RSG: A Simple but Effective Module for Learning Imbalanced Datasets

Jianfeng Wang[1], Thomas Lukasiewicz[1], Xiaolin Hu[2], Jianfei Cai[3], Zhenghua Xu[4]*
University of Oxford[1] Tsinghua University[2] Monash University[3] Hebei University of Technology[4]
{jianfeng.wang,thomas.lukasiewicz}@cs.ox.ac.uk, xlhu@tsinghua.edu.cn, jianfei.cai@monash.edu
zhenghua.xu@hebut.edu.cn

Abstract

Imbalanced datasets widely exist in practice and are a great challenge for training deep neural models with a good generalization on infrequent classes. In this work, we propose a new rare-class sample generator (RSG) to solve this problem. RSG aims to generate some new samples for rare classes during training, and it has in particular the following advantages: (1) it is convenient to use and highly versatile, because it can be easily integrated into any kind of convolutional neural network, and it works well when combined with different loss functions, and (2) it is only used during the training phase, and therefore, no additional burden is imposed on deep neural networks during the testing phase. In extensive experimental evaluations, we verify the effectiveness of RSG. Furthermore, by leveraging RSG, we obtain competitive results on Imbalanced CIFAR and new state-of-the-art results on Places-LT, ImageNet-LT, and iNaturalist 2018. The source code is available at https://github.com/Jianf-Wang/RSG.

1. Introduction

Computer vision research has made great progress in the past few years, driven by the development of deep convolutional neural networks (CNNs) [20, 27, 11, 15, 35, 33, 32, 5, 12, 28] as well as large-scale datasets of high quality [7, 22]. However, these large-scale datasets are usually well-designed, and the number of instances in each class is balanced artificially, which is inconsistent with the data distributions in real-world scenarios. It is common that the images of some categories are difficult to be collected, resulting in a dataset with an imbalanced data distribution. In general, imbalanced datasets can be classified into two categories in terms of data distributions: long-tailed imbalanced distributions [6] and step imbalanced distributions [2], which will both be the focus of this work.

Generating new samples for rare classes during training is a good solution [8, 36, 38], which is regarded as a data augmentation method. However, these methods have different drawbacks, which limit their performance. Firstly, some frameworks [8, 38] were not trained in an end-to-end manner, so that the gradients cannot be backpropagated from the top to the bottom of CNNs. But it is well known that deep models can usually benefit from end-to-end training. Secondly, some methods [8, 38] utilized variation information, such as different poses or lighting, among samples from the same frequent class to generate new rare-class samples. However, these methods did not introduce any mechanism to ensure that the variation information obtained from frequent classes is class-irrelevant. As a result, if the variation information (which still contains the class-relevant information) is directly combined with real rare-class samples to generate new rare-class ones for training the classifier and reshaping decision boundaries, the performance will be hurt due to the aliasing of different class-relevant information. Finally, Wang *et al.* [36] use noise vectors to encode the variation information mentioned above. But using such noise vectors for generation can possibly generate unstable or low-quality samples, since noise vectors are too random to reflect the true variations among real images.[1]

*Corresponding author.

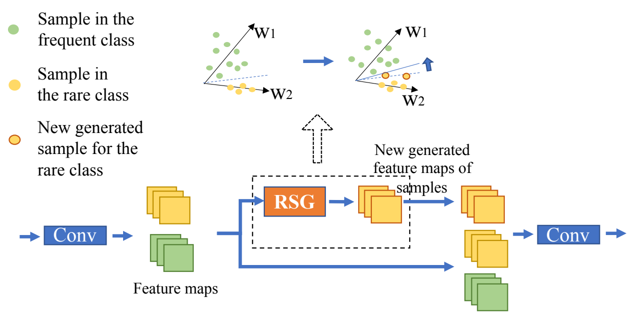
Sample in the frequent class
Sample in the rare class
New generated sample for the rare class
W1
W2
W1
W2
New generated feature maps of samples
RSG
Conv
Conv
Feature maps

Figure 1: RSG in a simple CNN. The part in the dotted box is only used during training. RSG learns to generate new rare-class samples, which are used to reshape the decision boundary and enlarge the feature space of rare classes.

[1]Note that [38] has proposed to avoid sampling random vectors due to their randomness, and [8] also has conducted experiments and verified that using random vectors to generate new samples for training classifiers can degrade the performance.

To alleviate the above drawbacks, in this paper, we propose a simple but efficient fully parameterized generator, called rare-class sample generator (RSG), which can be trained end-to-end with any backbone. RSG directly uses the variation information, which usually reflects different poses or lighting, among the real samples from the same frequent class to generate new samples rather than using random vectors to encode such information, and therefore, RSG can generate more reasonable and stable samples. Besides, RSG introduces a new module that is designed to further filter out the frequent-class-relevant information that possibly exists in the variation information, solving the aliasing problem mentioned above.

Figure 1 shows how it is integrated into a simple CNN for imbalanced datasets. RSG only requires the feature maps of samples from any specific layer, and it generates some new samples during training to impact on rare classes in order to adjust their decision boundaries and enlarging their feature space. In the testing phase, RSG is removed, so that no additional computational burden is imposed on the network. Note that we only show a simple CNN in Fig. 1, but RSG can be used in any network architecture, such as ResNet [11], DenseNet [15], ResNeXt [35], and Inception [27].

2. Related Work

Recent existing solutions for dealing with imbalanced datasets can be largely classified into approaches based on resampling and reweighting, new loss functions, meta-learning, utilizing unlabeled data, and sample generation.

Resampling techniques include oversampling the minority classes [25, 2, 3, 1, 19] and undersampling the majority classes [2, 18, 10], which aims to balance the data distribution. Reweighting methods [13, 14, 34, 6, 21] also try to balance the data distribution by assigning different weights to frequent-class and rare-class samples. Some approaches [39, 4] designed new loss functions by directly adding constraints to affect the decision boundaries for frequent and rare classes. Some meta-learning-based methods [24, 17, 16] were also proposed to solve the data imbalance problem. Very recently, Yang and Xu [37] analyzed the value of imbalanced labels, and utilized unlabeled data to boost class-imbalanced learning via semi-supervised and self-supervised strategies.

Previous sample generation methods are more relevant to this work than other approaches. A hallucinator [36] was designed to generate new samples for rare classes. It uses real instances from rare classes and noise vectors to produce new hallucinated instances for rare classes. A Δ-encoder framework [8] was proposed for generating new samples. It is first trained to reconstruct the pre-computed feature vector of input images from frequent classes. Thereafter, it is used to generate new samples by combining the real rare-class samples, and the newly generated ones are further used to train the classifier. A feature transfer learning (FTL) framework [38] was recently proposed, which consists of an auto-encoder, a feature filter, and fully-connected (FC) layers. The auto-encoder is initially pre-trained on a large-scale dataset for several epochs to converge to learn the latent representations. Then, principal component analysis (PCA) is leveraged to transfer the intra-class variance from frequent classes to rare classes by generating some new rare-class samples. A two-stage alternating training strategy was also proposed to jointly optimize the encoder, the feature filter, and FC layers.

3. Rare-Class Sample Generator (RSG)

The rare-class sample generator (RSG) is composed of a center estimation module, a contrastive module, and a vector transformation module (see Fig. 2). To optimize the parameters of RSG, two loss functions are used, namely, center estimation with sample contrastive (CESC) loss and maximized vector (MV) loss.

RSG assumes that samples from a class follow a unimodal distribution or a multi-modal distribution [17, 38], and thus there can be a center or a set of centers in each category to fit the distribution. In this paper, we define the notion of *feature displacement*, which indicates the displacement of a sample to its corresponding center in a class, caused by the same object with different conditions (e.g., angles, poses, or light conditions) in input images. Therefore, under ideal circumstances, feature displacement should not contain class-relevant information.

Given a mini-batch of samples consisting of both frequent-class and rare-class instances, RSG takes their feature maps as input and forwards them to these modules. The center estimation module aims to estimate a set of centers in each class, which is used as "anchors" for obtaining the feature displacement of each sample. The contrastive module is used to ensure that the feature displacement does not contain any frequent-class-relevant information during the sample generation process. The vector transformation module calculates the feature displacement of each frequent-class sample based on the estimated centers and uses it for generating new samples for rare classes. Intuitively, generating some new samples with such feature displacement that comes from abundant classes for rare classes may alleviate the problem caused by imbalanced datasets, as rare classes usually lack input variations.

The center estimation module is formulated as:

$$\gamma^l = f(A^l ave(x^l) + b^l), \tag{1}$$

where $x^l \in R^{D\times W\times H}$ is the feature map of an input sample, and we assume that the channel dimension, width, and height are D, W, and H, respectively. l is the class label of the sample, $ave(\cdot)$ denotes global average pooling across width

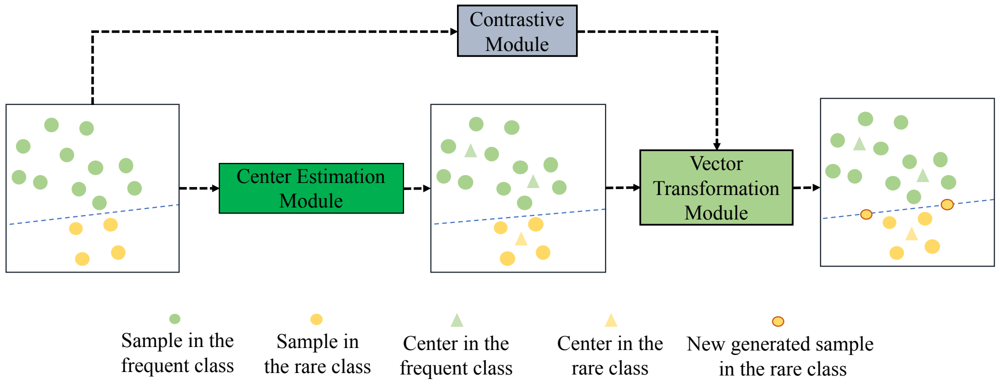
Contrastive Module
Center Estimation Module
Vector Transformation Module
Sample in the frequent class
Sample in the rare class
Center in the frequent class
Center in the rare class
New generated sample in the rare class

Figure 2: A diagram of RSG with samples' feature maps as input. The blue dashed line denotes a decision boundary.

and height, A^l and b^l are the parameters of this module performing a linear transformation on the input, and f is the softmax function that outputs a probability distribution (γ^l) for assigning the sample to the closest center in its corresponding class.

The center estimation module is designed to estimate a set of centers instead of only one center for each class, since the intra-class data distribution is unknown. If the intra-class data distribution is a multi-modal distribution, using a set of centers is better than using a single center. On the contrary, if it is a uni-modal distribution, those centers can be very close or overlapping, which is similar to using a single center.

The contrastive module is formulated as:

$$\gamma^* = f(A^* ave(h(cat[x_1, x_2])) + b^*), \tag{2}$$

where $x_1 \in R^{D\times W\times H}$ and $x_2 \in R^{D\times W\times H}$ are the feature maps of any two input samples from a given mini-batch, and $cat(\cdot)$ denotes the concatenation operation, which performs along the channel dimension. $h(\cdot)$ is implemented by stacking two 3×3 convolutional layers with 256 channels interleaved with a ReLU activation layer throughout the paper. A^* and b^* are the parameters of the linear layer, resulting in a probability distribution γ^* to show whether two samples come from the same class.

The vector transformation module is responsible for generating new rare-class samples through combining the feature displacement from real frequent-class samples with real rare-class samples. As Fig. 1 shows, an imbalanced dataset causes a bias in the decision boundary, resulting in a smaller feature space for rare classes than for frequent classes. Thus, we propose to use the vector transformation module to generate new samples for rare classes to enlarge the feature space and "push away" the decision boundaries.

To generate new samples, we first need to obtain the feature displacement from frequent classes, which is implemented by using the frequent-class samples and their corresponding centers estimated by the center estimation module:

$$x_{\text{fd-freq}} = x^l_{\text{freq}} - up(C^l_{\mathcal{K}}), \tag{3}$$

where $x^l_{\text{freq}} \in R^{D\times W\times H}$ denotes a sample in a frequent class l. We use $C^l_i \in R^D$ to denote the i-th center in class l with dimension D, and $\mathcal{K}$ is the index of the closest center to x^l_{freq}, i.e., $\mathcal{K} = \arg\max\ f(A^l ave(x^l_{\text{freq}}) + b^l)$. $up(\cdot)$ denotes the upsampling operation implemented by repeating the values of C^l_i along the width and height, forming feature maps of a center in the same size as the x^l_{freq}. After we subtract the corresponding center feature maps from x^l_{freq}, most of the class-relevant information is removed from x^l_{freq}; thus, we use $x_{\text{fd-freq}}$ to represent the feature displacement of the frequent-class sample.

Then, the second step is to generate new samples for rare classes by using $x_{\text{fd-freq}}$ and the real rare-class samples. Intuitively, $x_{\text{fd-freq}}$ can be added to the centers of rare classes, but we directly add $x_{\text{fd-freq}}$ to the real rare-class samples for two reasons: Firstly, the length of some $x_{\text{fd-freq}}$ may be smaller than the original variance of the feature space in rare classes. If we add $x_{\text{fd-freq}}$ to the centers, the new samples may have no impact on decision boundaries. Secondly, due to the limited sample size of rare classes, most rare-class samples can directly determine the decision boundaries, and adding $x_{\text{fd-freq}}$ to rare-class samples has a more straightforward impact on the decision boundaries.

So, the generation process of new rare-class samples is:

$$x^{l'}_{\text{new}} = \mathcal{T}(x_{\text{fd-freq}}) + x^{l'}_{\text{rare}}, \tag{4}$$

where $x^{l'}_{\text{rare}} \in R^{D\times W\times H}$ denotes a sample in a rare class l', $x^{l'}_{\text{new}}$ is a newly generated sample in that class, and $\mathcal{T}$ is a linear transformation defined as $\mathcal{T}(z) = conv(z)$, where $conv$ denotes a single convolutional layer containing a set of convolutional filters with the kernel size 3, the stride 1, and the padding size 1, whose number is the same as the number of channels of input feature maps.

The center estimation with sample contrastive loss ($L_{\textbf{CESC}}$) aims to update centers of each class and to optimize the contrastive module as well as the center estimation

Algorithm 1: Training Procedure of RSG

```
Input:
Batch size: s; feature maps of training data: $\{x^{(i)}\}_{i=1}^{s}$; epoch threshold: $T_{th}$; centers: $C$; training epochs: T;
transfer strength: $\beta \in (0,1]$; center estimation module: $CE_\theta$; contrastive module: $CM_\theta$;
vector transformation module: $VT_\theta$; weights of the backbone network: $\widetilde{\theta}$; frequent-class ratio: $\alpha \in (0,1]$.
Training:
for j in range(0, T):
  Compute $L_{\text{CESC}}$ with $\{x^{(i)}\}_{i=1}^{s}$. Compute gradient $\nabla_{\text{CESC}}$.
  Update: $\nabla_{\text{CESC}} \rightarrow CE_\theta$, $\nabla_{\text{CESC}} \rightarrow C$.
  if j <$T_{th}$:
    Compute $L_{\text{cls}}$ with $\{x^{(i)}\}_{i=1}^{s}$. Compute gradient $\nabla_{\text{cls}}$.
    Update: $\nabla_{\text{cls}} \rightarrow \widetilde{\theta}$, $\nabla_{\text{CESC}} \rightarrow CM_\theta$.
  else:
    Generate new samples with $\alpha$ and $\beta$: $\{x_{\text{new}}^{(i)}\}_{i=1}^{s_{\text{new}}}$. Concat: $\{x_{\text{aug}}^{(i)}\}_{i=1}^{s+s_{\text{new}}}$ = $[\{x^{(i)}\}_{i=1}^{s}, \{x_{\text{new}}^{(i)}\}_{i=1}^{s_{\text{new}}}]$.
    Compute $L_{\text{MV}}$ with $C$, $\{x_{\text{new}}^{(i)}\}_{i=1}^{s_{\text{new}}}$, and $\{x^{(i)}\}_{i=1}^{s}$. Compute gradient $\nabla_{\text{MV}}$.
    Compute $L_{\text{cls}}$ with $\{x_{\text{aug}}^{(i)}\}_{i=1}^{s+s_{\text{new}}}$. Compute gradient $\nabla_{\text{cls}}$.
    Update: $\nabla_{\text{MV}} + \nabla_{\text{cls}} \rightarrow VT_\theta$, $\nabla_{\text{cls}} \rightarrow \widetilde{\theta}$.
  end if
end for
```

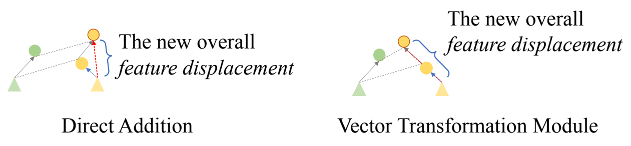

Figure 3: The objective and principle of the vector transformation module and MV loss. The triangles and circles in the figure have the same meaning as those in Fig. 2.

module. Therefore, it is composed of two classical loss terms, which can be written as:

$$L_{\text{CESC}} = \left\langle \sum_{i=0}^{K-1} \gamma_i^l \sum_{d,j,k} ||x^l_{(d,j,k)} - up(C_i^l)_{(d,j,k)}||^2 \right\rangle_s - \langle (ylog\gamma^* + (1-y)log(1-\gamma^*)) \rangle_{\frac{s}{2}}, \quad (5)$$

where d, j, and k denote the indices of the feature maps along the channel, width, and height. γ_i^l is the probability of the sample belonging to the i-th center obtained from Eq. (2), K is the number of centers in each class, s is the batch size. Considering a mini-batch with batch size s, $\frac{s}{2}$ sample pairs are formed by randomly picking samples from the mini-batch during training for the contrastive module. We denote by $y \in \{0,1\}$ the ground-truth showing whether the samples in each input pair come from the same class. $\langle \cdot \rangle_s$ and $\langle \cdot \rangle_{\frac{s}{2}}$ denote that the first term and the second term of L_{CESC} are calculated over s instances and $\frac{s}{2}$ pairs on average, respectively.

The maximized vector loss (L_{MV}) optimizes the parameter of the vector transformation module (namely, $\mathcal{T}$) and ensures that newly generated samples can enlarge the feature space of rare classes, where the basic idea is to maximize the feature displacement of newly generated samples relative to their centers (i.e., the new overall feature displacement in Fig. 3). Here, we treat the feature displacement of a sample as a vector starting from a center to the sample (see Fig. 3). To generate a new rare-class sample, one can directly add $x_{\text{fd-freq}}$ to a rare-class sample (Fig. 3, left), but the direction of $x_{\text{fd-freq}}$ is usually uncertain and the new overall feature displacement typically does not always have the largest length, because of the triangle inequality. Thus, we design the MV loss to make the transformed vector co-linear with the feature displacement of the rare-class sample in the same direction, and leave the length of the transformed vector unchanged (Fig. 3, right), to maximally impact the decision boundary. For example, if the direct addition is used, the newly generated samples may not impact the decision boundary due to the limited overall length. But leveraging the vector transformation module and MV loss ensures that the newly generated samples are widely distributed in the feature space of rare classes, because of the larger displacement relative to the centers, and it improves the probability that newly generated samples can appear around decision boundaries in each batch during training.

Moreover, as for a given frequent-class sample, although the frequent-class-relevant information has been largely removed when the feature displacement of a sample is calculated via Eq. (3), we still use the contrastive module to ensure that the feature displacement does not contain frequent-class-relevant information in order to further alleviate the possible class-relevant information aliasing problem when new rare-class samples are generated. W.l.o.g., γ^* is the probability that the two input samples of the contrastive module do not belong to the same category, and the MV loss is:

$$L_{\text{MV}} = \left\langle \sum_{j,k} (|\frac{\mathcal{T}(x_{\text{fd-freq}})^{(j,k)} \cdot x_{\text{fd-rare}}^{(j,k)}}{||\mathcal{T}(x_{\text{fd-freq}})^{(j,k)}||_2 ||x_{\text{fd-rare}}^{(j,k)}||_2} - 1|) \right\rangle_{s_{\text{new}}}$$
$$+ \left\langle \sum_{j,k} (| \, ||\mathcal{T}(x_{\text{fd-freq}})^{(j,k)}||_2 - ||x_{\text{fd-freq}}^{(j,k)}||_2|) \right\rangle_{s_{\text{new}}}$$
$$- \langle log\gamma^* \rangle_{s_{\text{new}}} , \tag{6}$$

where j and k denote the indices of the feature maps along the width and height, $|\cdot|$ takes the absolute value, and $x_{\text{fd-rare}}$ represents the feature displacement obtained from a sample and its closest center in a rare class via Eq. (3). The two input samples of the contrastive module are $\mathcal{T}(x_{\text{fd-freq}})$ and x^l_{freq}, respectively. The first term of L_{MV} is essentially to minimize the cosine angle of $\mathcal{T}(x_{\text{fd-freq}})$ and $x_{\text{fd-rare}}$ in order to make them co-linear in the same direction, the second term is to keep the length of $\mathcal{T}(x_{\text{fd-freq}})$ unchanged compared with $x_{\text{fd-freq}}$, and the third term makes $\mathcal{T}(x_{\text{fd-freq}})$ and x^l_{freq} not belong to the same category, ensuring that $\mathcal{T}(x_{\text{fd-freq}})$ will not have any frequent-class-relevant information. Given a mini-batch of samples, $\langle\cdot\rangle_{s_{\text{new}}}$ denotes that L_{MV} is calculated over newly generated samples on averages, where s_{new} is the number of newly generated samples.

Note that minimizing L_{MV} may encourage to generate some new samples with very large overall feature displacement, which can hurt the performance on frequent classes. Thus, the vector transformation module also receives the gradients from the classification loss function L_{cls}, reaching a trade-off between L_{cls} and the second term of L_{MV}, to generate more reasonable new samples for rare classes.

The training procedure and overall loss function of RSG are summarized in **Algorithm 1** and given as follows, respectively:

$$L_{total} = L_{\text{cls}} + \lambda_1 L_{\text{CESC}} + \lambda_2 L_{\text{MV}}, \tag{7}$$

where L_{cls} denotes any classification loss, such as softmax with cross-entropy loss, focal loss [23], AM-Softmax [30, 31], and LDAM [4], and λ_1 and λ_2 denote coefficients. The epoch threshold T_{th} is set to the index of epoch in which the learning rate is decayed to 0.001 in this paper.

The workflow of RSG is as follows (see Fig. 2). Before the epoch threshold T_{th}, given a mini-batch of samples, RSG splits them into two parts according to a manually set constant frequent-class ratio $\alpha = n_{\text{freq}}/n_{\text{cls}}$, where n_{freq} and n_{cls} denote the number of frequent classes and the total number of classes, respectively. For example, for a training set of 10 classes and $\alpha = 0.3$, the three classes with the largest number of samples are frequent classes, and the other classes are rare classes. (Note that for simplicity, only a frequent-class and a rare-class are plotted in Fig. 2.) Then, the data are forwarded to the center estimation module to update centers for each class and optimize the parameters of the center estimation module. In addition, those data are also forwarded to the contrastive module to optimize its parameters. After the epoch threshold T_{th}, RSG starts to generate new samples and the parameters of the contrastive module are not further updated. The feature displacement of each sample in frequent classes is calculated by the vector transformation module, which is then transformed with $\mathcal{T}$ and randomly added to the data in rare classes with a manually set parameter transfer strength β, resulting in newly generated samples. The contrastive module propagates gradients to the $\mathcal{T}$ in the vector transformation module to optimize $\mathcal{T}$ and filter out frequent-class-relevant information. In general, the number of samples in frequent classes is not smaller than that in rare classes in a given mini-batch. We define the transfer strength β as the number of samples in frequent classes involved in calculating the feature displacement and generating new samples for rare classes. Specifically, the number of newly generated samples is $s_{\text{new}} = max\{\lfloor\beta\times s_{\text{freq}}/s_{\text{rare}}\rfloor, 1\}\times s_{\text{rare}}$, where s_{freq} and s_{rare} are the numbers of samples in frequent and rare classes in a mini-batch, respectively, and $\lfloor\cdot\rfloor$ is the floor function. Finally, the feature maps of newly generated samples are concatenated with the original input feature maps along the batch dimension and forwarded to subsequent layers to calculate the loss and to optimize the whole framework.

4. Experimental Evaluation

Datasets. The experimental evaluation focuses on the Imbalanced CIFAR, the iNaturalist 2018, the Places-LT, and the ImageNet-LT datasets. Imbalanced CIFAR is based on the original CIFAR dataset, which is constructed by reducing the training samples per class, and the validation set is not changed. An imbalance ratio ρ is defined as the ratio between sample sizes of the most frequent class and the least frequent class, i.e., $\rho = N_{\text{max}}/N_{\text{min}}$. We conducted experiments on the long-tailed imbalance [6] and step imbalance [2] settings. The imbalance factors (ρ) that we used in our experiments are 50 and 100. The iNaturalist species classification dataset [29] is a large-scale imbalanced dataset of 437,513 training images classified into 8142 species in its 2018 version. The official training and validation set has a long-tailed distribution and a balanced distribution, respectively. Places-LT has 365 categories, with the maximum of 4980 images per class and the minimum of 5 images per class, while ImageNet-LT has 1000 categories, with the maximum of 1280 images per class and the minimum of 5 images per class. As for the evaluation on these two datasets, the classes are further categorized into three splits: many-shot (more than 100 samples), medium-shot (between 20 to 100), and few-shot (less than 20) in order to better examine performance variations across classes with different numbers of samples seen during training. We follow the experimental setting of these datasets in previous works [4, 19] for evaluation.

Implementation details. The training details on the four datasets are summarized as follows:

- **Imbalanced CIFAR:** We followed the basic data augmentation method [11] for training: 4 pixels are padded, and a 32×32 patch is randomly cropped from the image or its horizontal flip. The framework was trained with a batch size of 128 for 200 epochs. The learning rate was initially set to 0.1, and then it was decayed by 0.01 at the 160-th epoch and again at the 180-th epoch. The network was optimized by using stochastic gradient descend with a momentum of 0.9.

- **iNaturalist 2018:** We followed standard practice and performed data augmentation with random-size cropping [27] to 224×224 from images or their horizontal flip. The network was trained from scratch for 90 epochs with a batch size of 256. The learning rate was set to 0.1 initially, and then it was decayed by 0.1 at the 50-th epoch, the 70-th epoch, and the 85-th epoch, respectively. Besides, for a fair comparison, we followed Kang *et al.* [19] and also trained the model for the $2\times$ schedular (180 epochs). In our $2\times$ schedular experiment, the learning rate was decayed by 0.1 at the 100-th epoch, the 140-th epoch, and the 170-th epoch, respectively. During validation, images were center-cropped to 224×224 without further augmentation.

- **Places-LT:** We followed previous work [24] to perform the data augmentation and to fine-tune ResNet-152, which is pre-trained on the full ImageNet-2012 dataset. The network was trained with a batch size of 256 for 30 epochs. The initial learning rate was set to 0.01, and it was decayed by 0.1 at every 10 epoch, and the training was stopped after 30 epochs.

- **ImageNet-LT:** We followed previous work [19] to use ResNeXt-50-32x4d, which was trained with a batch size of 256 for 100 epochs. The initial learning rate was set to 0.1, and it was decayed by 0.1 at the 60-th epoch, the 80-th epoch, and the 95-th epoch, respectively.

Ablation studies. We performed ablation studies on Imbalanced CIFAR with $\rho = 50$. The mean error rates that are taken from three independent runs are reported. We comprehensively searched the hyperparameters of RSG and explored which level of feature is the most suitable for RSG to generate new samples by conducting experiments on ResNet-32 [11] with LDAM-DRW [4], where "DRW" denotes a deferred re-weighting training strategy proposed by Cao *et al.* [4]. Based on our exploration, in the following

CIFAR-10	Long-Tailed		Step	
	w/o RSG	w/ RSG	w/o RSG	w/ RSG
ERM	25.19	**20.25**	28.88	**26.07**
Focal Loss [23]	23.28	**21.58**	28.70	**26.01**
M-DRW [30, 4]	20.44	**17.72**	21.05	**20.09**
LDAM-DRW [4]	18.97	**17.20**	18.67	**17.90**
CIFAR-100	Long-Tailed		Step	
	w/o RSG	w/ RSG	w/o RSG	w/ RSG
ERM	56.15	**54.44**	59.32	**56.82**
Focal Loss [23]	55.68	**54.85**	58.50	**55.93**
M-DRW [30, 4]	56.06	**55.30**	56.26	**54.60**
LDAM-DRW [4]	53.38	**51.50**	50.97	**49.43**

Table 1: Top-1 error rates of ResNet-32 with RSG for different loss functions on Imbalanced CIFAR for $\rho = 50$.

CIFAR-10	Long-Tailed		Step	
	w/o RSG	w/ RSG	w/o RSG	w/ RSG
ResNet-32	18.97	**17.20**	18.67	**17.90**
ResNet-56	18.01	**16.83**	18.52	**17.20**
ResNet-110	17.70	**16.61**	17.96	**16.73**
DenseNet-40	17.46	**16.21**	17.40	**16.12**
ResNeXt-29, 8×64d	16.10	**15.26**	16.82	**15.99**
CIFAR-100	Long-Tailed		Step	
	w/o RSG	w/ RSG	w/o RSG	w/ RSG
ResNet-32	53.38	**51.50**	50.97	**49.43**
ResNet-56	51.63	**50.60**	49.22	**48.53**
ResNet-110	50.64	**49.83**	48.65	**47.90**
DenseNet-40	49.51	**48.75**	48.30	**47.13**
ResNeXt-29, 8×64d	49.62	**48.70**	50.68	**47.16**

Table 2: Top-1 error rates of different network architectures combined with LDAM-DRW [4] on Imbalanced CIFAR for $\rho = 50$.

experiments, we set the number of centers to 15, the frequent-class ratio to 0.2 and 0.5 for long-tailed and step imbalanced distributions, the transfer strength to 1.0 and 0.01 for long-tailed and step imbalanced distributions, and λ_1 and λ_2 to 0.1 and 0.01, respectively. The search process can be found in the supplementary material. Note that RSG was initially used before the second-to-last down-sampling layer.

Firstly, we fixed the network architecture to ResNet-32 [11] and tested RSG relative to different L_{cls}. By Table 1, the deep model equipped with RSG consistently performs better than the one without RSG when combined with different loss functions. RSG significantly improves the performance when the model is combined with standard softmax with cross-entropy loss (denoted ERM, i.e., empirical risk minimization). This is reasonable, as standard softmax does not have any mechanism against imbalanced datasets. As for focal loss, AM-softmax, and LDAM, although they are well-designed to tackle imbalanced datasets, RSG can still further improve the performance.

Secondly, we set L_{cls} to LDAM with DRW [4] (i.e., LDAM-DRW) and evaluated (mainly five) different network architectures combined with RSG on Imbalanced CIFAR, namely, ResNet-32, ResNet-56, ResNet-110, DenseNet-40,

	Long-Tailed		Step	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
MV Loss w/o 1st Term	18.03	52.15	18.58	50.34
MV Loss w/o 2nd Term	18.07	52.33	18.36	50.19
MV Loss w/o 3rd Term	17.67	52.12	18.23	49.84
Adding to Rare-class Centers	18.91	52.79	18.47	50.67
Direct Addition	18.87	52.48	18.33	49.99
Vector Transformation Module	**17.20**	**51.50**	**17.90**	**49.43**

Table 3: Ablation study on MV loss and the vector transformation module. Top-1 error rates of ResNet-32 combined with RSG and LDAM-DRW [4] on Imbalanced CIFAR for $\rho = 50$ are reported.

	Long-Tailed		Step	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
Δ-Encoder [8]	23.76	54.91	27.70	57.85
Imaginary [36]	23.99	55.08	28.23	58.46
FTL [38]	23.56	55.24	27.83	58.03
ERM-RSG (ours)	**20.25**	**54.44**	**26.07**	**56.82**

Table 4: Comparison with other sample generation methods on Imbalanced CIFAR ($\rho = 50$). All of them are based on ResNet-32 combined with ERM for a fair comparison.

and ResNeXt-29 (8×64d). Note that the used networks were built according to the experiments on CIFAR in their original papers [11, 35, 15]. As Table 2 shows, when RSG is integrated into the networks, all the models are consistently improved.

Thirdly, we did a comprehensive ablation study on MV loss and the vector transformation module, and we obtain the following conclusions based on Table 3: (1) Every subterm of MV loss is important and useful, since once we remove any subterm of it, an increase can be observed with regard to the error rate. (2) Adding the feature displacement to the centers of rare classes leads to an increase in terms of the error rate. This fact verifies what we have mentioned in Section 3, i.e., adding the feature displacement to real rare-class samples is a better choice than adding it to the centers of rare classes. (3) Using the vector transformation module with MV loss performs better than directly adding the feature displacement to the samples in rare classes, which thus verifies their effectiveness.

Moreover, RSG is compared with previous sample generation methods [8, 36, 38]. As Table 4 shows, RSG has outperformed previous methods with different margins, showing that RSG can solve the drawbacks in previous generation methods and improve the performance.

Finally, we leveraged RSG before different pooling layers of ResNet-32 to explore which level of feature is the most suitable for generating new samples. As Table 5 shows, RSG achieves the best result when it was used before the second-to-last down-sampling layer. Therefore, in the remaining experiments, RSG was still used before the second-to-last down-sampling layer.

Comparison with state of the art. For each of the following experiments, we report mean error rates or mean accuracies, which are taken from three independent runs. Table 6 shows the results on Imbalanced CIFAR with $\rho \in \{50, 100\}$. We first compare our LDAM-DRW-RSG with LDAM-DRW, as this comparison directly shows the improvement brought by RSG. After combining LDAM-DRW with RSG, we obtain a remarkable improvement for both long-tailed and step imbalanced distributions, which shows the power of RSG for handling imbalanced datasets. As a result, with the help of RSG, LDAM-DRW-RSG achieves superior results on Imbalanced CIFAR when compared with previous methods.

Table 7 shows the top-1 error rate of different methods using ResNet-50 [11] as the backbone on iNaturalist 2018, and we followed Kang *et al.* [19] to conduct experiments in two training settings, namely, the 1× schedular and the 2× schedular. In the 1× schedular experiment, we compare LDAM-DRW-RSG and LDAM-DRS-RSG with previous LDAM-DRW and LDAM-DRS, separately. Here, "DRS" denotes a deferred class-balanced resampling strategy proposed by Cao *et al.* [4]. Note that we cannot reproduce the result on iNaturalist 2018 reported in the original paper (32.0%) [4] by using LDAM-DRW. So, we report our reproduced results of LDAM-DRW and LDAM-DRS [4] based on their publicly available code. The results in Table 7 show that we can obtain better results by leveraging the proposed

	Long-Tailed		Step	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
1st down-sampling	18.13	53.22	18.66	50.81
2nd down-sampling	**17.20**	**51.50**	**17.90**	**49.43**
3rd down-sampling (GAP)	17.68	52.14	18.05	50.38

Table 5: Ablation study (top-1 error rates) with regard to the different layers, where RSG was used on Imbalanced CIFAR ($\rho = 50$). RSG was used before the three down-sampling layers in ResNet-32. ResNet-32 combined with LDAM-DRW was used, and GAP denotes global average pooling.

Training Schedular	Method	Error Rate
1× schedular	ERM	42.86
	CB Focal Loss [6]	38.88
	ERM-DRW [4]	36.27
	ERM-DRS [4]	36.44
	BBN [1]	33.71
	τ-normalized [19]	34.40
	LDAM-DRW [4]	34.00
	LDAM-DRS [4]	32.73
	LDAM-DRW-SSP [37]	33.70
	LDAM-DRW-RSG (ours)	33.22
	LDAM-DRS-RSG (ours)	**32.10**
2× schedular	BBN [1]	30.38
	τ-normalized [19]	30.70
	cRT [19]	32.40
	LWS [19]	30.50
	LDAM-DRS-RSG (ours)	**29.74**

Table 7: Top-1 error rates of ResNet-50 on iNaturalist 2018.

Dataset	Imbalanced CIFAR-10				Imbalanced CIFAR-100			
Imbalance Type	Long-Tailed		Step		Long-Tailed		Step	
Imbalance Ratio (ρ)	100	50	100	50	100	50	100	50
ERM	29.64	25.19	36.70	28.88	61.68	56.15	61.43	59.32
Focal loss [23]	29.62	23.28	36.09	28.70	61.59	55.68	61.65	58.50
CB Focal [6]	25.43	20.73	39.73	39.65	63.98	54.83	80.24	85.10
CB RW [6]	27.63	21.95	38.06	30.38	66.01	57.54	78.69	69.63
M-DRW [4]	24.94	20.44	27.67	21.05	59.49	56.06	58.91	56.26
BBN [1]	**20.18**	17.82	22.34	18.33	57.44	52.98	54.14	50.49
LDAM-DRW [4]	22.97	18.97	23.08	18.67	57.96	53.38	54.64	50.97
LDAM-DRW-SSP [37]	22.17	17.87	22.95	18.38	56.57	52.89	54.28	50.47
LDAM-DRW-RSG (ours)	20.45	**17.20**	**21.65**	**17.90**	**55.45**	**51.50**	**53.00**	**49.43**

Table 6: Top-1 error rates of ResNet-32 on Imbalanced CIFAR.

generator, which directly demonstrates the effectiveness of RSG. Moreover, as for the 2× schedular setting, the top-1 error rate of LDAM-DRS-RSG is further decreased. Thus, it can be seen that RSG helps the model achieve new state-of-the-art results in both training schedular settings, which demonstrates that RSG is capable of dealing with imbalanced datasets effectively.

Table 8 shows the top-1 accuracy on Places-LT. The results show that the performance can be further improved when RSG is combined with LDAM-DRS, showing that RSG is useful. Moreover, when compared with the recent two popular methods, namely, τ-normalized [19] and BBN [1], RSG can improve the performance of the model on medium-shot and few-shot classes with less accuracy loss on many-shot classes, resulting in a higher overall accuracy and a new state-of-the-art result.

Table 9 shows the top-1 accuracy on ImageNet-LT. When compared with LDAM-DRW, LDAM-DRW-RSG can achieve a higher accuracy, verifying that RSG is able to alleviate the problem caused by imbalanced datasets. RSG can enhance the model and greatly improve its generality on medium-shot and few-shot classes. In addition, by equipping RSG, we can also obtain a new state-of-the-art result on ImageNet-LT.

Since all hyperparameters of RSG were fixed after the hyperparameter searching process, we can conclude that the hyperparameters and RSG are quite robust to new datasets (i.e., Places-LT, ImageNet-LT, and iNaturalist 2018). If hyperparameters are further tuned on the new datasets, even better results might be obtained.

Method	Many	Medium	Few	All
Lifted Loss [26]	41.1	35.4	24.0	35.2
Focal Loss [23]	41.1	34.8	22.4	34.6
Range Loss [39]	41.1	35.4	23.2	35.1
FSLwF [9]	43.9	29.9	29.5	34.9
BBN [1]	42.5	40.3	30.6	38.7
OLTR [24]	**44.7**	37.0	25.3	35.9
τ-normalized [19]	37.8	40.7	31.8	37.9
LDAM-DRS [4]	43.3	38.3	30.7	38.6
LDAM-DRS-RSG (ours)	41.9	**41.4**	**32.0**	**39.3**

Table 8: Top-1 accuracy of ResNet-152 on Places-LT.

Method	Many	Medium	Few	All
Focal Loss [23]	63.3	37.4	7.7	43.2
OLTR [24]	52.1	39.7	20.3	41.2
Joint [19]	**65.9**	37.5	7.7	44.4
NCM [19]	56.6	45.3	28.1	47.3
cRT [19]	61.8	46.2	27.4	49.6
τ-normalized [19]	59.1	46.9	30.7	49.4
LWS [19]	60.2	47.2	30.3	49.9
LDAM-DRS [4]	63.7	47.6	30.0	51.4
LDAM-DRS-RSG (ours)	63.2	**48.2**	**32.3**	**51.8**

Table 9: Top-1 accuracy of ResNeXt-50 on ImageNet-LT.

5. Summary and Outlook

We have introduced a rare-class sample generator (RSG), which is a general building block to mitigate the issue of training on imbalanced datasets. RSG is simple yet effective, since it is an architecture-agnostic and loss-agnostic plug-in module, and it does not bring any additional burdens to the backbone network during the inference phase. In extensive experiments, we have verified the effectiveness of RSG, which has achieved excellent results on four public benchmarks. Since RSG is flexible and orthogonal to most previous methods, future research can focus on improving the RSG module directly by designing more elegant ways to generate higher-quality rare-class samples.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under the grant 61906063, by the Natural Science Foundation of Tianjin City, China, under the grant 19JCQNJC00400, and by the "100 Talents Plan" of Hebei Province, China, under the grant E2019050017. This work was also supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1 and by the AXA Research Fund. We also acknowledge the use of the Tier 2 facility JADE (EP/P020275/1) and GPU computing support by Scan Computers International Ltd.

References

[1] B. Zhou, C. Quan, W. Wei, and Z. Chen. BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Computer Vision and Pattern Recognition*, pages 1–8, 2020. 2, 7, 8

[2] M. Buda, A. Maki, and M. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018. 1, 2, 5

[3] J. Byrd and Z. Lipton. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pages 872–881, 2019. 2

[4] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in Neural Information Processing Systems*, 2019. 2, 5, 6, 7, 8

[5] S. Chen, J. Wang, Y. Chen, Z. Shi, X. Geng, and Y Rui. Label distribution learning on auxiliary label space graphs for facial expression recognition. In *Computer Vision and Pattern Recognition*, 2020. 1

[6] Y. Cui, M. Jia, T. Lin, Y. Song, and S. Belongie. Class-balanced loss based on effective number of samples. In *Computer Vision and Pattern Recognition*, pages 9268–9277, 2019. 1, 2, 5, 7, 8

[7] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1

[8] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, A. Kumar, R. Feris, R. Giryes, A. Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *Advances in Neural Information Processing Systems*, pages 2845–2855, 2018. 1, 2, 7

[9] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *Computer Vision and Pattern Recognition*, pages 4367–4375, 2018. 8

[10] H. He and E. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 2, 6, 7

[12] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Computer Vision and Pattern Recognition*, pages 7132–7141, 2018. 1

[13] C. Huang, Y. Li, C. Chen, and X. Tang. Learning deep representation for imbalanced classification. In *Computer Vision and Pattern Recognition*, pages 5375–5384, 2016. 2

[14] C. Huang, Y. Li, C. Chen, and X. Tang. Deep imbalanced learning for face recognition and attribute prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2

[15] G. Huang, Z. Liu, L. Maaten, and K. Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition*, pages 4700–4708, 2017. 1, 2, 7

[16] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng. Meta-Weight-Net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1917–1928, 2019. 2

[17] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. 2

[18] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002. 2

[19] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis. Decoupling representation and classifier for long-tailed recognition. *Internation Conference on Learning Representations*, 2020. 2, 6, 7, 8

[20] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 1

[21] B. Li, Y. Liu, and X. Wang. Gradient harmonized single-stage detector. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 8577–8584, 2019. 2

[22] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014. 1

[23] Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *International Conference on Computer Vision*, pages 2980–2988, 2017. 5, 6, 8

[24] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. Yu. Large-scale long-tailed recognition in an open world. In *Computer Vision and Pattern Recognition*, pages 2537–2546, 2019. 2, 6, 8

[25] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *European Conference on Computer Vision*, pages 467–482, 2016. 2

[26] H. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 8

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, pages 1–9, 2015. 1, 2, 6

[28] M. Tan and Q. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, 2019. 1

[29] G. Van, O. Mac, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The iNaturalist species classification and detection dataset. In *Computer Vision and Pattern Recognition*, pages 8769–8778, 2018. 5

[30] F. Wang, J. Cheng, W. Liu, and H. Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018. 5, 6

[31] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. CosFace: Large margin cosine loss for deep face recognition. In *Computer Vision and Pattern Recognition*, pages 5265–5274, 2018. 5

[32] J. Wang and X. Hu. Gated recurrent convolution neural network for OCR. In *Advances in Neural Information Processing Systems*, pages 334–343, 2017. 1

[33] J. Wang and X. Hu. Convolutional neural networks with gated recurrent connections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1

[34] Y. Wang, D. Ramanan, and M. Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pages 7029–7039, 2017. 2

[35] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition*, pages 1492–1500, 2017. 1, 2, 7

[36] Y. Wang, R. Girshick, M. Hebert, and B. Harihara. Low-shot learning from imaginary data. In *Computer Vision and Pattern Recognition*, pages 7278–7286, 2018. 1, 2, 7

[37] Y. Yang, and Z. Xu. Rethinking the value of labels for improving class-imbalanced learning. In *Advances in Neural Information Processing Systems*, 2020. 2, 7, 8

[38] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker. Feature transfer learning for deep face recognition with under-represented data. *Computer Vision and Pattern Recognition*, 2019. 1, 2, 7

[39] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao. Range loss for deep face recognition with long-tailed training data. In *International Conference on Computer Vision*, pages 5409–5418, 2017. 2, 8