FCC: Feature Clusters Compression for Long-Tailed Visual Recognition

Jian Li[1], Ziyao Meng[2], Daqian Shi[3], Rui Song[1], Xiaolei Diao[3], Jingwen Wang[1], Hao Xu[1,*]
[1]Jilin University, [2]University of Minho, [3]University of Trento
lijianjlu@126.com, id9272@alunos.uminho.pt, daqian.shi@unitn.it, songrui20@mails.jlu.edu.cn,
xiaolei.diao@unitn.it, wangjingwen_jlu@163.com, xuhao@jlu.edu.cn

Abstract

Deep Neural Networks (DNNs) are rather restrictive in long-tailed data, since they commonly exhibit an under-representation for minority classes. Various remedies have been proposed to tackle this problem from different perspectives, but they ignore the impact of the density of Backbone Features (BFs) on this issue. Through representation learning, DNNs can map BFs into dense clusters in feature space, while the features of minority classes often show sparse clusters. In practical applications, these features are discretely mapped or even cross the decision boundary resulting in misclassification. Inspired by this observation, we propose a simple and generic method, namely Feature Clusters Compression (FCC), to increase the density of BFs by compressing backbone feature clusters. The proposed FCC can be easily achieved by only multiplying original BFs by a scaling factor in training phase, which establishes a linear compression relationship between the original and multiplied features, and forces DNNs to map the former into denser clusters. In test phase, we directly feed original features without multiplying the factor to the classifier, such that BFs of test samples are mapped closer together and do not easily cross the decision boundary. Meanwhile, FCC can be friendly combined with existing long-tailed methods and further boost them. We apply FCC to numerous state-of-the-art methods and evaluate them on widely used long-tailed benchmark datasets. Extensive experiments fully verify the effectiveness and generality of our method. Code is available at https://github.com/lijian16/FCC.

1. Introduction

Recently, Deep Neural Networks (DNNs) have achieved considerable success in a variety of visual tasks, such as object detection [5,12] and visual recognition [18,38]. Despite with ingenious networks and powerful learning capabilities, the great success is inseparable from large-scale balanced datasets, such as ImageNet [8]. However, datasets collected from real-world scenarios normally follow imbalanced and long-tailed distributions, in which few categories (majority classes) occupy most of the data while many categories (minority classes) are under-represented [15]. DNNs trained on such datasets commonly exhibit a bias towards over-represented majority classes and produce low recognition accuracy for minority classes [19, 24, 33].

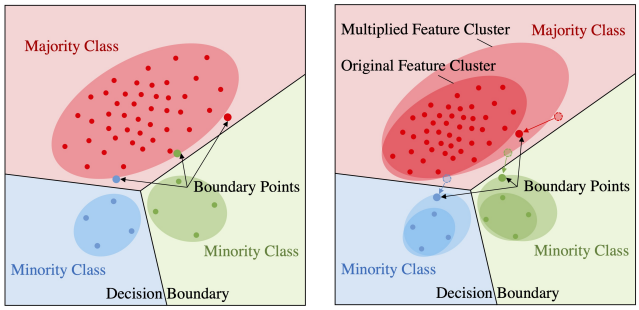


Figure 1. (a) DNNs can map backbone features into different clusters, while minority classes are mapped into sparse clusters compared to majority classes. The sparsity causes boundary points mapped far from their clusters or even cross the boundary. (b) FCC can compress original features compared with multiplied features, which makes these features are mapped closer together. Because the decision boundary remains unchanged in test phase, boundary points will be brought back within the boundary.

A number of remedies have been proposed to deal with this problem from different perspectives. For instance, re-sampling methods aim to balance the data distribution by designing different sampling strategies [13, 23]. Re-weighting methods attempt to alleviate the dominance of majority classes by assigning different weights to each class [1, 3, 21]. Two-stage training methods divide the vanilla training procedure into imbalanced learning and balanced fine-tuning [11, 19]. More recently, multi-expert networks have been proposed to employ multiple models to learn representation from different aspects [32, 37]. These methods show the outstanding performance on long-tailed recognition, but they ignore the impact of the density of Backbone

*Corresponding author

Features (BFs) on this issue.

DNNs primarily consist of the backbone networks and classifiers [10, 18]. The former, similar to unsupervised clustering algorithms [9, 20], can map BFs into different clusters by linear and non-linear operations (*e.g.*, convolution kernel and activation functions). The latter is similar to SVM [2], which can draw the decision boundary based on these clusters for recognition [7, 14]. DNNs trained on long-tailed datasets tend to map BFs of majority classes into dense clusters, but those of minority classes are mapped into sparse clusters, in which feature points are far from each other [41, 42]. In practical application, due to the sparsity, BFs of real samples cannot be mapped close enough to each other and are scattered in feature space, such that these feature points, especially boundary points that located at the margin of clusters, easily cross the decision boundary resulting in poor performance, as shown in Fig. 1a.

In view of this observation, we propose a Feature Clusters Compression (FCC) to increase the density of BFs by compressing backbone feature clusters. Our method can be easily achieved by only multiplying original BFs by a specific scaling factor τ ($\tau > 1$) and feeding the multiplied features to the classifier for training, which will establish a linear compression relationship among the original and multiplied features. As the multiplied features are mapped into clusters in training process, this relationship will force the backbone to map the original features into denser clusters, thereby improving the density. In test phase, we directly feed the original BFs without multiplying the factor τ to the trained classifier, these features are mapped closer together, and because the decision boundary remains consistent with training phase, boundary points will be brought back within the decision boundary resulting in performance improvement, as shown in Fig. 1b. We notice that the input of the classifier is different in training and test phases, and this problem will be discussed in Sec. 3.2.

As reported by [38], different long-tailed methods might hurt each other when they are employed inappropriately. For instance, applying re-sampling and re-weighting methods simultaneously might obtain similar or even worse accuracy than using them alone since they both try to enlarge the influence of minority classes. But our FCC is a generic method which only focuses on optimizing BFs without conflicting with other modules (*e.g.*, sampling strategies and loss functions), and it can be friendly combined with existing long-tailed methods and further boost them. Our contributions can be summarized as follows:

- We tackle long-tailed visual recognition from a novel perspective of increasing the density of BFs, which makes features mapped into denser clusters and boundary points brought back within the decision boundary.
- We propose a Feature Clusters Compression (FCC) to improve the density of BFs, and it can be easily achieved and friendly combined with existing long-tailed methods to boost them.
- Extensive experiments demonstrate that FCC applied to existing methods achieves significant performance improvement and state-of-the-art results on four popular datasets, including CIFAR-10-LT, CIFAR-100-LT, ImageNet-LT and iNaturalist 2018.

2. Related Work

Various approaches are proposed to tackle long-tailed recognition, which can be roughly grouped into four categories, including re-weighting, re-sampling, two-stage training and multi-expert methods. In this section, we introduce these methods that are recently published and representative of different types.

Re-weighting methods. Re-weighting methods commonly attempt to assign different weights to different classes to alleviate the dominance of majority classes. For instance, focal loss [21] down-weights the loss of well-classified samples by adding a weighting factor to the standard Cross Entropy (CE) loss function. Class Balanced (CB) focal loss and Class Balanced Cross Entropy (CBCE) loss [7] introduce the effective numbers of different classes in their method. Balanced Softmax Cross Entropy (BSCE) loss [27] is proposed to adapt the label distribution shift between training and testing. Besides, several recent approaches are also introduced since they consider the long-tailed distribution when calculating the loss. Cross Entropy Label Smooth (CELS) [28] proposes a strategy to regularize the classifier layer by estimating the marginalized effect of label-dropout. Cross Entropy Label Aware Smooth (CELAS) [39] aims to tackle different degrees of over-confidence for classes. The Label Distribution Aware Margin (LDAM) is proposed to minimize the margin-based generalization bound [3]. Class Dependent Temperatures (CDT) [35] simulates feature deviation in training phase to enlarge the decision values for minority classes. These methods show good results on long-tailed data but they might impair the accuracy of head classes.

Re-sampling methods. Re-sampling is a classic and widely used method for imbalanced learning by balancing the data distribution using different sampling strategies. Over-sampling [4] and under-sampling [23] methods are well known for balancing data but they have notable weaknesses. The former commonly causes models to over-fit to the duplicated samples, while the latter easily deteriorates the overall performance when the data is especially insufficient. Recently, more ingenious methods have been proposed. Class Balance (CB) sampling [13] tries to make each class has an equal probability of being selected. Square Root (SR) sampling [13] uses the square root of the number of samples to calculate the sampling probability. Pro-

gressively Balanced (PB) [13] changes the sampling probability of each class in training phase. And classifier Re-Training (cRT) [13] proposes to re-train the classifier with class-balanced sampling. However, it is difficult for them to overcome the fundamental deficiency in data.

Two-stage training. Two-stage training methods commonly divide training process into deferred re-balancing by re-sampling (DRS) and by re-weighting (DRW) [3]. For example, Kang *et al*. [13] decouple the training procedure into representation learning and classifier balancing. Li *et al*. [19] propose a two-stage training strategy depended on basis of knowledge distillation. DiVE [11] is also introduced since it trains a teacher model firstly and then use it to distill a student model. However, these methods often suffer from high sensitivity to hyper-parameters.

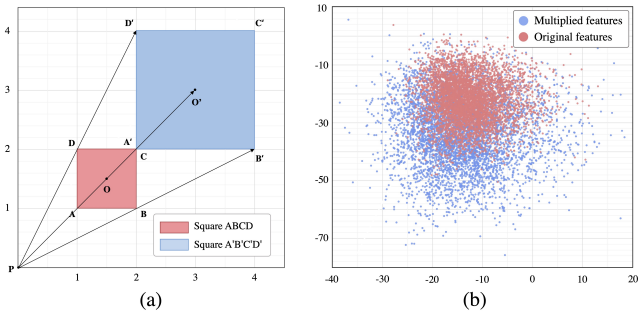
Multi-expert Networks. Recently, multi-expert networks have received more and more concern due to their outstanding performance on long-tailed data. For example, the unified Bilateral-Branch Network (BBN) [40] is designed to take care of both representation learning and classifier learning simultaneously. Wang *et al*. [32] propose a novel classifier called Routing Diverse Experts (RIDE) which can reduce the model variance with multiple experts and reduce the model bias with a distribution-aware diversity loss. Self-Supervised Aggregating Diverse Experts (SADE) [37] trains diverse experts to handle different class distributions. Li *et al*. [18] use a Nested Collaborative Learning (NCL) for collaboratively learning multiple experts. Multi-expert methods exhibit significant advantages over methods from other families, while they commonly have high complexity in training and inference procedures. Aforementioned Methods perform well on long-tailed recognition, but they still have space for improvement. In this work, we propose a Feature Clusters Compression (FCC) which can be friendly combined with existing methods and effectively boost them.

3. Methodology

We firstly introduce the principle and implement process of the proposed FCC in Sec. 3.1, and then the feasibility of FCC (i.e., directly feeding original BFs to the classifier in test phase) is mathematically illustrated in Sec. 3.2.

3.1. Feature Clusters Compression

To begin with, we present a simple example to illustrate the core principle and process of FCC. Assume that there is a square $ABCD$ in two-dimensional space and it will be transformed to the square $A'B'C'D'$ when its vertex coordinates are multiplied by a scaling factor τ (*e.g.*, $\tau = 2$), as shown in Fig. 2a. We can easily observe the square $ABCD$ is compressed relative to the square $A'B'C'D'$, and the distance between points $\{A, B, C, D\}$ is shorten by τ times and the density is enlarged to τ^2 times. Similarly, expanding to N-dimensional space, the distance is also shorten by τ times and the density is enlarged to τ^N times.



(a) (b)

Figure 2. (a) The square $ABCD$ will be transformed to the square $A'B'C'D'$ when its vertex coordinates are multiplied by 2. (b) Visualization of the original and multiplied features of class 0. Experiment is conducted on CIFAR-10-LT-100, where FCC with γ of 0.5 is used to ResNet-32.

In our method, BFs are viewed as the points in N-dimensional space, where N denotes the number of parameters in BFs and each parameter represents a dimension. We multiply original BFs (similar to the square $ABCD$) of each class by a specific scaling factor τ ($\tau > 1$) and further feed the multiplied features (similar to the square $A'B'C'D'$) to the classifier in each batch. This operation establishes a linear compression relationship (depending on τ) among the original and multiplied features, which forces the backbone to consistently map the original features into denser clusters than the multiplied features. This operation is defined as follows:

$$f_M^i = f_O^i * \tau_i \tag{1}$$

where f_M^i and f_O^i denote the multiplied and original features of class i, respectively. For the scaling factor τ_i, we define three strategies for setting it to control compression degrees of each class, as followings:

Uniform compression. Set the same τ_i for all classes as:

$$\tau_i = 1 + \gamma \tag{2}$$

Equal difference compression. τ_i is reduced in sequence from majority to minority classes, as following:

$$\tau_i = 1 + \gamma * (1 - i/C) \tag{3}$$

Half compression. Equal difference compression is only used for top 50% or bottom 50% classes, otherwise τ_i is set to 1 for other classes, it can be formulated as following:

$$\tau_i = 1 + \gamma * (1 - i/C) * \varphi((-1)^\beta * (i - C/2)) \tag{4}$$

where $\gamma > 0$ is a scaling hyper-parameter, C is the number of classes, and $i \in [0, C)$ is the index of class. $\varphi(\cdot)$ is 1 when its parameter is negative, otherwise it is 0. And β is 0 when only compress top 50% classes, otherwise it is 1. We will evaluate the performance of these compression strategies in Sec. 4.3.

After training with FCC, the multiplied features will be mapped to different clusters in feature space, while the original features are mapped into denser clusters. In order to clearly show the results of compression, we use Principal Component Analysis (PCA) [29] to visualize these clusters, as shown in Fig. 2b. We can observe that the original feature clusters (marked in red) are actually compressed and their feature points are closer to each other. In test phase, we directly feed the original features to the classifier, such that these features can be mapped closer by the trained backbone and do not easily cross the decision boundary.

3.2. Feasibility Demonstration of FCC

It is noteworthy that we feed the multiplied features to the classifier in training, while the original features are fed in test phase. The input of the classifier is different in these two stages. It is indispensable to demonstrate that the classifier can also normally work on the original features. We present a Fully Connected (FC) network of binary classification to prove its feasibility and this can be easily extended to our method.

Setting: The FC network contains an input layer with 3 neurons, a hidden layer with 3 neurons $\{a_1, a_2, a_3\}$ and a output layer with 2 neurons $\{o_1, o_2\}$. $\{\tau x_1, \tau x_2, \tau x_3\}$ and $\{x_1, x_2, x_3\}$ are the multiplied and original features, respectively, and they both belong to class 1. The scaling factor of class 1 is τ ($\tau > 1$). $\{y_1, y_2\}$ and $\{y_1{}', y_2{}'\}$ are outputs of the multiplied and original features produced by the FC network, respectively. $\{w_{i1}, w_{i2}, w_{i3}\}$ and b_i are weights and bias of the neuron a_i ($i \in \{1, 2, 3\}$), respectively. $\{n_{j1}, n_{j2}, n_{j3}\}$ and z_j are weights and bias of the neuron o_j ($j \in \{1, 2\}$), respectively.

Target: If the FC network can normally work, the classification result of the original feature will be equal to that of the multiplied feature, i.e., $y_1{}' > y_2{}'$ when $y_1 > y_2$.

The outputs of the multiplied feature can be expressed as follows:

$$\begin{aligned} y_i = {} & n_{i1}(\tau w_{11}x_1 + \tau w_{12}x_2 + \tau w_{13}x_3) + n_{i1}b_1 + \\ & n_{i2}(\tau w_{21}x_1 + \tau w_{22}x_2 + \tau w_{23}x_3) + n_{i2}b_2 + \\ & n_{i3}(\tau w_{31}x_1 + \tau w_{32}x_2 + \tau w_{33}x_3) + n_{i3}b_3 + \\ & z_i \end{aligned} \tag{5}$$

where $i \in \{1, 2\}$, then we denote $(y_1 - y_2)$ as η, $(w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$ as X_1, $(w_{21}x_1 + w_{22}x_2 + w_{23}x_3)$ as X_2, $(w_{31}x_1 + w_{32}x_2 + w_{33}x_3)$ as X_3 and $(n_{11}b_1 + n_{12}b_2 + n_{13}b_3 + z_1) - (n_{21}b_1 + n_{22}b_2 + n_{23}b_3 + z_2)$ as B, further η is converted as follows:

$$\eta = \tau k_1 X_1 + \tau k_2 X_2 + \tau k_3 X_3 + B \tag{6}$$

where k_i is $(n_{1i} - n_{2i}), i \in \{1, 2, 3\}$. We can notice that η is a (decision) plane in geometric space when $\eta = 0$. Due to $y_1 > y_2$, $\eta > 0$ and Eq. (6) can be formulated as follows:

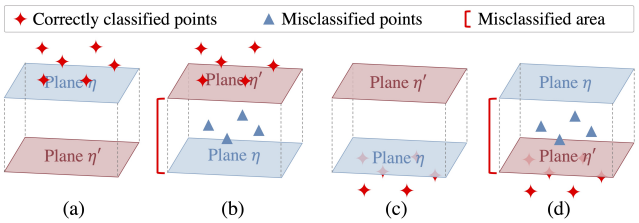


Figure 3. Relationship between planes η and η' and feature points in geometric space. When feature points are above plane η, (a) plane η' is below plane η, or (b) plane η' is above plane η. When feature points are below plane η, (c) plane η' is above plane η, or (d) plane η' is below plane η.

$$\begin{cases} \tau d_1 X_1 + \tau d_2 X_2 + \tau d_3 X_3 > 1, & B < 0 \\ \tau d_1 X_1 + \tau d_2 X_2 + \tau d_3 X_3 < 1, & B > 0 \end{cases} \tag{7}$$

where d_i denotes $-k_i/B, i \in \{1, 2, 3\}$. In geometric space, the point (X_1, X_2, X_3) is above the plane when $B < 0$, while it is below the plane when $B > 0$. The case of $B = 0$ will be discussed later. On the same principle, we make $y_1' - y_2'$ equal to η', which can be formulated as follows:

$$\eta' = k_1 X_1 + k_2 X_2 + k_3 X_3 + B \tag{8}$$

When $\eta' = 0$, Eq. (8) can be formulated as follows:

$$d_1 X_1 + d_2 X_2 + d_3 X_3 = 1 \tag{9}$$

where η' is also a plane when $\eta' = 0$. We can observe that $\{1/d_1, 1/d_2, 1/d_3\}$ and $\{1/\tau d_1, 1/\tau d_2, 1/\tau d_3\}$ are intercepts of planes η' and η, respectively. And the intercepts of plane η' are τ times of that of plane η, so planes η and η' are parallel in geometric space. Meanwhile, plane η' is either above plane η or below it depending on the intercepts.

Next, we will discuss the relationship between planes η and η' and feature points in geometric space to explore whether y_1' is also greater than y_2' under $y_1 > y_2$. (1) When $B < 0$, the point (X_1, X_2, X_3) is above plane η based on Eq. (7). If plane η' is below plane η, the point is also above plane η', as shown in Fig. 3a, so $d_1 X_1 + d_2 X_2 + d_3 X_3 > 1$ in Eq. (9), and then we can get $\eta' > 0$ (i.e., $y_1' > y_2'$) based on Eqs. (8) and (9). That implies the FC can normally work on this point. If plane η' is above plane η, the point might be above or below plane η', as shown in Fig. 3b. The point can also be correctly classified when it is above plane η' since $d_1 X_1 + d_2 X_2 + d_3 X_3 > 1$, but when it is below plane η', $d_1 X_1 + d_2 X_2 + d_3 X_3 < 1$ and $y_1' < y_2'$, which means the FC will misclassify the point. (2) On the same principle, when $B > 0$, the point is below plane η. $y_1' > y_2'$ holds when plane η' is above plane η as shown in Fig. 3c, while $y_1' < y_2'$ when plane η' is below plane η and the point is above plane η', as shown in Fig. 3d. (3) When $B = 0$, planes η and η' coincide with each other, which makes $y_1' > y_2'$ when

$y_1 > y_2$. To sum up, the classifier can normally work on original features, except those falling between planes η and η', i.e., "misclassified area" in Fig. 3.

According to Eqs. (7) and (9), we observe that it is a positive relationship between the size of "misclassified area" and τ, so τ cannot be set too large avoiding inoperation of the classifier. In fact, the "misclassified area" does not affect the overall performance in practical application when τ is set to an appropriate value, since few features will fall into this area. Detailed analysis is presented in Sec. 4.3.

Furthermore, we can also explain this issue from another intuitive perspective. Similar to the process in Fig. 2a, original feature clusters (square $ABCD$) will shift towards the origin point (point P in Fig. 2a) relative to multiplied feature clusters (square $A'B'C'D'$) when they are compressed, and the distance of shifting d can be denoted as:

$$d = D_i * (1 - 1/\tau_i) \tag{10}$$

where D_i is the distance from the center of the multiplied feature cluster of class i to the origin point (the feature point with all parameters of 0 in feature space). It is also a positive relationship between d and τ. The bigger τ is, the farther original feature clusters move, such that they might cross the decision boundary and be misclassified. In view of this, we define different compression strategies to set τ for each class to control the shifting distance. Especially, we assign smaller τ to minority classes, since these clusters are closer to the boundary and easier to cross it [14, 16]. We will specifically discuss it in Sec. 4.3.

4. Experiments and Results

This section presents extensive experiments on and analysis of FCC. The details of our experimental implementation are described in Sec. 4.1. Next, we demonstrate the effectiveness and generality of FCC using four long-tailed benchmarks in Sec. 4.2. An in-depth analysis of FCC is provided in Sec. 4.3 to study its inherent characteristics.

4.1. Experimental Setup

Datasets. In this work, we use four long-tailed benchmarks datasets, including CIFAR-10-LT, CIFAR-100-LT, ImageNet-LT and iNaturalist 2018. **CIFAR-10/100-LT** [7] are created by downsampling per-class training examples of the original CIFAR-10/100 datasets [17]. Specifically, by varying the Imbalanced Factor (IF) $\in \{50, 100\}$, which is defined as the number of training samples in the largest class divided by that of the smallest, we create four distinct long-tailed training sets. **ImageNet-LT** [22] consists of 115.8K images from 1,000 categories, created by artificially truncating the popular ImageNet dataset [8]. **iNaturalist 2018** [30] contains 435.7K images from 8,142 categories, which is the largest real-world long-tailed dataset that suffers from extremely imbalanced distribution.

Implementation. For CIFAR10/100-LT, following [3, 38], we utilize ResNet-32 [10] as the backbone network. All models are trained for 200 epochs with a batch size of 128. Learning rate is initialized to 0.1 and divided by 100 at 160^{th} and 180^{th} epoch, and warm-up is used for the first five epochs. For ImageNet-LT, we adopt ResNet-10 [10] as our backbone network and follow [18] to set training parameters. The number of training epochs is 200 and batch size is 64. Learning rate is initialized to 0.2 and divided by 10 at the 160^{th} and 180^{th} epoch without warm-up. For iNaturalist 2018, ResNet-32 [10] is also used as the backbone network and trained for 100 epochs with a batch size of 128. The initial learning rate is set to 0.2 and divided by 10 at the 60^{th} and 80^{th} epoch. The random seed is set to 42. All experiments are performed on 2 NVIDIA RTX3090 GPUs by using PyTorch toolbox [25]. **Top-1 error rate** is used to compare the experimental results.

For FCC, equal difference compression is used in all experiments, and γ is set to 0.5, 1, 0.1 and 0.1 for CIFAR-10-LT, CIFAR-100-LT, ImageNet-LT and iNaturalist 2018, respectively. FCC starts from the 50^{th} epoch except iNaturalist 2018, where it starts from the 0^{th} epoch. The impact of these hyper-parameters will be discussed in Sec. 4.3. If it is not specifically mentioned, the settings of experiments that appear in this paper refer to this part.

4.2. Benchmark Results

Compared Methods. To evaluate the effectiveness and generality of our proposed FCC, we extensively apply FCC to 31 state-of-the-art methods, which come from four families, i.e., re-weighting, re-sampling, two-stage training and multi-expert methods (described in Sec. 2). The chosen methods are recently published and representative of different families, such as focal loss [21] for re-weighting and NCL [18] for multi-expert. Especially, for two-stage training, we follow [38] to use chosen re-sampling and re-weighting methods for DRS and DRW, respectively. Meanwhile, we also introduce mixup training methods (i.e., Input Mixup [36], Manifold Mixup [31] and Remix [6]) into our experiments in terms of their good performance on long-tailed visual recognition, as reported by [38]. For comparison, we report the top-1 error rates of raw methods and those with FCC, respectively.

Results. For CIFAR-10/100-LT datasets, the results of baseline (vanilla ResNet-32), re-weighting, re-sampling, mixup, two-stage training and multi-expert methods are respectively presented in Tab. 1. In 98 experimental groups of Tab. 1, our proposed FCC significantly improves 94 of them (marked in red) by an average of 1.55% (4.89% max and 0.04% min). FCC achieves new state-of-the-art performance on all datasets, in which top-1 error rates on CIFAR-10-LT-50, CIFAR-10-LT-100, CIFAR-100-LT-50 and CIFAR-100-LT-100 are reduced to 12.72%, 14.20%,

Method	CIFAR-10-LT-50			CIFAR-10-LT-100			CIFAR-100-LT-50			CIFAR-100-LT-100		
	Raw	FCC	Incr	Raw	FCC	Incr	Raw	FCC	Incr	Raw	FCC	Incr
Baseline (Vanilla ResNet32) [10]	22.99%	19.78%	+3.21%	27.59%	24.08%	+3.51%	57.38%	54.83%	+2.55%	60.92%	58.93%	+1.99%
Focal loss (ICCV 2017) [21]	23.29%	20.49%	+2.80%	27.94%	26.23%	+1.71%	57.25%	55.24%	+2.01%	62.29%	58.63%	**+3.66%**
CB Focal loss (CVPR 2019) [7]	22.63%	21.37%	+1.26%	25.63%	25.37%	+0.26%	56.79%	54.84%	+1.95%	61.28%	59.42%	+1.86%
CBCE (CVPR 2019) [7]	21.48%	19.51%	+1.97%	27.50%	24.15%	+3.35%	56.58%	54.60%	+1.98%	61.56%	59.59%	+1.97%
BSCE (NeurIPS 2020) [27]	17.84%	16.85%	+0.99%	21.78%	20.87%	+0.91%	52.47%	52.61%*	-0.14%	58.55%	57.30%	+1.25%
CELS (CVPR 2016) [28]	22.70%	18.97%	+3.73%	27.49%	26.40%	+1.09%	56.96%	54.80%	+2.16%	61.93%	60.13%	+1.80%
CELAS (CVPR 2021) [39]	21.42%	19.17%	+2.25%	27.45%	24.53%	+2.92%	57.23%	55.34%	+1.89%	61.95%	60.78%	+1.17%
LDAM (NeurIPS 2019) [3]	21.47%	21.06%	+0.41%	26.58%	26.35%	+0.23%	56.94%	56.54%	+0.40%	61.26%	60.83%	+0.43%
CDT [35]	18.04%	17.12%	+0.92%	21.36%	20.32%	+1.04%	56.41%	56.37%	+0.04%	60.76%	60.84%*	-0.08%
CB sampling (ICLR 2020) [13]	22.31%	21.06%	+1.25%	27.02%	26.51%	+0.51%	60.67%	59.24%	+1.43%	66.47%	64.75%	+1.72%
SR sampling (ICLR 2020) [13]	20.89%	20.41%	+0.48%	28.03%	25.82%	+2.21%	57.94%	55.83%	+2.11%	63.26%	61.60%	+1.66%
PB sampling (ICLR 2020) [13]	21.11%	19.76%	+1.35%	25.16%	23.70%*	+1.46%	55.15%	53.33%	+1.82%	60.61%	58.98%	+1.63%
Input Mixup (ICLR 2018) [36]	21.39%	17.48%	**+3.91%**	25.84%	22.44%	+3.40%	54.48%	51.35%	+3.13%	59.14%	55.81%	+3.33%
Manifold Mixup (ICML 2019) [31]	21.24%	19.97%	+1.27%	23.58%	22.89%*	+0.69%	56.24%	51.35%	**+4.89%**	61.48%	60.35%	+1.13%
Remix (ECCV 2020) [6]	20.53%	17.00%	+3.53%	25.95%	22.03%	**+3.92%**	54.25%	51.36%	+2.89%	59.16%	56.23%	+2.93%
CB sampling+DRS	19.86%	18.4%	+1.46%	23.36%	21.91%	+1.45%	54.28%	52.93%	+1.35%	58.32%	57.00%	+1.32%
SR sampling+DRS	20.49%	19.16%	+1.33%	25.59%	24.09%	+1.50%	55.92%	54.11%	+1.81%	59.73%	57.29%	+2.44%
PB sampling+DRS	19.73%	18.44%	+1.29%	24.58%	22.70%	+1.88%	54.56%	53.19%	+1.37%	58.82%	57.29%	+1.53%
BSCE+DRW	18.79%	17.74%	+1.05%	21.88%	20.73%	+1.15%	53.68%	53.46%	+0.22%	57.63%	57.37%	+0.26%
CELAS+DRW	22.48%	19.19%	+3.29%	27.20%	23.97%	+3.23%	56.70%	55.01%	+1.69%	61.31%	59.93%	+1.38%
CDT+DRW	18.45%	17.81%	+0.64%	21.82%	20.83%	+0.99%	53.70%	53.32%*	+0.38%	57.76%	57.54%*	+0.22%
cRT (ICLR 2020) [13]	20.01%	19.62%	+0.39%	22.81%	22.36%	+0.45%	54.92%	55.02%	-0.10%	58.37%	58.17%	+0.20%
DiVE (ICCV 2021) [11]	17.34%	15.93%	+1.41%	21.32%	19.99%	+1.33%	50.19%	50.63%	-0.44%	55.84%	54.73%	+1.11%
LTR-WB +WD&Max (CVPR 2022) [1]	–	–	–	–	–	–	–	–	–	47.40%	46.50%*	+0.90%
SADE (NeurIPS 2022) [37]	–	–	–	–	–	–	–	–	–	51.02%	50.58%*	+0.44%
NCL (CVPR 2022) [18]	**12.92%**	**12.72%***	+0.20%	**14.50%**	**14.20%***	+0.30%	**41.67%**	**41.56%***	+0.11%	**46.14%**	**45.49%***	+0.65%

Table 1. Top-1 error rates comparisons between raw methods and those with FCC on **long-tailed CIFAR**. The results are presented in the order of baseline, re-weighting, re-sampling, mixup, two-stage training and multi-expert methods. $*$ denotes γ of 0.1 is used in FCC.

Method	ImageNet-LT			iNaturalist 2018		
	Raw	FCC	Incr	Raw	FCC	Incr
ResNet10/32 [10]	61.07%	60.60%	+0.47%	72.49%	71.99%	+0.50%
Focal loss [21]	63.10%	62.71%	+0.39%	–	–	–
CBCE [7]	60.92%	60.86%	+0.06%	69.85%	69.12%	+0.73%
LDAM-DRW [3]	63.53%	63.25%	+0.28%	**59.62%**	**59.54%**	+0.08%
BBN [40]†	51.80%	50.72%	+1.08%	–	–	–
cRT [13]	58.20%	56.59%	+1.61%	64.38%	63.86%	+0.52%
τ-norm [13]	66.10%	64.48%	+1.62%	76.39%	75.49%	+0.90%
DiVE [11]	56.93%	56.32%	+0.61%	–	–	–
RIDE [32]	55.72%	55.49%	+0.23%	–	–	–
SADE [37]⋆	**41.08%**	**39.47%**	+1.61%	–	–	–
NCL [18]†	47.32%	45.34%	**+1.98%**	63.46%	61.17%	**+2.29%**

Table 2. Top-1 error rates comparisons. † and ⋆ indicate the backbone is ResNet-50 and ResNeXt-50, respectively.

41.56% and 45.49%, respectively. Extensive experimental results fully verify the effectiveness of our method on long-tailed recognition. Meanwhile, these results demonstrate FCC is a generic method, which can be friendly combined with existing methods and then significantly boost them. Especially, mixup training methods are greatly improved (average 2.92%) by FCC compared with other families. For multi-expert methods, γ of 0.1 achieves better performance than γ of 0.5 and 1, we assume smaller γ is suitable for these complex networks because they are more sensitive to the shifting of feature clusters. We observe that FCC only fails to boost the performance in four experimental group (marked in green in Tab. 1). We speculate that this is because the decision boundary drawn by these methods is closer to feature clusters, such that these clusters will cross the boundary once they shift.

For ImageNet-LT, experimental results are listed in the left of Tab. 2. FCC consistently improves the performance of introduced methods by an average of 0.9% (1.98% max and 0.06% min). Especially, applying FCC to SADE [37] achieves a new state-of-the-art performance (top-1 error rate 39.47%). When applied to methods with simple tricks (*e.g.*, Focal loss and CBCE in Tab. 2), FCC obtains small improvement (less than 0.5%), while it achieves remarkable progress (over 1.5%) when employed to complex networks (*e.g.*, SADE [37] and NCL [18]). We assume this is because complex networks can create a better foundation for compression under the same compression factor. Moreover, the backbones of BBN [40], NCL [18] and SADE [37] are ResNet-50 [10] or ResNeXt-50 [34], which does not affect the conclusion of the effectiveness of our method, since their training settings are the same as those with FCC. For iNaturalist 2018, the results are shown in the right of Tab. 2. FCC reinforces the performance by an average of 0.84% (2.29% max and 0.08% min), which illustrates FCC can also effectively work on large-scale real-world datasets.

4.3. Analysis of FCC

Impact of the misclassified area. Original features might fall into the "misclassified area" when they are directly fed to the trained classifier (described in Sec. 3.2),

Parameter	CIFAR-10-LT-50			CIFAR-10-LT-100			CIFAR-100-LT-50			CIFAR-100-LT-100			ImageNet-LT		
	ResNet32	FCC	Incr	ResNet32	FCC	Incr	ResNet32	FCC	Incr	ResNet32	FCC	Incr	ResNet10	FCC	Incr
$\gamma = 0.1$	22.99%	23.21%	-0.22%	27.59%	28.74%	-1.15%	57.38%	57.15%	+0.23%	60.92%	61.51%	-0.59%	61.07%	**60.60%**	**+0.47%**
$\gamma = 0.5$	22.99%	**19.78%**	**+3.21%**	27.59%	**24.08%**	**+3.51%**	57.38%	56.97%	+0.41%	60.92%	59.64%	+1.28%	61.07%	60.79%	+0.28%
$\gamma = 1$	22.99%	20.46%	+2.53%	27.59%	25.45%	+2.14%	57.38%	**54.84%**	**+2.54%**	60.92%	**58.93%**	**+1.99%**	61.07%	61.25%	-0.18%
$\gamma = 2$	22.99%	23.93%	-0.94%	27.59%	27.31%	+0.28%	57.38%	55.27%	+2.11%	60.92%	59.22%	+1.70%	61.07%	61.73%	-0.66%
$\gamma = 3$	22.99%	20.78%	+2.21%	27.59%	30.11%	-2.52%	57.38%	56.96%	+0.42%	60.92%	59.31%	+1.61%	61.07%	63.88%	-2.81%

Table 3. Top-1 error rates comparisons between raw methods (ResNet-32 and ResNet-10) and those using **FCC with different γ** on long-tailed CIFAR and ImageNet-LT datasets.

Value	CIFAR-10-LT-100			CIFAR-100-LT-100		
	NMF	NOF	Ratio	NMF	NOF	Ratio
γ=0.1	11411	**3**	**0.02%**	10775	**0**	**0.00%**
γ=0.5	12356	77	0.62%	10734	112	1.04%
γ=1	12361	367	2.97%	10754	201	1.87%
γ=2	11575	2946	25.5%	10757	1653	15.4%
γ=3	11558	4566	39.5%	10720	3391	31.6%

Table 4. Analysis that original features fall into the "misclassified area". Experiments are conducted using ResNet-32 with FCC.

studying how many the features will fall into this area is a matter of great concern. We apply FCC with a series of $\gamma \in \{0.1, 0.5, 1, 2, 3\}$ to ResNet-32 on CIFAR10/100-LT-100 datasets. The Number of Multiplied Features (NMF) which are correctly classified by the classifier, the Number of Original Features (NOF) which fall into "misclassified area" and their ratio (NOF/NMF) are presented in Tab. 4. When setting γ to 0.1, original features barely fall into the area. γ of 0.5 and 1 also produce acceptable ratios (<3%). Referring to experimental results in Sec. 4.2, even if some features fall in the "misclassification area", the overall performance will not be affected. But γ cannot be set too large (*e.g.* 2 and 3), since the large value brings numerous original features to fall into the area.

Compression strategies. To explore the performance of different strategies (mentioned in Sec. 3.1) of FCC, we evaluate them on CIFAR-100-LT-100 dataset. The results, in Fig. 4a, display equal difference compression outperforms other strategies, especially, it obtains the best performance (accuracy 41.1%) with γ of 1. We visualize its recall results of each class, as shown in Fig. 4b, where it (marked in orange) achieves a remarkable improvement in minority classes compared with baseline (vanilla ResNet32 marked in blue). Half (top 50%) compression achieves the second-best result (accuracy 40.6%) with γ of 2, but it lacks improvement of minority classes. Uniform and half (bottom 50%) compression both deteriorate the overall performance compared with baseline. We assume this is because compression for all classes is necessary and compression degree for each class needs to be different. Based on this result, we utilize equal difference compression in our all experiments.

Hyper-parameter γ. In order to further search appropriate γ, we set a series of $\gamma \in \{0.1, 0.5, 1, 2, 3\}$ for FCC and employ it to ResNet-32 on long-tailed CIFAR and ImageNet

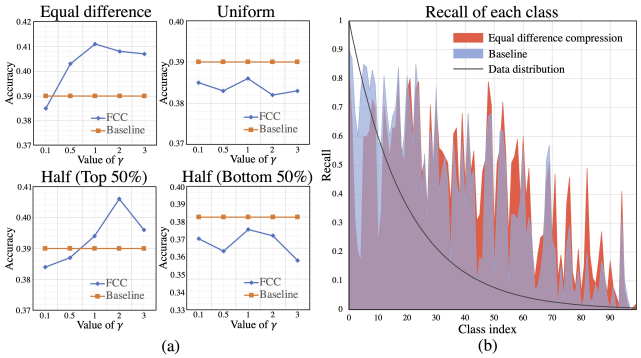


Figure 4. (a) Accuracy comparisons of each compression strategy. (b) Recall comparisons of each class between baseline (vanilla ResNet-32) and FCC with equal difference compression ($\gamma = 1$). Analysis is conducted on CIFAR-100-LT-100.

datasets, as shown in Tab. 3. For CIFAR-10-LT, the best performance is achieved when setting γ to 0.5 and FCC boosts the performance by over 3% compared with raw method. For CIFAR-100-LT, γ of 1 obtains the best result (around 2% improvement), while γ of 0.1 produces the best result (0.47% improvement) on ImageNet-LT. We observe that the optimal γ for different datasets is inconsistent, but γ of 0.1, 0.5 and 1 are generally the best choices. As the value of γ decreases, the performance improvement of FCC gradually decreases. This is because the smaller compression degree, the fewer boundary points brought back to the decision boundary. But excessive γ (*e.g.* $\gamma = 3$) will damage performance, since original feature clusters might cross the decision boundary. Based on these results, we respectively set γ to 0.5, 1, 0.1 and 0.1 for CIFAR-10-LT, CIFAR-100-LT, ImageNet-LT and iNaturalist 2018.

When to start FCC in training phase. Inspired by two-stage training [3], we study the impact of when to employ FCC during the training process on performance. We train ResNet-10/32 for 100/200 epochs on the long-tailed CIFAR datasets, in which FCC is used from the 0^{th}, 10^{th}, 20^{th}, 30^{th}, 40^{th}, 50^{th}, 60^{th} and 70^{th} epoch, respectively. The results are shown in Fig. 5, we use "relative accuracy" (the accuracy on the same dataset is subtracted by a specific constant) as the ordinate to clearly show the impact of the timing of using FCC on accuracy. For 200 training epochs, using FCC from the 50^{th} epoch generally yields the best

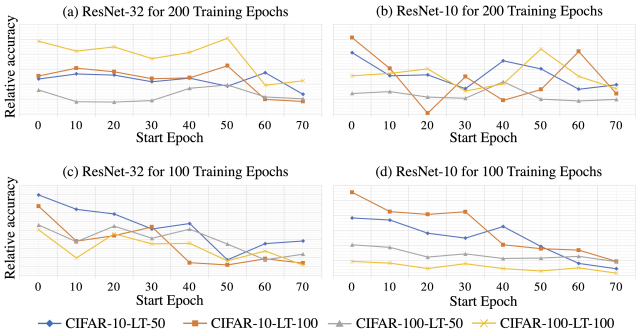


Figure 5. Start to employ FCC from different epoch. On long-tailed CIFAR datasets, we train (a) ResNet-32 and (b) ResNet-10 for 200 epochs, and (c) ResNet-32 and (d) ResNet-10 for 100 epochs, and start to use FCC from different epochs.

Method	CIFAR-10-LT		CIFAR-100-LT		ImageNet-LT
	IF-50	IF-100	IF-50	IF-100	
Baseline	6.6	4.9	5.4	4.8	216.0
Baseline+FCC	6.9	6.3	6.3	5.2	222.6

Table 5. Run-time comparison between baseline and that with FCC on five long-tailed datasets (in seconds). The baseline is ResNet-32 on CIFAR-10/100-LT and ResNet-10 on ImageNet-LT.

results. We speculate feature clusters are pulled away from each other after the initial phase of vanilla training. On this foundation, further employing FCC can make them farther apart and easier to recognize. However, for 100 training epochs, using FCC from the 0^{th} epoch achieves the best results, which implies sufficient epochs need to be reserved for FCC to compress features under fewer training epochs. Meanwhile, we observe that ResNet-10 with FCC from the 0^{th} epoch obtains better performance than that from the 50^{th} epoch on some datasets, as shown in Fig. 5b. We suspect that weaker networks require more epochs for FCC.

Impact of FCC on boundary points. To demonstrate FCC can bring boundary points back within the decision boundary, we present the results of FCC on three other imbalanced datasets (IF=10), which are created based on commonly used datasets from scikit-learn [26], including two circles, two blobs and two moons. In Fig. 6, the top row shows the results without FCC while the bottom row shows that with FCC, and the majority and minority classes are plotted in orange and blue, respectively. The results show that FCC can compress feature clusters and make boundary points of the minority class back within the boundary. In some cases, we observe that some feature points of the majority class move towards or cross the boundary (i.e., some orange dots are misclassified), but more feature points of the minority class are actually back inside the boundary, and extensive experimental results in Sec. 4.2 also demonstrate it does not damage the overall performance.

Run-time. FCC is a very simple method that only multiplies features by a specific compression factor in each batch, which is low computational cost. We use ResNet-32 and ResNet-10 to measure average run-time per epoch on CIFAR-10/100-LT and ImageNet-LT, respectively (see Tab. 5). We observe that FCC only increases about 6 seconds on ImageNet-LT, which is only an increase of 2.78%. For CIFAR-LT datasets, FCC roughly increases the run-time by 1 second, which seems like a big increase, this is because the training time on these datasets is very short. In fact, FCC only increases by about 3 minutes for 200 epochs.

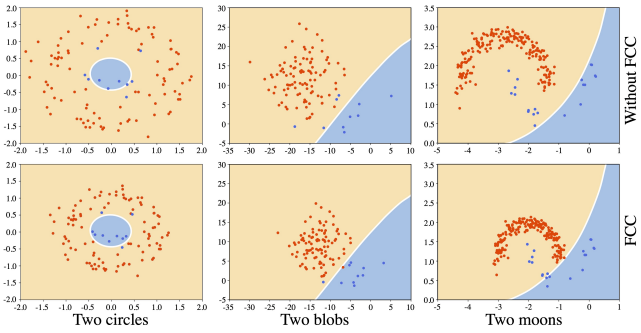


Figure 6. Impact of FCC on boundary points. Majority and minority classes are plotted in orange and blue, respectively. FCC can bring the points of minority classes back within the boundary.

5. Conclusion

In this work, we tackle long-tailed visual recognition from a novel perspective of increasing the density of BFs. In view of this, we propose a simple and generic method to improve the density, namely Feature Clusters Compression (FCC), which can be easily achieved and friendly combined with existing long-tailed methods to further boost them. Extensive experiments have fully verified the effectiveness and generality of our method.

Limitations. When using FCC, the shifting of feature clusters limits compression degree for higher performance improvement, or even degrades performance in some cases. Further work should be designed to eliminate the shifting.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.62077027), the Department of Science and Technology of Jilin Province, China (20230201086GX), the EU's Horizon 2020 FET proactive project (grant agreement No.823783), and the Department of Science and Technology of Jilin Province, China (20230601092FG).

References

[1] Shaden Alshammari, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. Long-tailed recognition via weight balancing.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6897–6907, 2022. 1, 6

[2] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998. 2

[3] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019. 1, 2, 3, 5, 6, 7

[4] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6, 2004. 2

[5] Pin-Chun Chen, Bo-Han Kung, and Jun-Cheng Chen. Class-aware robust adversarial training for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10420–10429, 2021. 1

[6] Hsin-Ping Chou, Shih-Chieh Chang, Jia-Yu Pan, Wei Wei, and Da-Cheng Juan. Remix: rebalanced mixup. In *European Conference on Computer Vision*, pages 95–110. Springer, 2020. 5, 6

[7] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019. 2, 5, 6

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 5

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 5, 6

[11] Yin-Yin He, Jianxin Wu, and Xiu-Shen Wei. Distilling virtual examples for long-tailed recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 235–244, 2021. 1, 3, 6

[12] KJ Joseph, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Towards open world object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5830–5840, 2021. 1

[13] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019. 1, 2, 3, 6

[14] Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 103–112, 2019. 2, 5

[15] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Roberto Togneri, and Ferdous A Sohel. A discriminative representation of convolutional features for indoor scene recognition. *IEEE Transactions on Image Processing*, 25(7):3372–3383, 2016. 1

[16] Byungju Kim and Junmo Kim. Adjusting decision boundary for class imbalanced learning. *IEEE Access*, 8:81674–81685, 2020. 5

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009. 5

[18] Jun Li, Zichang Tan, Jun Wan, Zhen Lei, and Guodong Guo. Nested collaborative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6949–6958, 2022. 1, 2, 3, 5, 6

[19] Tianhao Li, Limin Wang, and Gangshan Wu. Self supervision to distillation for long-tailed visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 630–639, 2021. 1, 3

[20] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003. 2

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2, 5, 6

[22] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2537–2546, 2019. 5

[23] Inderjeet Mani and I Zhang. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126, pages 1–7. ICML, 2003. 1, 2

[24] Seulki Park, Youngkyu Hong, Byeongho Heo, Sangdoo Yun, and Jin Young Choi. The majority can help the minority: Context-rich minority oversampling for long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6887–6896, 2022. 1

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[26] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 8

[27] Jiawei Ren, Cunjun Yu, Xiao Ma, Haiyu Zhao, Shuai Yi, et al. Balanced meta-softmax for long-tailed visual recognition. *Advances in neural information processing systems*, 33:4175–4186, 2020. 2, 6

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2, 6

[29] Alaa Tharwat. Principal component analysis: an overview. *Pattern Recognit*, 3(3):197–240, 2016. 4

[30] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 5

[31] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019. 5, 6

[32] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella X Yu. Long-tailed recognition by routing diverse distribution-aware experts. *arXiv preprint arXiv:2010.01809*, 2020. 1, 3, 6

[33] Tong Wu, Ziwei Liu, Qingqiu Huang, Yu Wang, and Dahua Lin. Adversarial robustness under long-tailed distribution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8659–8668, 2021. 1

[34] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 6

[35] Han-Jia Ye, Hong-You Chen, De-Chuan Zhan, and Wei-Lun Chao. Identifying and compensating for feature deviation in imbalanced deep learning. *arXiv preprint arXiv:2001.01385*, 2020. 2, 6

[36] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 5, 6

[37] Yifan Zhang, Bryan Hooi, Lanqing Hong, and Jiashi Feng. Self-supervised aggregation of diverse experts for test-agnostic long-tailed recognition. *Advances in Neural Information Processing Systems*, 3, 2022. 1, 3, 6

[38] Yongshun Zhang, Xiu-Shen Wei, Boyan Zhou, and Jianxin Wu. Bag of tricks for long-tailed visual recognition with deep convolutional neural networks. In *AAAI*, pages 3447–3455, 2021. 1, 2, 5

[39] Zhisheng Zhong, Jiequan Cui, Shu Liu, and Jiaya Jia. Improving calibration for long-tailed recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16489–16498, 2021. 2, 6

[40] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9719–9728, 2020. 3, 6

[41] Ye Zhu, Kai Ming Ting, and Mark J Carman. Density-ratio based clustering for discovering clusters with varying densities. *Pattern Recognition*, 60:983–997, 2016. 2

[42] Ye Zhu, Kai Ming Ting, Mark J Carman, and Maia Angelova. Cdf transform-and-shift: An effective way to deal with datasets of inhomogeneous cluster densities. *Pattern Recognition*, 117:107977, 2021. 2