Sample-specific and Context-aware Augmentation for Long Tail Image Classification

Anonymous authors
Paper under double-blind review

Abstract

Recent long-tail classification methods generally adopt the two-stage pipeline and focus on learning the classifier to tackle the imbalanced data in the second stage via re-sampling or re-weighting, but the classifier is easily prone to overconfidence in head classes. Data augmentation is a natural way to tackle this issue. Existing augmentation methods either perform low-level transformations or apply the same semantic transformation for all samples. However, meaningful augmentations for different samples should be different. In this paper, we propose a novel sample-specific and context-aware augmentation learning method for long-tail image classification. We model the semantic within-class transformation range for each sample by a specific Gaussian distribution and design a semantic transformation generator (STG) to predict the distribution from the sample itself. To encode the context information accurately, STG is equipped with a memory-based structure. We train STG by constructing ground-truth distributions for samples of head classes in the feature space. We apply STG to samples of tail classes for augmentation in the classifier-tuning stage. Extensive experiments on four imbalanced datasets show the effectiveness of our method.

1 Introduction

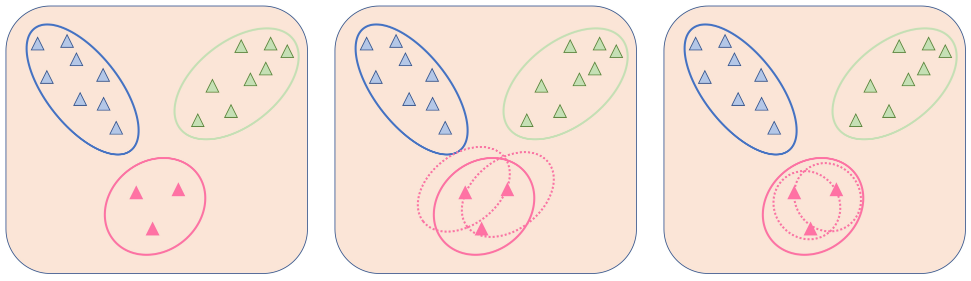
With the available of large-scale datasets such as ImageNet ILSVRC2012 (Russakovsky et al., 2015) and MS COCO (Lin et al., 2014), Convolutional Neural Networks(CNN) have achieved great success in image classification. The numbers of labeled samples of different classes in these datasets are balanced. However, data in the real-world often follows a long-tail distribution, i.e., a few dominant classes occupy most of the samples, while much fewer examples are available for most other classes. CNNs trained with such imbalanced data generalize well for head classes but easily overfit tail classes, and hence obtain degraded performances on balanced test data.

It has been demonstrated in (Kang et al., 2019; Zhou et al., 2020) that directly tackling the long-tail distribution by re-weighting or re-sampling techniques to train CNNs will hurt the capability of the learned representation. To alleviate the dilemma, many studies (Kang et al., 2019; Zhong et al., 2021) split the long-tail classification pipeline into a feature learning stage and a classifier learning stage. Solutions for adjusting the long-tail distribution are only applied to re-balance the classifier in the second stage. Such methods can well exploit the good feature representation learned from the original distribution. However, due to the imbalanced data, the classifier is still prone to overconfidence toward head classes and overfit to the limited samples of tail classes.

A natural way to alleviate such overconfidence is to augmenting samples for tail classes. Low-level image transformations such as random rotation, horizontal and color jittering are commonly used for augmentation (Cubuk et al., 2018). Although being effective, these augmentation techniques only slightly change the lighting, orientation, location, or color of samples, but are not capable of performing semantic transformations, e.g., changing the background of an object or the shape and texture of a foreground object. Therefore, the generated augmentations only distribute locally around existing samples and cannot effectively recover the real distribution. In (Wang et al., 2019), each class is modeled by a Gaussian distribution and augmentations are sampled from the distribution estimated in the feature space. It is difficult to reliably estimate the mean and covariance of a tail class since they are directly calculated from the few training samples. In (Li et al., 2021), a validation set is employed to calculate the mean and covariance in the way of meta learning. These methods treat each sample of the same class equally. However, different samples may locate in different regions of the real distribution and hence their meaningful within-class transformations are different. As is shown in Figure.1 (b), the within-class change ranges for the three samples are quite different. Applying the same global semantic variance range to all samples may lead to illegal augmentations that are out of the real distribution.

In this paper, we propose a novel three-stage method for long-tail image classification by exploring an additional sample-specific augmentation learning stage. In the first stage, we follow effective methods (Zhong et al., 2021) to learn the feature representation from the original imbalanced training samples. In the second stage, we propose a sample-specific and context-aware augmentation generation module to learn semantic augmentations for each sample. Since the within-class variance ranges for different samples are different, we explicitly model the augmentation distribution for each sample by an individual Gaussian distribution and use a Semantic Transformation Generator (STG) to estimate the distribution. We train STG from sufficient samples of head classes. To estimate the sample-specific statistics accurately, STG adopts a dictionary-based memory mechanism to preserve the context information. In the third stage, we apply STG to each sample of tail classes and achieve semantic data augmentation for fine-tuning the classifier. For each tail class, STG generates different meaningful and large-scale semantic distributions for augmenting different samples, so the augmented distribution better approaches the real distribution from few samples as shown in Figure.1 (c). Our major contributions are summarized as:

1) We propose a novel STG network to generate different transformation distributions for different samples. To train STG, we estimate the within-class variance distribution for each sample of the head class as the ground-truth, so that STG obtains sample-specific meaningful semantic augmentations and generalizes well to samples of tail classes.
2) To explore the relationship between other samples and encode the context information, we develop a dictionary-based memory mechanism in STG to make the estimations of the sample-specific mean and covariance more accurate.
3) We design a three-stage method for long-tail classification by learning data augmentations for tail classes in the feature space. We conduct extensive experiments on the CIFAR-10-LT, CIFAR-100-LT Krizhevsky et al. (2009), ImageNet-LT, Places-LT Liu et al. (2019) datasets and the results demonstrate the effectiveness of our method.



(a) Long-tail distribution (b) Class level augmentation (c) Sample level augmentation

Figure 1: Visualization of semantic data augmentation. (a) Example of the long-tail distribution, where the distributions of the two head classes are shown in blue and green, respectively, while the distribution of the tail class is shown in pink. Solid line denotes the original distribution. (b) Class-level augmentation. Since the same transformation is applied to all samples indiscriminately, the adjusted distribution contains illegal augmentations. Dotted line denotes the adjusted distribution. (c) The proposed Sample-specific augmentation. The semantic transformation is different for different sample. The adjusted distribution better recovers the real distribution of the tail class.

2 Related Work

Re-balancing strategies. Re-balancing strategies can be decomposed into re-sampling and re-weighting. The basic idea of re-sampling is to extend the amount of samples in tail classes or reduce the number of samples in head classes. These methods can be summarized as up-sampling (Buda et al., 2018; Byrd & Lipton, 2019) and under-sampling (Drummond et al., 2003; Buda et al., 2018). Up-sampling aims to repeat samples of tail classes while under-sampling focuses on removing a part of samples in the head classes. Compared with up-sampling, under-sampling has a smaller risk of overfitting, but it incurs the dilemma of removing important samples (Drummond et al., 2003).

Re-weighting strategies aim to allocate larger weights for training samples of tail classes and smaller weights for training samples of head classes in loss functions (Huang et al., 2016; Wang et al., 2017). Huang et al. (2016) and Mahajan et al. (2018) assigns the weights by the inverse of class frequency or the smooth version. In (Cui et al., 2019), a function is proposed to calculate the weights via the number of samples for each class. In (Tan et al., 2020), the re-weighting strategy is realized from another aspect and it randomly ignores discouraging losses for tail classes. Focal loss (Lin et al., 2017) is a typical and effective method to associate the weight with respect to the probability of each sample, where the gradient of high probability samples will be impaired heavily.

In (Zhou et al., 2020; Kang et al., 2019), it is shown that distribution of features and the distribution of category labels are inherently uncoupled. Therefore, many recent methods (Kang et al., 2019) follow the two-stage framework by decoupling representation learning and classifier learning. The re-balancing strategies are applied to turning the classifier rather than learning representations. Compared with these conventional re-balancing methods, our method generates new semantic samples of tail classes from few samples after the representation learning stage. An additional stage for learning to generate augmentations is performed before using augmented samples to learn the classifier. From the perspective that viewing augmentation as up-sampling, our method can be seen as an extension of traditional re-sampling strategy.

Augmentation-based methods. Augmentation-based methods aim to generate new samples based on original training data. Such methods can be decomposed into data level augmentation and feature level augmentation. Mixup (Zhang et al., 2017) generates synthetic data by interpolating pairs of samples and their labels. In (Kim et al., 2020) new samples of tail classes are generated by adding noises to samples of head classes to improve the generalization ability of the model. However, such low-level augmentations only change samples in the color space, which not only cannot guarantee to make sense in reality, but also cannot explore the uncovered spaces of the real distribution.

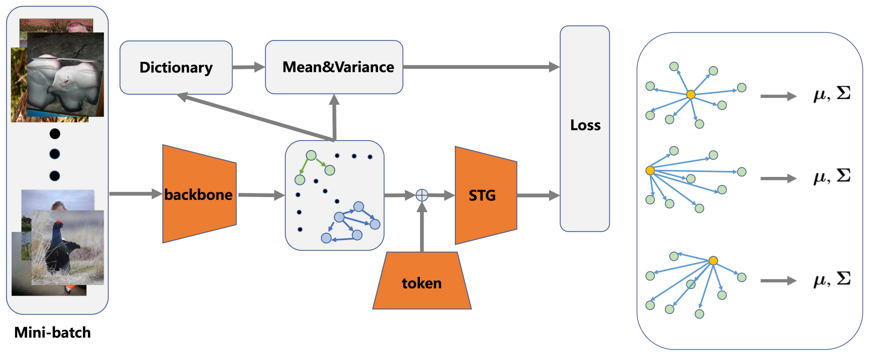
Differently, feature space augmentation (Chu et al., 2020) operates on the learned features. Its feature-level augmentation is performed by sampling two samples and fuse their feature maps. The shortcoming of such fusion-based method lies in the fact that it is easy to confuse the differences of different classes. BBN (Zhou et al., 2020) proposes a bilateral branch network and combines different features from instance balanced sampler and reversed sampler, respectively. MetaSAug (Li et al., 2021) uses semantic augmentations with meta-learning to enlarge features for tail classes.

Recently, in (Upchurch et al., 2017) extensive human annotations are used to explicitly find semantic transformation directions with additional manual consumption. ISDA (Wang et al., 2019) realizes implicit semantic augmentation by calculating mean and covariance of samples for each class and views samples of this distribution as available semantic directions. Since only few samples are available for tail classes, it is difficult to estimate the mean and covariance accurately. Actually, our method is also attributed as feature level augmentation. Different from other methods that apply the same transformation (e.g., adding random noises) to all samples, we distinguish different samples and model the transformation space of each samples as a separate Gaussian distribution. The statistics of the distribution for each sample are generated by a network learned from head classes rather than estimation from data. Sample-specific augmentation is achieved by sampling different transformations from the corresponding spaces for different samples. Compared with other methods, our method fully considers individual differences to generate semantically rational augmentations.

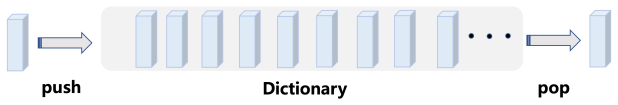
3 METHOD

3.1 PROBLEM DEFINITION AND NOTATIONS

We are given an imbalanced training dataset $\mathbb{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, where $\boldsymbol{y}_i \in \{1, \cdots, C\}$ is the label of the i^{th} sample $\boldsymbol{x}_i$, C is the number of classes, and n_c denotes the number of samples belongs to the c-th class. Without loss of generality, we assume that the classes are sorted by cardinality



(a) The learning progress of STG (b) Estimating $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$



(c) The memory-based dictionary

Figure 2: (a) The learning process of the proposed STG. (b) The estimation of $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$. The Yellow dot indicates the anchor feature, blue lines denote samples of semantic transformation and we calculate $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ by estimating from such transformations. Every feature has its own specific semantic transformations. (c) The memory-based dictionary. We push and pop features to refresh the dictionary.

in decreasing order, i.e., $n_1 \geq n_2 \geq ... \geq n_C$. The data obeys the long tail distribution, i.e., most samples belong to only a few head classes, while each of the other tail classes only has a few samples. In the test set, all classes have the same number of test samples, i.e., $n_1 \approx n_2 \cdots \approx n_C$. As a consequence, the model trained on imbalanced datasets performs poorly to recognize the tail classes.

Previous works (Kang et al., 2019) have already demonstrated the effectiveness of the two-stage strategy, which decouples the training process into the feature extraction learning stage and the classifier learning stage. In the first stage, a feature vector $\boldsymbol{a}_i$ is extracted from each sample $\boldsymbol{x}_i$ by the backbone network $f(\boldsymbol{x}_i;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the set of parameters of the backbone. In the second stage, the final classifier is denoted as $g(\boldsymbol{a}_i;\boldsymbol{w})$, where $\boldsymbol{w}$ is the parameter set of the classifier.

3.2 Overall Architecture

Different from the two-stage strategy, we add a semantic transformation learning stage and hence split the total pipeline into three stages. The first stage remains the same, in which we learn feature representations by training the parameters in $\boldsymbol{\theta}$ of backbone $f(\boldsymbol{x};\boldsymbol{\theta})$ from the imbalanced data. Mixup (Zhang et al., 2017) and self-supervised learning (Yang & Xu, 2020) have proved their effectiveness in tackling the long-tail problem. We choose mixup as the training strategy to learn the backbone. In the second stage, we freeze parameters of the backbone learned in the first stage and learn the proposed STG network from the features of the head classes, where details are described in Section 3.3. In the third stage, we utilize the learned STG to generate semantic transformation distributions for samples of the tail classes. Sample-specific augmentations are generated by sampling from the corresponding distribution and then used to fine-tune the final classifier $g(\boldsymbol{a};\boldsymbol{w})$. Details of the augmentation strategy are presented in Section 3.4.

3.3 Semantic Transformation Generator

It has been shown in (Upchurch et al., 2017) that translations in the well-learned feature space can produce meaningful semantic transformations, e.g., changing the color or posture of the object. In (Wang et al., 2019), the Implicit Semantic Data Augmentation (ISDA) method models semantic transformations for each class by a Gaussian distribution. By calculating class-wise covariances $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\Sigma}_C\}$, it defines the distribution of semantic augmentation transformations for each feature vector $\boldsymbol{a}_i$ as $\mathcal{N}(\boldsymbol{a}_i, \lambda\boldsymbol{\Sigma}_{y_i})$, where λ is a scale coefficient to rectify the vector length. Since only few samples are available for a tail class, it is hard to estimate the covariance and model the distribution of semantic transformations directly. MetaSAug (Li et al., 2021) utilizes a validation set to obtain semantic transformations indirectly. However, this method still does not consider the fact that semantic transformation ranges for different samples should be different. For instance, the semantic transformation vector that aims to change the color from red to blue cannot be applied to blue objects. On the contrary, our STG is proposed to learn semantic transformations at the instance level. The training progress of STG is shown in Figure 2.

We also make a hypothesis that the semantic transformations for a sample follow a Gaussian distribution. Since only a few samples are available for tail classes, the distribution cannot be reliably estimated from these samples. To tackle this problem, we train the STG network from head classes and use it to generate distributions for samples of tail classes. Different from ISDA and MetaSAug that estimate a single transformation covariance for all samples of the same class, STG aims to generate ad-hoc distribution for each sample in the feature space. Specifically, STG takes the feature of a sample as input and outputs the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the transformation distribution of this sample. We use two multilayer perceptrons with batch normalization (Ioffe & Szegedy, 2015) and the Relu activation as two modules to constitute the STG model and predict $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively.

In the additional second stage of our three-stage pipeline, we train STG from head classes. Since head classes have sufficient samples, the within-class variances for each sample can be directly estimated from other samples of the same class. Instead of estimating fixed statistics for each head sample offline from all training samples, we dynamically update such estimations with batches on the fly, which not only improves the efficiency but increases the diversity and randomness. Moreover, since each time a different local Gaussian is estimated, this also alleviates the limitation of single Gaussian modeling. Given a feature $\boldsymbol{a}_i$, we collect samples with the same class of $\boldsymbol{a}_i$ in the mini-batch into a set $\{\boldsymbol{a}_{1y_i}, \cdots, \boldsymbol{a}_{ky_i}\}$. The corresponding semantic transformation set of $\boldsymbol{a}_i$ is $\{\boldsymbol{a}_{1y_i} - \boldsymbol{a}_i, \cdots, \boldsymbol{a}_{ky_i} - \boldsymbol{a}_i\}$. We view elements in this set as samples of the Gaussian distribution of semantic transformations of $\boldsymbol{a}_i$. As is shown in Figure 2 (b), the mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$ can be estimated from these samples directly as follows:

$$\boldsymbol{\mu}_i = \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{a}_{jy_i} - \boldsymbol{a}_i \qquad\qquad \boldsymbol{\Sigma}_i = \frac{1}{k}\sum_{j=1}^{k} (\boldsymbol{\mu}_i - (\boldsymbol{a}_{jy_i} - \boldsymbol{a}_i))^2 \tag{1}$$

In addition, we hold the assumption that each channel of feature is independent and only calculate the diagonal of the covariance matrix for simplicity. The estimated $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ serve as the ground-truth statistics of $s(\boldsymbol{a}_i; \boldsymbol{\theta}_s)$ for the supervised learning of STG.

Although directly using $\boldsymbol{a}_i$ as the input of STG is available, the difference between different classes is not explicitly taken into consideration. Feasible semantic transformations for samples in the same class may be similar. Therefore, we define learnable class-specific tokens $\{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_C\}$ to distinguish different classes explicitly. These tokens are initiated by the normal distribution and participate in the updates by back-propagation. During the training phase, the corresponding class token $\boldsymbol{p}_{y_i}$ is added to $\boldsymbol{a}_i$, thus STG takes the input of $\boldsymbol{a}_i + \boldsymbol{p}_{y_i}$ and regress $\hat{\boldsymbol{\mu}}_i$ and $\hat{\boldsymbol{\Sigma}}_i$ directly. This process can be formulated as:

$$\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i = s(\boldsymbol{a}_i + \boldsymbol{p}_{y_i}; \boldsymbol{\theta}_s) \tag{2}$$

where $\boldsymbol{\theta}_s$ denotes the parameter set of the STG network s, $\hat{\boldsymbol{\mu}}_i$ and $\hat{\boldsymbol{\Sigma}}_i$ are the estimations of the mean and covariance, respectively.

Actually, the more abundant the samples, the more accurate the estimated statistics. In the STG learning stage, we only collect samples from the mini-batch and thus the number of samples is small. A plain strategy is to expand the size of the mini-batch, but it is limited by cuda memory. Inspired by MoCo (He et al., 2020), we add an external dictionary to store features of previous iterations. Our dictionary pushes in features of the recent mini-batch and pops out old features, as shown in Figure 2 (c). Therefore, features in our dictionary are dynamically updated to supply samples in each mini-batch to better estimate the statistics. Moreover, the make the predictions more stable, we smoothly update the estimated statistics for the same sample.

3.4 Semantic Transformation Augmentation

In the third stage, we use the trained STG to generate augmentations for tail classes. Because the tail classes have not participated in the learning stage of STG, no class-specific tokens with respect to these classes are learned. Therefore, we use the token of the most similar head class for each sample of tail classes. Given each sample feature $\boldsymbol{a}_i$ of the i-th tail class, we obtain its semantic transformation distribution with $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$ predicted by STG. Then we employ the reparameteric trick (Kingma & Welling, 2013) to generate augmentation samples as follows:

$$\boldsymbol{a}_i^{aug} = \boldsymbol{a}_i + (\sqrt{\boldsymbol{\Sigma}_i} \times \epsilon + \boldsymbol{\mu}_i) \times \lambda \times \beta_{y_i} \tag{3}$$

where $\epsilon \sim \mathcal{N}(0,1)$ and $(\sqrt{\boldsymbol{\Sigma}_i} \times \epsilon + \boldsymbol{\mu}_i)$ is a sampled semantic transformation vector. The hyper-parameter λ controls the strengthen of augmentations globally. β_{y_i} is another knob to control the strengthen at the class level, which is calculated as $\beta_{y_i} = (\frac{n_{min}}{n_{y_i}})^{1/2}$. n_{y_i} is the number of samples belonging to class y_i and $n_{min} = min(n_1, \cdots, n_C)$. Considering the number of samples in different classes also differs, we need to distinguish samples in different tail classes. The effect of this function is to inhibit the strengthen of classes with more samples than other tail classes.

Finally, we combine the augmented feature $\boldsymbol{a}_i^{aug}$ with the original feature $\boldsymbol{a}_i$ to obtain the enhanced representation. Since our augmentation is random and illegal features can be generated, we preserve part of the original feature to alleviate such illegal situation. We randomly sample a vector $\boldsymbol{m} \sim U(0,1)$ which has the same dimension as $\boldsymbol{a}_i$. We use a threshold $t \in (0,1)$ to control the augmentation rate. The final augmentation feature for $\boldsymbol{a}_i$ is defined by Eq. (4):

$$\tilde{\boldsymbol{a}}_i = \mathbf{1}_{[\mathrm{m}>\mathrm{t}]} \times \boldsymbol{a}_i + \mathbf{1}_{[\mathrm{m}<\mathrm{t}]} \times \boldsymbol{a}_i^{aug} \tag{4}$$

Empirically, we set $t = 0.5$, which means that we preserve 50% of the original features.

The loss function by using the augmented representation to train the classes is presented in Eq.(5):

$$L = \frac{1}{N} \sum_{i=1}^{N} -\log \frac{e^{g(\tilde{\boldsymbol{a}}_i, \boldsymbol{w}_{y_i})}}{\sum_{j=1}^{C} e^{g(\tilde{\boldsymbol{a}}_i, \boldsymbol{w}_{y_j})}} \tag{5}$$

Kang et al. (2019) propose different classifier re-training strategies such as cRT, LWS, τ-normalized, etc. Zhong et al. (2021) propose the generalized classifier for long-tail distribution. Experiments show that cRT is more proper for small scale datasets while LWS and the generalized classifier are more adaptable for large scale datasets. Therefore, we use cRT to re-train the classifier for common situations while adopt the generalization classifier on large scale datasets.

4 Experiment

4.1 Datasets and experimental setup

We evaluate our method on the CIFAR-10-LT dataset, the CIFAR-100-LT dataset, the ImageNet-LT dataset, and the Places365-LT dataset. To verify that our method can be applied to multiple backbone models, deep residual networks (ResNets) (He et al., 2016) with various depths are applied as the feature extraction backbone.

Datasets. CIFAR-10-LT and CIFAR-100-LT are simulated from balanced datasets. There are 10 classes and 100 classes in the two datasets, respectively. For both classes, on the test set, every class has an equal number of samples. We conduct experiments with different imbalance factor (IM), which is defined as N_{max}/N_{min}. N_{max} and N_{min} denote the volumes of the most frequent class and the least frequent class, respectively. We follow the same protocol in (Cao et al., 2019) to generate training datasets with imbalance factors of 10, 50, and 100, respectively. ImageNet-LT is generated from the ImageNet dataset (Russakovsky et al., 2015) and contains 115.8k images. The imbalance factor is 256, and the number of images per class ranges from 1280 to 5. Places365-LT is constructed from the Places-2 dataset and contains 6.5k images from 365 categories. Compared with the CIFAR-LT datasets, it suffers from extreme imbalance. The largest class contains 4980 images while the smallest one has only 5 images (IM = 996).

Experimental Setup. For CIFAR-10-LT and CIFAR-100-LT, we use ResNet-32 as our backbone following (Cao et al., 2019). In the first stage, we use the SGD optimizer and set the initial learning rate to 0.1. The first five epochs are trained with the linear warm-up (Goyal et al., 2017) learning rate schedule. The learning rate drops by 0.1 at epoch 160 and epoch 180, respectively. We follow the most popular setting to set the momentum and the weight decay to 0.9 and 2×10^{-4}, respectively. In the STG learning stage, we use the ADAM optimizer and set the learning rate to 0.001. In the third stage, we also use ADAM as the optimizer and set the learning rate to 0.0001 and 0.001 for CIFAR-100-LT and CIFAR-10-LT, respectively. For ImageNet-LT, we use ResNet-50 as the backbone and adopt the cosine learning rate schedule that gradually decays from 0.1 to 0 in the first stage. For Places365-LT, we use ResNet-152 as the backbone and the ImageNet pretrained parameters for initiation. Following OLTR (Liu et al., 2019), we set the initial learning rate to 0.01 and drop by 0.1 every 10 epochs. In the STG learning stage, we follow previous CIFAR-LT training strategies using the ADAM optimizer. In the classifier learning stage, we use the cosine learning rate that decays from 0.001 to 0.

Table 1: Performance on CIFAR-10-LT.

Method	100	50	10
CE	70.4	74.8	86.4
Mixup	73.1	77.8	87.1
LDAM-DRW	77.1	81.1	88.4
BBN	79.9	82.2	88.4
Remix-DRW	79.8	-	89.1
MetaSAug	80.7	84.3	89.7
cRT+mixup	79.1	84.2	89.8
LWS+mixup	76.3	82.6	89.6
MiSLAS	82.1	85.7	90.0
Ours	**83.1**	**85.9**	**90.4**

Table 2: Performance on CIFAR-100-LT.

Method	100	50	10
CE	38.4	43.9	55.8
Mixup	39.6	45.0	58.2
LDAM-DRW	42.1	46.7	58.8
BBN	42.6	47.1	59.2
Remix-DRW	46.8	-	61.3
MetaSAug	48.0	52.3	61.3
cRT+mixup	45.1	50.9	62.1
LWS+mixup	44.2	50.7	62.3
MiSLAS	47.0	52.3	63.2
Ours	**48.6**	**53.9**	**63.3**

4.2 Comparison with State-of-the-Arts

Results on CIFAR-LT. To demonstrate the effectiveness of our method, we compare our method with mixup (Zhang et al., 2017), LDAM-DRW (Cao et al., 2019), BBN (Zhou et al., 2020), Remix-DRW (Chou et al., 2020), MetaSAug (Li et al., 2021), cRT, LWS (Kang et al., 2019) and MiSLAS (Zhong et al., 2021). The performances on CIFAR-10-LT and CIFAR-100-LT are shown in Table 1 and Table 2, respectively. Compared with other methods, our method generally achieves better performances on both datasets. The improvement made by our method is larger with the imbalanced factor of 100 and weaker with the imbalanced factor of 10. Especially, our method outperforms other methods by a margin of 1% with the imbalanced factor 100 on both datasets. The inner reason is that the more extreme imbalanced distribution suffers from more severe overfitting, and our semantic augmentations can alleviate such overfitting better by exploring more accurate within-class covariances.

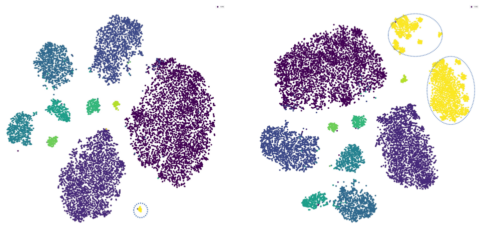

(a) w/o augmentation (b) w augmentation

Figure 3: The T-SNE (Van der Maaten & Hinton, 2008) visualizations of training samples on CIFAR-10-LT with IM=100. (a) Feature distribution of training samples. (b) Semantic augmentation samples. Some augmentation features are split into two parts since the most similar head classes for different features are different.

To better understand how our method influences classification, we visualize the decision boundary of our method and compared it with CE and cRT. As shown in Figure 3 and Figure 4, our method obtains better decision boundaries compared with other methods. This further supports our point that our method can alleviate the overfitting of tail classes and has better generalization ability.

Results on ImageNet-LT/Places365-LT. Our method can well scale to large scale datasets such as ImageNet-LT and Places365-LT. On ImageNet-LT, we add focal loss (Lin et al., 2017) and CE-DRW (Cao et al., 2019) for comparison. On Places365, we add OLTR (Liu et al., 2019), OLTR-LMFE (Xiang et al., 2020), and FeatureAug (Chu et al., 2020) for comparison. Results on the two datasets are shown on Table 3 and Table 4, respectively.

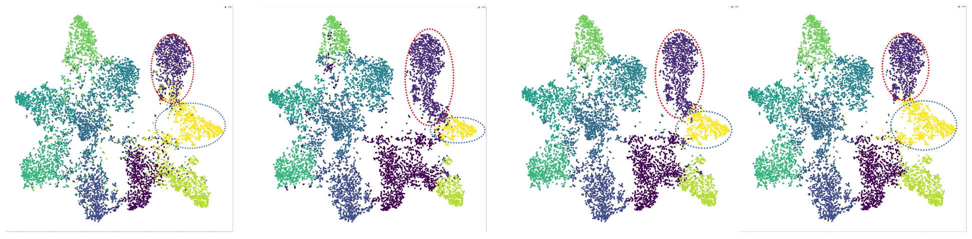

(a) Truth (b) CE (c) cRT (d) Ours

Figure 4: The T-SNE visualizations of different strategies and their decision boundaries on CIFAR-10-LT with IM=100. (a) The truth boundary of distributions on the validation set. Samples from the same class are shown in the same color. (b) The decision boundary by using the Cross Entropy loss. The yellow class (tail class) is easy to be decided as the purple class (head class). (c) The decision boundary of cRT. The misclassification phenomenon is alleviated but still exists. (d) The decision boundary of our method, which is more similar to the truth boundary compared with other methods.

Our method achieves better performance compared with other methods in ImageNet-LT. However, we obtain comparable performance with MiSLAS on Places365-LT. Specifically, we split Places-LT into ”many”, ”medium”, and ”few” classes according to the number of samples per class, where the detailed principle follows OLTR. As shown in Table 4, MiSLAS boosts the accuracy of the whole dataset by dropping performances on head classes heavily. In contrast, our method can achieve the same overall accuracy while preserving performances on head classes.

Table 3: Performance on ImageNet-LT.

Method	
CE	44.6
Mixup	45.5
LDAM-DRW	48.8
CE-DRW	48.5
Focal-DRW	47.9
MetaSAug	52.6
cRT+mixup	51.7
LWS+mixup	52.0
MiSLAS	52.7
Ours	**53.2**

Table 4: Performance on Places365-LT.

Method	all	many	medium	few
CE	27.2	45.9	22.4	0.36
Mixup	29.2	-	-	-
OLTR	35.9	44.7	37	25.3
OLTR-LMFE	36.2	-	-	-
FeatureAug	36.4	42.8	37.5	22.7
cRT+mixup	38.3	44.1	38.5	27.1
LWS+mixup	39.7	41.7	41.3	33.1
MiSLAS	**40.4**	39.6	43.3	36.1
Ours	**40.4**	43.5	41.9	31.6

Table 5: The KL divergence between semantic transformations. The numerical values show the differences of transformations for different samples. The KL divergence of the 4-th and 5-th samples are the most similar since their most similar head classes are the same.

	Sample1	Sample2	Sample3	Sample4	Sample5
Sample1	0	43.1	33.1	38.6	29.2
Sample2	37.2	0	41.2	62.8	54.5
Sample3	42.1	59.1	0	69.5	57.1
Sample4	28.2	55.8	45.9	0	22.2
Sample5	28.1	61.1	45.2	27.4	0
The most similar class	15	9	18	16	16

4.3 Ablation studies

What is learned from STG. To better understand what STG has learned, we randomly choose five samples of the same tail class from CIFAR-100-LT with imbalanced factor 100 and obtain their semantic transformations. As shown in Table 5, the most similar head classes for the 4-th and 5-th samples are the same, but are different for all other samples. Such differences determine that these samples use different knowledge, even if they come from the same tail class. Specifically, we calculate the KL divergence to estimate the difference between semantic transformations of different samples. We observe that the transformations differ greatly. The transformations for the 4-th and 5-th samples are similar because they both utilize the knowledge transferred from the same header class. For all other sample pairs, the divergence increases significantly. These results indicate the necessity of constructing instance-level semantic transformations and verify that our method can indeed generate sample-specific augmentations by taking the context information into consideration.

Effect of the dictionary-based memory. Precise estimations of the mean and covariance as ground-truth for samples of head classes are important since they directly influence the training of STG and hence the final performance. We estimate the statistics within mini-batches for efficiency and diversity, and employ the dictionary-based memory to tackle the problem of limited samples within mini-batches. We explore the influence of the dictionary by conducting experiments on the memory size with m=0, m=600, and m=1200. As shown in Table 6, a larger size of memory leads to better performances. Compared with CIFAR-100-LT, the performance on ImageNet-LT is more sensitive to the memory size. The reason is that ImageNet-LT has 1000 classes and there are more head classes. Since the number of head classes is large, the average samples per class in each mini-batch is smaller. Increasing the memory size can alleviate this issue effectively.

Table 6: Ablation study about the memory size of the dictionary on ImageNet-LT and CIFAR-100-LT with IM=100.

	ImageNet-LT	CIFAR-100-LT
m=0	52.6	48.3
m=600	52.9	48.4
m=1200	**53.2**	**48.6**

Different strategies against overfitting. One of the most important hyper-parameter in our method is λ controlling the strengthen of augmentations. There are also other strategies such as dropout (Srivastava et al., 2014) and adding Gaussian noises to features. We perform experiments to demonstrate the effectiveness of our method compared with such strategies. We set up three experiments of dropout , i.e., randomly setting zeros with probabilities p=0.1, p=0.2, and p=0.3, respectively, and four experiments of Gaussian noises with standard deviations std=0.1, std=0.5, std=1.0, and std=1.5, respectively. For our method, we also conduct three experiments with λ=0.5, λ=1.0, and λ=1.5, respectively. As shown in Table 7, our method achieves the best performance with λ=1.5. Compared with modifying features directly, our STG can capture better transformation directions to tackle overfitting.

Table 7: Ablation study about the hyper-parameter λ on CIFAR-10-LT and CIFAR-100-LT with IM=100. Comparisons with other methods against overfitting like dropout and adding Gaussian noises.

	CIFAR-10-LT	CIFAR-100-LT
Gaussian noisy-std=0.1	79.7	47.7
Gaussian noisy-std=0.5	80.0	47.6
Gaussian noisy-std=1.0	80.8	47.5
Gaussian noisy-std=1.5	81.4	47.0
Dropout-p=0.1	81.4	47.6
Dropout-p=0.2	81.8	47.8
Dropout-p=0.3	82.0	47.9
Ours-λ=0.5	81.0	48.1
Ours-λ=1	82.3	48.3
Ours-λ=1.5	**83.1**	**48.6**

5 Conclusion

In this paper, we have presented a novel instance-level and context-aware augmentation method for long tail image classification. Different from other augmentation methods that apply the same predefined or learned augmentation strategy, our method is able to generate customized transformations by STG for different samples of the tail classes. STG employs a dictionary-based memory mechanism to encode sample-specific context information. We estimate the transformation distributions for sufficient samples of head classes as ground-truth to train STG. Extensive experiments on four datasets show the effectiveness of our method. In the future, we intend to expand our method to other tasks such as long tail object detection and segmentation.

REFERENCES

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pp. 872–881, 2019.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, pp. 1567–1578, 2019.

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

Peng Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. *arXiv preprint arXiv:2008.03673*, 2020.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

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 Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277, 2019.

Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pp. 1–8. Citeseer, 2003.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

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*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5375–5384, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

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.

Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. M2m: Imbalanced classification via major-to-minor translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13896–13905, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Shuang Li, Kaixiong Gong, Chi Harold Liu, Yulin Wang, Feng Qiao, and Xinjing Cheng. Metasaug: Meta semantic augmentation for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5212–5221, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

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*, pp. 2980–2988, 2017.

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 Conference on Computer Vision and Pattern Recognition*, pp. 2537–2546, 2019.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 181–196, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11662–11671, 2020.

Paul Upchurch, Jacob Gardner, Geoff Pleiss, Robert Pless, Noah Snavely, Kavita Bala, and Kilian Weinberger. Deep feature interpolation for image content changes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7064–7073, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pp. 7029–7039, 2017.

Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. Implicit semantic data augmentation for deep networks. *Advances in Neural Information Processing Systems*, 32: 12635–12644, 2019.

Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *European Conference on Computer Vision*, pp. 247–263. Springer, 2020.

Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. *arXiv preprint arXiv:2006.07529*, 2020.

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

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*, pp. 16489–16498, 2021.

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*, pp. 9719–9728, 2020.