely on generated images may reduce model performance because the generated images cannot fully reflect the characteristics of the original data, and excessive introduction of generated images may bring noise to the training of the model. As λ approaches 0.1, the proportion of original images increases, leading to a decline in accuracy across the three datasets. This is due to the lack of diversity in the mixed images, which limits the ability to expand the feature space and, in turn, restricts the model's generalization capability. To ensure a balanced ratio between the generated images and the original images in the mixed data, we set the value of λ to 0.5 in the previous experiments, which is a general and effective setting.

3) Analysis of Running Time and Parameters: Table VIII shows the execution time for each epoch of our method on three long-tailed remote sensing datasets, where Stage I represents the execution time for each epoch in the first stage, and Stage II represents the execution time for each epoch in the second stage. Due to differences in the size and complexity of the datasets, the execution times vary significantly. PatternNet-LT, with a larger number of training samples, takes more time to train, while RSI-CB256-LT and SIRI-WHU-LT, with fewer samples, take less time. During the model training process, the second stage requires both original and mixed samples, which involves loading more data; therefore, Stage II typically takes longer than Stage I. Table IX presents the comparison of parameter efficiency, where all methods adopt ResNet50 as the backbone network. Our method demonstrates significantly lower parameter count (Parameters) and floating-point operations per second (FLOPs) compared with BCL, DECOR,

TABLE VII
COMPARISON OF TOP-1 ACCURACY (%) BASED ON DIFFERENT λ ON THE DATASET SIRI-WHU-LT

λ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\mu = 0.05$	74.07±1.25	74.95±0.99	75.77±0.62	76.45±0.48	**77.68±0.47**	75.42±0.82	75.65±0.94	76.16±0.29	76.39±1.15	75.23±2.13
$\mu = 0.02$	69.54±0.36	69.26±1.02	70.07±0.21	69.24±0.07	**71.85±0.17**	70.46±0.76	69.58±0.98	70.21±0.21	70.76±0.07	70.14±0.83
$\mu = 0.01$	63.47±1.18	65.07±0.21	64.65±0.07	63.75±0.69	**65.42±0.49**	63.70±0.82	63.71±1.21	64.26±0.68	64.21±0.36	62.59±0.86

TABLE VIII
EXECUTION TIME (IN SECONDS) FOR ONE EPOCH OF TRAINING ON THREE DIFFERENT DATASETS

Dataset	Imbalance Factor	Stage-I	Stage-II
PatternNet-LT	0.05	42.48	135.21
	0.02	34.03	107.32
	0.01	31.16	93.37
RSI-CB256-LT	0.1	5.61	35.53
	0.02	3.40	22.74
	0.01	3.31	19.39
SIRI-WHU-LT	0.05	2.04	12.53
	0.02	1.86	10.54
	0.01	1.67	9.76

TABLE IX
ANALYSIS OF PARAMETER EFFICIENCY. P DENOTES THE PARAMETER COUNT. FLOPS REPRESENTS THE NUMBER OF MULTIPLY-AND-ACCUMULATE OPERATIONS

	ResNet50	BCL	DECOR	DONA
FLOPs	4133.74M	10437.64M	16544.39M	4132.28M
P	25.56M	36.11M	32.32M	24.10M

and the original ResNet50, while simultaneously achieving effective performance improvement.

V. CONCLUSION

In this article, we first analyze the long-tailed data distribution problem in remote sensing image classification, and then discuss the existing solutions and the limitations of these methods. On this basis, we propose a DONA method. The core idea of DONA is to fuse the key semantic information of the original image with the diversity information of the generated image as additional samples to expand the original dataset, thereby solving the problem of limited tail class samples. To avoid the problem of inconsistent sample distribution introduced by directly using generated images to train the model, we propose two strategies to calibrate the distribution in the feature space through DiffCam-Mix and self-supervised contrastive learning, respectively. DONA has shown strong performance on three long-tailed remote sensing datasets, proving its effectiveness. However, our method also has certain limitations. First, data generation and mixing operations require additional overhead, and using mixed data for training in the second stage takes more time. Second, the quality of the generated data needs to be further ensured, especially for other long-tailed image classification tasks, such as fine-grained recognition, where it is crucial to ensure that the generated images accurately capture detailed information. In the future, we will continue to explore the application of diffusion models in data augmentation to enhance the versatility of DONA and design more effective methods.

REFERENCES

[1] G. Cheng, L. Guo, T. Zhao, J. Han, H. Li, and J. Fang, "Automatic landslide detection from remote-sensing imagery using a scene classification method based on BoVW and pLSA," *Int. J. Remote Sens.*, vol. 34, no. 1, pp. 45–59, Jan. 2013.

[2] T. Zhang and X. Huang, "Monitoring of urban impervious surfaces using time series of high-resolution remote sensing images in rapidly urbanized areas: A case study of Shenzhen," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 8, pp. 2692–2708, Aug. 2018.

[3] B. Zhao et al., "Intermediate domain prototype contrastive adaptation for spartina alterniflora segmentation using multitemporal remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5401314.

[4] N. Longbotham, C. Chaapel, L. Bleiler, C. Padwick, W. J. Emery, and F. Pacifici, "Very high resolution multiangle urban classification analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 4, pp. 1155–1170, Apr. 2012.

[5] A. Tayyebi, B. C. Pijanowski, and A. H. Tayyebi, "An urban growth boundary model using neural networks, GIS and radial parameterization: An application to Tehran, Iran," *Landscape Urban Planning*, vol. 100, nos. 1–2, pp. 35–44, Mar. 2011.

[6] T. Wellmann et al., "Remote sensing in urban planning: Contributions towards ecologically sound policies?" *Landscape Urban Planning*, vol. 204, Dec. 2020, Art. no. 103921.

[7] W. Xie, W. Shao, D. Li, Y. Li, and L. Fang, "MIFNet: Multi-scale interaction fusion network for remote sensing image change detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 35, no. 3, pp. 2725–2739, Mar. 2025.

[8] L. Hu, S. Li, J. Ruan, and F. Gao, "SemiPSENet: A novel semi-supervised change detection network for remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5634214.

[9] Y. Huang, L. Tang, D. Jing, Z. Li, Y. Tian, and S. Zhou, "Research on crop planting area classification from remote sensing image based on deep learning," in *Proc. IEEE Int. Conf. Signal, Inf. Data Process. (ICSIDP)*, Dec. 2019, pp. 1–4.

[10] H. Shirmard, E. Farahbakhsh, R. D. Müller, and R. Chandra, "A review of machine learning in processing remote sensing data for mineral exploration," *Remote Sens. Environ.*, vol. 268, Jan. 2022, Art. no. 112750.

[11] F. Ghazouani, I. R. Farah, and B. Solaiman, "A multi-level semantic scene interpretation strategy for change interpretation in remote sensing imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 8775–8795, Nov. 2019.

[12] Y. Wang et al., "A three-layered graph-based learning approach for remote sensing image retrieval," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6020–6034, Oct. 2016.

[13] X. Yang, W. Yan, W. Ni, X. Pu, H. Zhang, and M. Zhang, "Object-guided remote sensing image scene classification based on joint use of deep-learning classifier and detector," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 2673–2684, 2020.

[14] Y. Li, Z. Zhu, J.-G. Yu, and Y. Zhang, "Learning deep cross-modal embedding networks for zero-shot remote sensing image scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 12, pp. 10590–10603, Dec. 2021.

[15] X. Zhao, J. Zhang, J. Tian, L. Zhuo, and J. Zhang, "Residual dense network based on channel-spatial attention for the scene classification of a high-resolution remote sensing image," *Remote Sens.*, vol. 12, no. 11, p. 1887, Jun. 2020.

[16] J. Wang, M. Zhang, W. Li, and R. Tao, "A multistage information complementary fusion network based on flexible-mixup for HSI-X image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 12, pp. 17189–17201, Dec. 2024.

[17] X. Ma et al., "An ultralightweight hybrid CNN based on redundancy removal for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5506212.

[18] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng, "Deep long-tailed learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 10795–10816, Sep. 2023.

[19] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," in *Proc. NIPS*, 2019, pp. 1565–1576.

[20] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9268–9277.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[22] J. Ren et al., "Balanced meta-softmax for long-tailed visual recognition," in *Proc. NIPS*, 2020, pp. 4175–4186.

[23] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, "Feature transfer learning for face recognition with under-represented data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5697–5706.

[24] S. Parisot, P. M. Esperanca, S. McDonagh, T. J. Madarasz, Y. Yang, and Z. Li, "Long-tail recognition via compositional knowledge transfer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 6929–6938.

[25] H.-P. Chou, S.-C. Chang, J.-Y. Pan, W. Wei, and D.-C. Juan, "Remix: Rebalanced mixup," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 95–110.

[26] Y. Zang, C. Huang, and C. Change Loy, "FASA: Feature augmentation and sampling adaptation for long-tailed instance segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 3437–3446.

[27] B. Kang et al., "Decoupling representation and classifier for long-tailed recognition," 2019, *arXiv:1910.09217*.

[28] B. Kang, Y. Li, S. Xie, Z. Yuan, and J. Feng, "Exploring balanced feature spaces for representation learning," in *Proc. Int. Conf. Learn. Representations*, May 2021, pp. 1–15.

[29] B. Zhou, Q. Cui, X.-S. Wei, and Z.-M. Chen, "BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9716–9725.

[30] X. Wang, L. Lian, Z. Miao, Z. Liu, and S. X. Yu, "Long-tailed recognition by routing diverse distribution-aware experts," 2020, *arXiv:2010.01809*.

[31] P. Wang, K. Han, X.-S. Wei, L. Zhang, and L. Wang, "Contrastive learning based hybrid networks for long-tailed image classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 943–952.

[32] T.-Y. Wu, P. Morgado, P. Wang, C.-H. Ho, and N. Vasconcelos, "Solving long-tailed recognition with deep realistic taxonomic classifier," 2020, *arXiv:2007.09898*.

[33] M. Wang, H. Chen, D. Shen, B. Li, and S. Hu, "RSRNeT: A novel multi-modal network framework for named entity recognition and relation extraction," *PeerJ Comput. Sci.*, vol. 10, p. e1856, Feb. 2024.

[34] Y. Liu, S. Yan, L. Leal-Taixé, J. Hays, and D. Ramanan, "Soft augmentation for image classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 16241–16250.

[35] S. G. Müller and F. Hutter, "TrivialAugment: Tuning-free yet state-of-the-art data augmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 774–782.

[36] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "RandAugment: Practical automated data augmentation with a reduced search space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 3008–3017.

[37] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020, *arXiv:2006.11239*.

[38] B. Trabucco, K. Doherty, M. Gurinas, and R. Salakhutdinov, "Effective data augmentation with diffusion models," 2023, *arXiv:2302.07944*.

[39] S. Azizi, S. Kornblith, C. Saharia, M. Norouzi, and D. J. Fleet, "Synthetic data from diffusion models improves ImageNet classification," 2023, *arXiv:2304.08466*.

[40] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.

[41] M. Kang and S. Kim, "GuidedMixup: An efficient mixup strategy guided by saliency maps," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, Jun. 2023, pp. 1096–1104.

[42] K. Islam, M. Z. Zaheer, A. Mahmood, and K. Nandakumar, "Diffusemix: Label-preserving data augmentation with diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 27611–27620.

[43] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 839–847.

[44] Y. Liu et al., "Collaborative self-supervised evolution for few-shot remote sensing scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 4509215.

[45] J. Li, M. Gong, H. Liu, Y. Zhang, M. Zhang, and Y. Wu, "Multi-form ensemble self-supervised learning for few-shot remote sensing scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 4500416.

[46] W. Zhao, J. Liu, Y. Liu, F. Zhao, Y. He, and H. Lu, "Teaching teachers first and then student: Hierarchical distillation to improve long-tailed object recognition in aerial images," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5624412.

[47] W. Miao, J. Geng, and W. Jiang, "Multigranularity decoupling network with pseudolabel selection for remote sensing image scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 5603813.

[48] J. Liu, R. Feng, P. Chen, X. Wang, and Y. Ni, "Dynamic loss reweighting method based on cumulative classification scores for long-tailed remote sensing image classification," *Remote Sens.*, vol. 15, no. 2, p. 394, Jan. 2023.

[49] Y. Bai, S. Shao, S. Zhao, W. Liu, D. Tao, and B. Liu, "EME: Energy-based multiexpert model for long-tailed remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 4701812.

[50] J. Xie et al., "DECOR: Dynamic decoupling and multiobjective optimization for long-tailed remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5612517.

[51] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," 2015, *arXiv:1503.03585*.

[52] C.-M. Feng, K. Yu, Y. Liu, S. Khan, and W. Zuo, "Diverse data augmentation with diffusions for effective test-time prompt tuning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 2704–2714.

[53] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6022–6031.

[54] V. Verma et al., "Manifold mixup: Better representations by interpolating hidden states," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6438–6447.

[55] H. Zheng, L. Zhou, H. Li, J. Su, X. Wei, and X. Xu, "BEM: Balanced and entropy-based mix for long-tailed semi-supervised learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 22893–22903.

[56] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 1597–1607.

[57] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9726–9735.

[58] J.-B. Grill et al., "Bootstrap your own latent: A new approach to self-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2020, pp. 18507–18519.

[59] X. Chen and K. He, "Exploring simple Siamese representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15745–15753.

[60] T. Brooks, A. Holynski, and A. A. Efros, "InstructPix2Pix: Learning to follow image editing instructions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 18392–18402.

[61] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2921–2929.

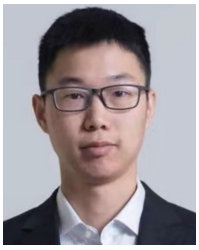
[62] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

[63] Q. Zhu, Y. Zhong, B. Zhao, G.-S. Xia, and L. Zhang, "Bag-of-visual-words scene classifier with local and global features for high spatial resolution remote sensing imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 6, pp. 747–751, Jun. 2016.

[64] W. Zhou, S. Newsam, C. Li, and Z. Shao, "PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 197–209, Nov. 2018.
[65] H. Li et al., "RSI-CB: A large-scale remote sensing image classification benchmark using crowdsourced data," *Sensors*, vol. 20, no. 6, p. 1594, Mar. 2020.
[66] S. Zhang, C. Chen, X. Hu, and S. Peng, "Balanced knowledge distillation for long-tailed learning," *Neurocomputing*, vol. 527, pp. 36–46, Mar. 2023.
[67] J. Zhu, Z. Wang, J. Chen, Y.-P.-P. Chen, and Y.-G. Jiang, "Balanced contrastive learning for long-tailed visual recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 6898–6907.
[68] X. Chen et al., "AREA: Adaptive reweighting via effective area for long-tailed classification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 19220–19230.
[69] J.-X. Shi, W. Tong, Y. Xiang, and Y.-F. Li, "How re-sampling helps for long-tail learning?" in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2023, pp. 75669–75687.
[70] F. Du, P. Yang, Q. Jia, F. Nan, X. Chen, and Y. Yang, "Global and local mixture consistency cumulative learning for long-tailed visual recognitions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 15814–15823.
[71] Q. Zhao, Y. Dai, H. Li, W. Hu, F. Zhang, and J. Liu, "LTGC: Long-tail recognition via leveraging LLMs-driven generated content," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 19510–19520.

Qianqian Wang received the B.S. degree in computer science and technology from Hefei University of Technology, Hefei, China in 2022. She is currently pursuing the Ph.D. degree with the MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, Nanjing University of Aeronautics and Astronautics, Nanjing, China.

Her research interests include computer vision and long-tailed remote sensing image classification.



Haibo Ye received the Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 2016.

He is currently an Associate Professor with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing. His research interests include computer vision, recommender systems, and machine learning for the Internet of Things.

Dong Liang received the B.S. degree in telecommunication engineering and the M.S. degree in circuits and systems from Lanzhou University, Lanzhou, China, in 2008 and 2011, respectively, and the Ph.D. degree from the Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan, in 2015.

He is currently an Associate Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. He has published several journal articles, including IEEE TRANSACTIONS ON IMAGE PROCESSING, *Pattern Recognition*, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. His research interests include model communication in pattern recognition and image processing.

Sheng-Jun Huang received the B.Sc. and Ph.D. degrees in computer science from Nanjing University, Nanjing, China, in 2008 and 2014, respectively.

He is currently a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing. His research interests include machine learning and data mining.

Dr. Huang was selected to the Young Elite Scientists Sponsorship Program by CAST in 2016 and won the China Computer Federation Outstanding Doctoral Dissertation Award in 2015, the KDD Best Poster Award in 2012, and the Microsoft Fellowship Award in 2011. He is a Junior Associate Editor of the *Frontiers of Computer Science*.