Logit Perturbation

Mengyang Li[1,2], **Fengguang Su**[2], **Ou Wu**[2*], **Ji Zhang**[3]

[1] Jiuantianxia Inc., China
[2] National Center for Applied Mathematics, Tianjin University, China
[3] The University of Southern Queensland, Australia
limengyang99@gmail.com, {fengguangsu, wuou}@tju.edu.cn, Ji.Zhang@usq.edu.au

Abstract

Features, logits, and labels are the three primary data when a sample passes through a deep neural network. Feature perturbation and label perturbation receive increasing attention in recent years. They have been proven to be useful in various deep learning approaches. For example, (adversarial) feature perturbation can improve the robustness or even generalization capability of learned models. However, limited studies have explicitly explored for the perturbation of logit vectors. This work discusses several existing methods related to logit perturbation. Based on a unified viewpoint between positive/negative data augmentation and loss variations incurred by logit perturbation, a new method is proposed to explicitly learn to perturb logits. A comparative analysis is conducted for the perturbations used in our and existing methods. Extensive experiments on benchmark image classification data sets and their long-tail versions indicated the competitive performance of our learning method. In addition, existing methods can be further improved by utilizing our method.

Introduction

There are several main paradigms (which may overlap) among numerous deep learning studies, including new network architecture, new training loss, new training data perturbation scheme, and new learning strategy (e.g., weighting). Training data perturbation mainly refers to feature and label perturbations.

In feature perturbation, many data augmentation tricks can be viewed as feature perturbation methods when the input is the raw feature (i.e., raw samples). For example, cropped or rotated images can be seen as the perturbed samples of the raw images in computer vision; sentences with modified words can also be seen as the perturbed texts in text classification. Another well-known feature perturbation technique is about the generation of adversarial training samples (Madry et al. 2018), which attracts great attention in various AI applications especially in computer vision (Xie et al. 2020) and natural language processing (Jin et al. 2020). Adversarial samples are those that can fool the learned models. They can be obtained by solving the following objective function:

$$x_{adv} = x + \underset{\|\delta\| \leq \epsilon}{\arg\max}\, l(f(x+\delta), y), \tag{1}$$

where x is the input or the hidden feature, δ is the perturbation term, ϵ is the perturbation bound, y is the label, and x_{adv} is the generated adversarial sample. A number of methods have been proposed to optimize Eq. (1) (Goodfellow, Shlens, and Szegedy 2015; Madry et al. 2018). Adversarial samples can be used to train more robust models.

In label perturbation, the labels are modified or corrected to avoid overfitting and noises. For example, a popular yet simple training trick, label smoothing (Szegedy et al. 2016), generates a new label for each sample according to $y' = y + \lambda(\frac{I}{C} - y)$, where C is the number of categories, I is a vector with all elements equaling to 1, $(\frac{I}{C} - y)$ is the perturbation term, and λ is a hyper-parameter. Other methods such as Boostrapping loss (Reed et al. 2015), label correction (Patrini et al. 2017; Wang et al. 2021), and Meta label corrector (Wu et al. 2021b) can be seen as a type of label perturbation. Mixup (Zhang et al. 2018) can be attributed to the combination of feature and label perturbation.

Logit vectors (or logits) are the outputs of the final feature encoding layer in almost deep neural networks (DNNs). Although logits are important in the DNN data pipeline, only several learning methods in data augmentation and long-tail classification directly (without optimization) or implicitly employ class-level logit perturbation. Based on the loss variations of these methods, the loss variations incurred by logit perturbation are highly related to the purpose of positive/negative augmentation[1] on training data. Accordingly, a new method is proposed to learn a class-level logit perturbation (LPL) in this study. A comparative analysis is conducted for two classical methods and our LPL method. Extensive experiments are run on benchmark data sets. The results show the competitiveness of our method. The contributions of our study are as follows:

- A new method is proposed to learn to perturb logits which can be used in data augmentation and long-tail classification contexts. Experimental results show that our method outperforms existing state-of-the-art methods related to logit perturbation in both contexts.
- Several classical methods are re-discussed in terms of

*Corresponding author.

[1] In this study, negative augmentation denotes the augmentation which aims to reduce the (relative) performances of some categories. Accordingly, existing augmentation methods are positive.

logit perturbation and positive/negative augmentation. The differences between existing and our proposed logit perturbation methods are also analyzed.

Related Work

Data Augmentation

Data augmentation is prevailed in almost all deep learning approaches. In its early stage, heuristic operations on raw samples are utilized, such as image flip, image rotation, and word replacing in sentences. Recently, advanced tricks are investigated, such as mixup (Zhang et al. 2018) and semantic data augmentation (Wang et al. 2019). In mixup, given a sample $\{x_1, y_1\}$, its perturbation term is $\{\lambda(x_2 - x_1), \lambda(y_2 - y_1)\}$, where λ is a random parameter (not a hyper-parameter), and $\{x_2, y_2\}$ is another randomly selected sample. Hu et al. (2019) introduced reinforcement learning to automatically augment data.

In this study, existing data augmentation is called positive data augmentation. Negative data augmentation, which is proposed in this study, may be helpful when we aim to restrain the (relative) performance of certain categories (e.g., to keep fairness in some tasks).

Long-tail Classification

Real data usually conform to a skewed or even a long-tail distribution. In long-tail classification, the proportions of tail samples are considerably small compared with those of head samples. Long-tail classification may be divided into two main strategies. The first strategy is to design new network architectures. Zhou et al. (2020) designed a bilateral-branch network to learn the representations of head and tail samples. The second strategy is to modify the training loss. In this way, the weighting scheme (Fan et al. 2017) is the most common practice. Relatively larger weights are exerted on the losses of the tail samples. Besides weighting, some recent studies modify the logits to change the whole loss, such as logit adjustment (LA) (Menon et al. 2021). This new path achieves higher accuracies in benchmark data corpora compared with weighting (Wu et al. 2021a).

Methodology

This section first discusses several typical learning methods related to logit perturbation.

The notations and symbols are defined as follows. Let $S = \{x_i, y_i\}_{i=1}^N$ be a corpus of N training samples, where x_i is the input feature and y_i is the label. Let C be the number of category and $\pi_c = N_c/N$ be the proportion of the samples in the cth category in S. Without loss of generality, we assume that $\pi_1 > \cdots > \pi_c > \cdots > \pi_C$. Let u_i be the logit vector of x_i which can be obtained by $u_i = f(x_i, W)$, where $f(\cdot, \cdot)$ is the deep neural network with parameter W. Let δ_i be the perturbation term of x_i. Let $\mathcal{L}$ be the training loss and l_i be the loss of x_i. The standard cross-entropy (CE) loss is used throughout the study.

Logit Perturbation in Existing Methods

Logit adjustment (LA) (Menon et al. 2021). This method is designed for long-tail classification and achieves competitive performance in benchmark data sets (Wu et al. 2021a). The employed loss in LA is defined as follows:

$$\begin{aligned}\mathcal{L} &= \sum\nolimits_i l(\text{softmax}(u_i + \delta_i), y_i) \\ &= -\sum\nolimits_i \log \frac{\exp(u_{i,y_i} + \lambda \log \pi_{y_i})}{\sum_c \exp(u_{i,c} + \lambda \log \pi_c)}.\end{aligned} \tag{2}$$

In Eq. (2), the perturbation term δ_i is as follows:

$$\delta_i = \tilde{\delta} = \lambda[\log\pi_1, \cdots, \log\pi_c, \cdots, \log \pi_C]^T, \tag{3}$$

where $\tilde{\delta}$ represents that δ_i is a corpus-level vector; thus the values of δ_i for all the samples in the corpus S are identical. Eq. (2) can be re-written as follows:

$$\mathcal{L} = -\sum\nolimits_i \log \frac{\exp(u_{i,y_i})}{\sum_c \exp(u_{i,c} + \lambda(\log \pi_c - \log \pi_{y_i}))}. \tag{4}$$

Previously, we assumed that $\pi_1 > \cdots > \pi_c > \cdots > \pi_C$; hence, the losses of the samples in the first category (head) are decreased, while those of the samples in the last category (tail) are increased. The variations of the losses of the rest categories depend on the concrete loss of each sample.

Implicitly semantic data augmentation (ISDA) (Wang et al. 2019). ISDA is a newly proposed data augmentation method. Given a sample x_i, ISDA assumes that each (virtual) new sample can be sampled from a distribution $\mathcal{N}(x_i, \Sigma_{y_i})$, where Σ_{y_i} is the co-variance matrix for the y_ith category. With the M (virtual) new samples for each sample, the loss becomes

$$\mathcal{L} = -\frac{1}{N \cdot M} \sum\nolimits_{i=1}^{N} \sum\nolimits_{m=1}^{M} l(f(x_{i,m}, W), y_i), \tag{5}$$

where $x_{i,m}$ is the mth (virtual) new sample for x_i. When $M \to +\infty$, the upper bound of the loss in Eq. (5) becomes

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(u_{i,y_i})}{\sum\limits_{c=1}^{C} \exp(u_{i,c} + \frac{\lambda}{2}(\mathrm{w}_c - \mathrm{w}_{y_i})^T \Sigma_{y_i} (\mathrm{w}_c - \mathrm{w}_{y_i}))}, \tag{6}$$

where w_c is the network parameter for the logit vectors and $u_{i,c} = \mathrm{w}_c{}^T \tilde{x}_i + b_c$, where $\tilde{x}_i$ is the output of the last feature encoding layer. In contrast with previous data augmentation methods, ISDA does not generate new samples or features. In Eq. (6), there is an implicit perturbation term δ_i defined as follows:

$$\delta_i = \tilde{\delta}_{y_i} = \frac{\lambda}{2} \begin{bmatrix} (\mathrm{w}_1 - \mathrm{w}_{y_i})^T \Sigma_{y_i} (\mathrm{w}_1 - \mathrm{w}_{y_i}) \\ \vdots \\ (\mathrm{w}_C - \mathrm{w}_{y_i})^T \Sigma_{y_i} (\mathrm{w}_C - \mathrm{w}_{y_i}) \end{bmatrix}, \tag{7}$$

which is a category-level vector. Each element of δ_i is non-negative. Therefore, the new loss of each category from Eq. (7) is larger than the loss from the standard CE loss.

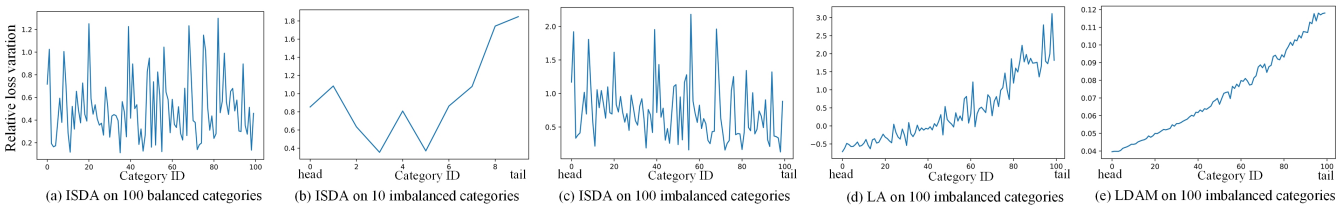
LDAM (Cao et al. 2019). In this method, the new loss is defined as

$$\mathcal{L} = -\sum_{i=1}^{N} \log \frac{\exp(u_{i,y_i} - C(\pi_{y_i})^{-1/4})}{\exp(u_{i,y_i} - C(\pi_{y_i})^{-\frac{1}{4}}) + \sum_{c \neq y_i} \exp(u_{i,c})}. \tag{8}$$

The perturbation term δ_i is as follows:

$$\delta_i = \tilde{\delta}_{y_i} = \lambda[0, \cdots, -C(\pi_{y_i})^{-\frac{1}{4}}, \cdots, 0]^T, \tag{9}$$

which is also a category-level vector. The losses for all categories are increased in LDAM.



(a) ISDA on 100 balanced categories (b) ISDA on 10 imbalanced categories (c) ISDA on 100 imbalanced categories (d) LA on 100 imbalanced categories (e) LDAM on 100 imbalanced categories

Figure 1: The relative loss variations ($\frac{l'-l}{l}$) of the three methods on different categories on different data sets.

Our Proposed Method LPL

The losses of the three example methods analyzed in the previous subsection can be written as follows:

$$\mathcal{L}=\sum_i l(\text{softmax}(u_i+\tilde{\delta}_{y_i}), y_i). \tag{10}$$

All the three methods directly infer the perturbation term without a numerical optimization approach. Logit perturbations result in the loss variations. Fig. 1 shows the statistics for the relative loss variations incurred by ISDA, LA, and LDAM for each category on a balanced data set (CIFAR100) and two long-tail sets (CIFAR10-LT and CIFAR100-LT) which are introduced in the experimental section. The loss variations of all categories are positive using ISDA. ISDA achieves the worst results on CIFAR100-LT (shown in the experimental parts), indicating that the non-tail-priority augmentation in long-tail problems is ineffective (ISDA achieves relatively better results on CIFAR10-LT). Only the curves on CIFAR100-LT are shown for LA and LDAM because similar trends can be observed on CIFAR10-LT. The loss variations of head categories are negative, and those of tail are positive using LA. All the variations are positive yet there is an obvious increasing trend using LDAM.

We propose two conjectures based on the above observations and from a unified data augmentation viewpoint:

- If one aims to positively augment the samples in a category, the loss of this category should be increased. The larger the loss increment, the greater the augmentation.
- If one aims to negatively augment the samples in a category, then the loss of this category should be reduced. The larger the loss decrement, the greater the negative augmentation.

The above two conjectures are supported in the aforementioned three methods. To handle a long-tail problem, LA should positively augment tail samples and negatively augment head samples; hence, the losses of tails are increased, and those of heads are decreased. ISDA aims to positively augment samples in all categories; thus, the losses for all categories are increased. LDAM aims to positively augment tail samples more than head samples. Hence, the increments of tail categories are larger than those of the head.

On the basis of our conjectures, we establish the following new learning loss with logit perturbation:

$$\begin{aligned}\mathcal{L}=&\sum_{c\in\mathcal{N}_a}\sum_{x_i\in S_c}\min_{\|\tilde{\delta}_c\|\le\epsilon_c} l(\text{softmax}(u_i+\tilde{\delta}_c), c)\\ &+\sum_{c\in\mathcal{P}_a}\sum_{x_i\in S_c}\max_{\|\tilde{\delta}_c\|\le\epsilon_c} l(\text{softmax}(u_i+\tilde{\delta}_c), c),\end{aligned} \tag{11}$$

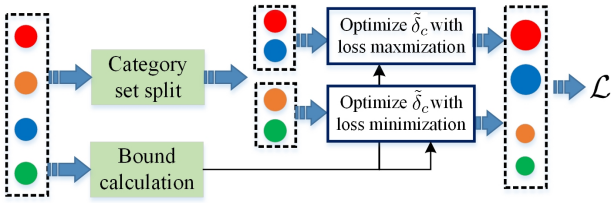


Figure 2: Overview of the logit perturbation-based new loss. Four solid circles denote four categories. Two categories are positively augmented via loss maximization and the rest two are negatively augmented via minimization.

where ϵ_c is the perturbation bound related to the extent of augmentation; $\mathcal{N}_a$ is the index set of categories which should be negatively augmented; $\mathcal{P}_a$ is the index set of categories which should be positively augmented; and S_c is the set of samples in the cth category. The loss maximization for the $\mathcal{P}_a$ categories is actually the category-level adversarial learning on the logits; the loss minimization for the $\mathcal{N}_a$ categories is the opposite. Fig. 2 illustrates the calculation of the logit perturbation-based new loss in Eq. (11).

The split of the category set (i.e., $\mathcal{N}_a$ and $\mathcal{P}_a$) and the definition (calculation) of ϵ_c are crucial for the learning with Eq. (11). Category set split determines the categories that should be positively or negatively augmented. Meanwhile, the value of ϵ_c determines the augmentation extent.

Category set split. The split depends on specific learning tasks. Two common cases are explored in this study. The first case splits categories according to their performances. In this case, Eq. (11) becomes the following compact form:

$$\begin{aligned}\mathcal{L}=&\sum_c\{\mathbb{S}(\tau-\bar{q}_c)\times\\ &\sum_{x_i\in S_c}\max_{\|\tilde{\delta}_c\|\le\epsilon_c}[l(\text{softmax}(u_i+\tilde{\delta}_c), c)\mathbb{S}(\tau-\bar{q}_c)]\},\end{aligned} \tag{12}$$

where $\mathbb{S}(\cdot)$ is the signum function (if $a\ge 0$, then $\mathbb{S}(a)=1$), τ is a threshold, and $\bar{q}_c$ is calculated by

$$\bar{q}_c=\frac{1}{N_c}\sum_{x_i\in S_c} q_{i,c}=\frac{1}{N_c}\sum_{x_i\in S_c}\frac{\exp(u_{i,c})}{\sum_{c'}\exp(u_{i,c'})}. \tag{13}$$

When $\tau=\text{mean}(\bar{q}_c)=\sum_{c=1}^{C}\bar{q}_c/C$, Eq. (12) indicates that if the performance of a category is below the mean performance, it will be positively augmented. Meanwhile, when the performance is above the mean, it will be negatively augmented. When $\tau>\max_c\{\bar{q}_c\}$, all the categories will be positively augmented as in ISDA.

The second case is special for a long-tail problem, and it splits categories according to the proportion order of each

Algorithm 1: Learning to Perturb Logits (LPL)

Input: S, τ, max iteration T, hyper-parameters for PGD-like optimization, and other conventional training hyper-parameters.

1: Randomly initialize Θ.
2: **for** $t = 0$ to T **do**
3: Sample a mini-batch from S.
4: Update τ if it is not fixed (e.g., mean$(\bar{q}_c)$ is used) and split the category set.
5: Compute ϵ_c for each category using (18) if varied bounds are used.
6: Infer δ_c for each category using a PGD-like optimization method for (12) in balanced classification or (14) in long-tail classification.
7: Update the logits for each sample and compute the new cross entropy loss.
8: Update Θ with SGD.
9: **end for**

Output: Θ

category. Eq. (11) becomes the following compact form:

$$\mathcal{L} = \sum\nolimits_c \{\mathbb{S}(c-\tau)\times \sum_{x_i\in S_c} \max_{\|\tilde{\delta}_c\|\le\epsilon_c} [l(\text{softmax}(u_i+\tilde{\delta}_c), c)\mathbb{S}(c-\tau)]\}, \quad (14)$$

where τ is a threshold for the category index. In (14), the tail categories locate in $\mathcal{P}_a$ and will be positively augmented.

Eqs. (12) and (14) can be solved with an optimization approach similar to PGD (Madry et al. 2018). First, we have

$$\frac{\partial l(\text{softmax}(u_i+\tilde{\delta}_{y_i}), y_i)}{\partial \tilde{\delta}_{y_i}} = \text{softmax}(u_i) - \hat{y}_i, \quad (15)$$

where $\hat{y}_i$ is the one-hot vector of y_i. In the maximization of (12) and (14), $\tilde{\delta}_{y_i}$ is updated by

$$\tilde{\delta}'_{y_i} = \tilde{\delta}_{y_i} + \frac{\lambda}{N_{y_i}} \sum_{j:y_j=y_i} (\text{softmax}(u_j) - \hat{y}_j), \quad (16)$$

where λ is the hyper-parameter. In the minimization part, $\tilde{\delta}_{y_i}$ is updated by

$$\tilde{\delta}'_{y_i} = \tilde{\delta}_{y_i} - \frac{\lambda}{N_{y_i}} \sum_{j:y_j=y_i} (\text{softmax}(u_j) - \hat{y}_j). \quad (17)$$

The details are introduced in the supplementary file.

Bound calculation. The category with a relatively low/high performance should be more positively/negatively augmented; the category closer to the tail/head should be more positively/negatively augmented. We define

$$\epsilon_c = \epsilon + \Delta\epsilon\,|\tau - \bar{q}_c| \quad \text{or} \quad \epsilon_c = \begin{cases} \epsilon + \Delta\epsilon \frac{\bar{q}_c}{\bar{q}_1} & c \le \tau \\ \epsilon + \Delta\epsilon \frac{\bar{q}_C}{\bar{q}_c} & c > \tau \end{cases}. \quad (18)$$

In Eq. (18), the larger the difference between the performance ($\bar{q}_c$) of the current category and the threshold τ, or the larger the ratio $\bar{q}_c/\bar{q}_1$ (and $\bar{q}_C/\bar{q}_c$), the larger the bound ϵ_c. This notion is in accordance with our previous conjecture. When $\Delta\epsilon$ in Eq. (18) equals to zero, the bound is fixed.

The algorithmic steps of our LPL are in Algorithm 1. The optimization in Step 6 is detailed in the supplementary file.

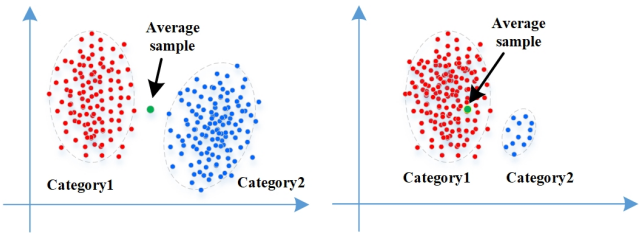


Figure 3: Illustrative example for LA. All samples are perturbed by the average sample.

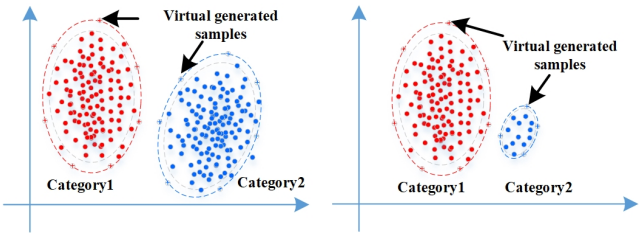


Figure 4: Illustrative example for ISDA. Both categories are positively augmented (new samples are virtually generated) according to feature distributions.

Comparative Analysis

This subsection compares the perturbations in LA, ISDA, and our LPL in terms of data augmentation. We first have the following Lemma for LA.

Lemma 1. Assume that there is an average sample x_{avg} that satisfies softmax$(u_{avg}) = \{\pi_1, \cdots, \pi_C\}$. The perturbation term on the logit vector of x_i in LA equals to λu_{avg}.

The proof is in the supplementary file. Fig. 3 presents an illustrative example. The two categories in the left case are balanced, while they are imbalanced in the right. With LA, all samples are perturbed by the average sample x_{avg}.

In the ISDA's rationale, new samples are (virtually instead of really) generated based on the distribution of each category. Fig. 4 shows the (virtually) generated samples by ISDA. In the right case, the positive augmentation for head category may further hurt the performance of the tail category. ISDA failed in the long-tail problem. Li et al. (2021) leveraged meta learning to adapt ISDA for the long-tail problem.

In contrast with the above-mentioned two methods, our proposed LPL method conducts positive or negative augmentation according to the directions of loss maximization and minimization. Theoretically, loss maximization will force the category to move close to the decision boundary (i.e., the category is positively augmented or virtual samples are generated for this category). By contrast, loss minimization will force the category to be far from the boundary (i.e., the category is negatively augmented or samples are virtually deleted for this category). Fig. 5 shows an illustrative example.

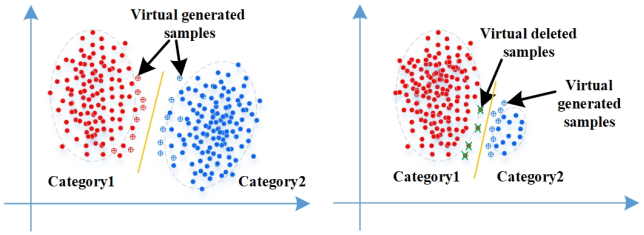


Figure 5: Illustrative example for LPL. Samples near the classification boundary are virtually generated or deleted.

Experiments

Our proposed LPL is first evaluated on data augmentation and long-tail classification tasks. The properties of LPL are then analyzed with more experiments. All the codes are available and attached in the supplementary file.

Experiments on Data Augmentation

Datasets and competing methods. In this subsection, two benchmark image classification data sets, namely, CIFAR10 and CIFAR100, are used. Both data consist of 32×32 natural images in 10 classes for CIFAR10 and 100 classes for CIFAR100. There are 50,000 images for training and 10,000 images for testing. The training and testing configurations used in (Wang et al. 2019) are followed. Several classical and state-of-the-art robust loss functions and (semantic) data augmentation methods are compared: Large-margin loss (Liu et al. 2016), Disturb label (Xie et al. 2016), Focal Loss (Lin et al. 2017), Center loss (Wen et al. 2016), Lq loss (Zhang and Sabuncu 2018), CGAN (Mirza and Osindero 2014), ACGAN (Odena, Olah, and Shlens 2017), infoGAN (Chen et al. 2016), ISDA, and ISDA + Dropout.

The Wide-ResNet-28 (Zagoruyko and Komodakis 2016) and ResNet-110 (He et al. 2016) are used as the base neural networks. Considering that the training/testing configuration is fixed for both sets, the results of the above competing methods reported in the ISDA paper (Wang et al. 2019) are directly presented (some are from their original papers). The training settings for the above base neural networks also follow the instructions of ISDA paper and its released codes. Our methods have two variants.

- LPL (mean+fixed bound). In this version, the optimization in Eq. (12) is used. Mean denotes that the threshold is mean($\bar{q}_c$). Fixed bound means that the value of ϵ_c is fixed and identical for all categories during optimization. It is searched in $\{0.1, 0.2, 0.3, 0.4\}$.
- LPL (mean+varied bound). In this version, the optimization in Eq. (12) is used. Theoretically, varied bound means that the value of ϵ_c is varied according to Eq. (18). However, the varied bounds in the same batch make the implementation more difficult and increase the training complexity. In our implementation, we choose to set a varied number of updating steps for each category in our PGD-like optimization, which is described in the supplementary file. The value of $\Delta\epsilon$ is searched in $\{0.1, 0.2\}$.

The PGD-like optimization in Algorithm 1 contains two hyper-parameters, namely, step size and #steps. The setting of these parameters is introduced in the supplementary file.

Method	Wide-ResNet-28-10	ResNet-110
Basic	3.82 ± 0.15%	6.76 ± 0.34%
Large Margin	3.69 ± 0.10%	6.46 ± 0.20%
Disturb Label	3.91 ± 0.10%	6.61 ± 0.04%
Focal Loss	3.62 ± 0.07%	6.68 ± 0.22%
Center Loss	3.76 ± 0.05%	6.38 ± 0.20%
Lq Loss	3.78 ± 0.08%	6.69 ± 0.07%
CGAN	3.84 ± 0.07%	6.56 ± 0.14%
ACGAN	3.81 ± 0.11%	6.32 ± 0.12%
infoGAN	3.81 ± 0.05%	6.59 ± 0.12%
ISDA	3.58 ± 0.15%	6.33 ± 0.19%
ISDA+DropOut	3.58 ± 0.15%	5.98 ± 0.20%
LPL (mean+ fixed ϵ_c)	3.39 ± 0.04%	5.83 ± 0.21%
LPL (mean+ varied ϵ_c)	**3.37 ± 0.04%**	**5.72 ± 0.05%**

Table 1: Mean values and standard deviations of the test Top-1 errors for all the involved methods on CIFAR10.

Method	Wide-ResNet-28-10	ResNet-110
Basic	18.53 ± 0.07%	28.67 ± 0.44%
Large Margin	18.48 ± 0.05%	28.00 ± 0.09%
Disturb Label	18.56 ± 0.22%	28.46 ± 0.32%
Focal Loss	18.22 ± 0.08%	28.28 ± 0.32%
Center Loss	18.50 ± 0.25%	27.85 ± 0.10%
Lq Loss	18.43 ± 0.37%	28.78 ± 0.35%
CGAN	18.79 ± 0.08%	28.25 ± 0.36%
ACGAN	18.54 ± 0.05%	28.48 ± 0.44%
infoGAN	18.44 ± 0.10%	27.64 ± 0.14%
ISDA	17.98 ± 0.15%	27.57 ± 0.46%
ISDA+DropOut	17.98 ± 0.15%	26.35 ± 0.30%
LPL (mean+ fixed ϵ_c)	18.19 ± 0.07%	26.09 ± 0.16%
LPL (mean+ varied ϵ_c)	**17.61 ± 0.30%**	**25.87 ± 0.07%**

Table 2: Mean values and standard deviations of the test Top-1 errors for all the involved methods on CIFAR100.

The Top-1 error is used as the evaluation metric. The performances of the base neural networks with the standard cross-entropy loss are re-run before running our methods to conduct a fair comparison. Almost identical results are obtained compared with the published results in the ISDA paper.

Results. Tables 1 and 2 present the results of all competing methods on the CIFAR10 and CIFAR100, respectively. Our LPL method (two versions) achieves the best performance almost under both the two base neural networks. ISDA achieves the second-best performance. Only in the case of Wide-ResNet-28-10 on CIFAR100, LPL (mean+fixed ϵ_c) is inferior to ISDA. However, the former still achieves the fourth lowest error.

The results of LPL with varied bounds are better than those of LPL with fixed bounds. This comparison indicates the rationality of our motivation that the category with relatively low (high) performance should be more positively (negatively) augmented. In the final part of this section, more analyses will be conducted to compare ISDA and our

Ratio	100:1	10:1
Class-balanced CE loss	61.23%	42.43%
Class-balanced fine-tuning	58.50%	42.43%
Meta-weight net	58.39%	41.09%
Focal Loss	61.59%	44.22%
Class-balanced focal loss	60.40%	42.01%
LDAM	59.40%	42.71%
LDAM-DRW	57.11%	41.22%
ISDA + Dropout	62.60%	44.49%
LA	56.11%	41.66%
LPL (varied τ + fixed ϵ_c)	58.03%	41.86%
LPL (varied τ + varied ϵ_c)	**55.75%**	**39.03%**

Table 3: Test Top-1 errors on CIFAR100-LT (ResNet-32).

Ratio	100:1	10:1
Class-balanced CE loss	27.32%	13.10%
Class-balanced fine-tuning	28.66%	16.83%
Meta-weight net	26.43%	12.45%
Focal Loss	29.62%	13.34%
Class-balanced focal loss	25.43%	12.52%
LDAM	26.45%	12.68%
LDAM-DRW	21.88%	11.63%
ISDA + Dropout	27.45%	12.98%
LA	22.33%	11.07%
LPL (varied τ + fixed ϵ_c)	23.97%	11.09%
LPL (varied τ + varied ϵ_c)	**22.05%**	**10.59%**

Table 4: Test Top-1 errors on CIFAR10-LT (ResNet-32).

method. Naturally, the varied threshold will further improve the performances. More results are in the supplementary file.

Experiments on Long-tail Classification

Datasets and competing methods. In this experiment, the long-tail version of CIFAR10 and CIFAR100 compiled by Cui et al. (2019) are used and called CIFAR 10-LT and CIFAR 100-LT, respectively. The training and testing configurations used in (Menon et al. 2021) are followed. Several classical and state-of-the-art robust loss functions and semantic data augmentation methods are compared: Class-balanced CE loss (Wang et al. 2019), Class-balanced fine-tuning (Cui et al. 2018), Meta-weight net (Shu et al. 2019), Focal loss (Lin et al. 2017), Class-balanced focal loss (Cui et al. 2019), LDAM (Cao et al. 2019), LDAM-DAR (Cao et al. 2019), ISDA, and LA.

Menon et al. (2021) released the training data when the imbalance ratio (i.e., π_1/π_{100}) is 100:1; hence, their data and reported results for the above competing methods are directly presented. When the ratio is 10:1, the results of ISDA+Dropout and LA are obtained by running their released codes. The results of the rest methods are from the study conducted by Li et al. (2021). The hyper-parameter λ in LA is searched in $\{0.5, 1, 1.5, 2, 2.5\}$ according to the suggestion in (Menon et al. 2021). Similar to the experiments in (Menon et al. 2021), ResNet-32 (He et al. 2016) is used as the base network. The results of ISDA, LA, and LPL are the average of five repeated runs.

Our methods have two variants: LPL (varied threshold + fixed bound) and LPL (varied threshold + varied bound).

Ratio	100:1		10:1	
LA	56.11%	22.33%	41.66%	11.07%
LPL	55.75% (-0.36%)	22.05% (-0.28%)	39.03% (-2.63%)	10.59% (-0.48%)

Table 5: The error reduction of LPL (varied τ+varied ϵ) over LA on the two data sets.

Method	CIFAR10-LT100	CIFAR100-LT100
LA	22.33%	56.11%
LPL	22.05%	55.75%
LA+LPL	**21.46%**	**53.89%**

Table 6: Test Top-1 errors of three methods on two data sets.

The threshold τ is searched in $\{0.4C, 0.5C, 0.6C\}$. The parameters step size and #steps for the PGD-like optimization are detailed in the supplementary materials. The value of ϵ is set to 0, and $\Delta\varepsilon$ is searched in $\{1.5, 2.5, 5\}$.

Only one meta-based method, Meta-weight net, is involved, because we mainly aim to compare methods that only modify the training loss. In addition, meta-based methods require an auxiliary high-quality validation set (Li et al. 2021). Other methods, such as BBN (Zhou et al. 2020), which focus on the new network structure are also not included in the comparisons.

Results. The Top-1 error is also used. Table 3 shows the results of all the methods on the CIFAR100-LT data. On the ratios 100:1 and 10:1, LPL (varied τ + varied ϵ_c) yields the lowest Top-1 errors. It exceeds the best competing method "LA" by 0.36% and 2.63% on the ratios 100:1 and 10:1, respectively. Table 4 shows the results of all the methods on the CIFAR10-LT data. LPL (varied τ + varied ϵ_c) still obtains the lowest Top-1 errors on both ratios. On all the comparisons, the semantic augmentation method ISDA obtains poor results. On CIFAR100-LT, ISDA achieves the worst performances on both ratios. This result is expected because ISDA aims to positively augment all categories equally and does not favor tail categories, which may lead to tail categories suffering from this positive augmentation. Nevertheless, ISDA has a better performance on CIFAR10-LT than on CIFAR100-LT. In Fig. 1(b), the loss increments of tail categories are larger than those of the head ones. That is, larger augmentations are exerted on tail categories.

We listed the Top-1 errors of LA and LPL (varied τ + varied ϵ_c) on Table 5 to better present the comparison. When the ratio is smaller, the improvements (error reductions) are relatively larger. This result is reasonable because when the ratio becomes small, the effectiveness of LA will be subsequently weakened. When the imbalance ratio is one, indicating that there is no imbalance, LA will lose effect; however, our LPL can still augment the training data effectively.

More Analysis for Our Method

Improvements on existing methods. Our LPL method seeks the perturbation via an optimization scheme. In ISDA and LA, the perturbations are directly calculated rather than optimization. A natural question arises, that is, whether the perturbations in existing methods further improved via our method. Therefore, we propose a combination method with the following loss in imbalance image classification:

Method	#Params	CIFAR10	CIFAR100
ResNet-32+ISDA	0.5M	7.09 ± 0.12%	30.27 ± 0.34%
ResNet-32+LPL (mean + fixed ϵ_c)	0.5M	7.01 ± 0.16%	29.59 ± 0.27%
ResNet-32+LPL (mean + varied ϵ_c)	0.5M	**6.66 ± 0.09%**	**28.53 ± 0.16%**
SE-Resnet110+ISDA	1.7M	5.96 ± 0.21%	26.63 ± 0.21%
SE-Resnet110+LPL (mean + fixed ϵ_c)	1.7M	5.87 ± 0.17%	26.12 ± 0.24%
SE-Resnet110+LPL (mean + varied ϵ_c)	1.7M	**5.39 ± 0.10%**	**25.70 ± 0.07%**
Wide-ResNet-16-8+ISDA	11.0M	4.04 ± 0.29%	19.91 ± 0.21%
Wide-ResNet-16-8+LPL (mean + fixed ϵ_c)	11.0M	3.97 ± 0.09%	19.87 ± 0.02%
Wide-ResNet-16-8+LPL (mean + varied ϵ_c)	11.0M	**3.93 ± 0.10%**	**19.83 ± 0.09%**

Table 7: Number of parameters and test Top-1 errors of ISDA and LPL with different base networks.

$$\sum_{c\in\mathcal{N}_a}\sum_{x_i\in S_c}\min_{\|\tilde{\delta}_{y_i}\|\le\epsilon_c} l(\text{softmax}(u_i+\lambda\log\pi_{y_i}+\tilde{\delta}_{y_i}),y_i)$$
$$+\sum_{c\in\mathcal{P}_a}\sum_{x_i\in S_c}\max_{\|\tilde{\delta}_{y_i}\|\le\epsilon_c} l(\text{softmax}(u_i+\lambda\log\pi_{y_i}+\tilde{\delta}_{y_i}),y_i).$$

When all ϵ_cs are zero, the above-mentioned loss becomes the loss of LA; when λ is zero, the above loss becomes our LPL (with fixed bound). We conducted experiments on CIFAR10-LT100 and CIFAR100-LT100. The results are shown in Table 6. ResNet-32 is used as the basic model. The value of λ is searched in $\{0.5, 1, 1.5, 2, 2.5\}$ and other parameters are detailed in the supplementary file.

The combination method LA+LPL achieves the lowest errors on both comparisons, indicating that our LPL can further improve the performances of existing SOTA methods. ISDA can likewise be improved with the same manner.

More comparisons with ISDA. ISDA claims that it does not increase the number of parameters compared with the direct learning with the basic DNN models. Our method also does not increase the number of model parameters. The reason lies in that the perturbation terms are no longer used in the final prediction.

Table 7 shows the comparisons between ISDA and LPL (two variants) on three additional base DNN models, namely, SE-ResNet110 (Hu, Shen, and Sun 2018), Wide-ResNet-16 (Zagoruyko and Komodakis 2016), and ResNet-32. The numbers of parameters are equal for ISDA and LPL. Nevertheless, the two variants of our method LPL outperform ISDA on both data sets under all the five base models.

Loss variations of LPL during training. We plot the loss variations of LPL on two balanced and two long-tail data sets to assess whether our method LPL is in accordance with the two conjectures. The curves are shown in Fig. 6. On the balanced data, the relative loss variations are similar to those of ISDA; on the long-tail data, the losses of head categories are reduced, whereas those of tail ones are increased, which is similar to those of LA. The two variations indicate that LPL can implement appropriate positive/negative augmentation according to the characteristics of training data.

Performances of LPL under different τ and ϵ_c. Both the threshold for category set split and the bound for augmentation extent are two important hyper-parameters in LPL. Based on our experiments, the following observations are obtained. On the balanced data sets, the results are relatively stable when the bound locates in [0.1, 0.5]; when the threshold is searched around the mean$(\bar{q}_c)$, the results are usually better. On the long-tail data sets, the results are relatively stable when the bound locates in [1.5, 5.0]. When the threshold is searched in $\{0.4C, 0.5C, 0.6C\}$, the results are usually good in our experiment. Long-tail problems require larger extent of data augmentation.

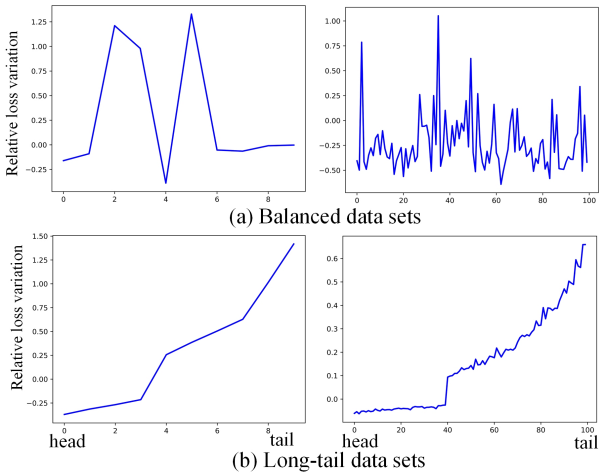


Figure 6: Relative loss variations of our LPL on two balanced and two long-tail data sets.

Training time consumption. Our proposed LPL is efficient as the optimization is directly run on logits. In balanced data, the time consumption of LPL is about 35% lower than that of ISDA. In long-tail data, the time consumption of LPL is about 15% higher than those of LA and vanilla CE loss.

Conclusions

This study investigates the category-level logit perturbation in deep learning. A conjecture for the relationship between (logit perturbation-incurred) loss increment/decrement and positive/negative data augmentation is proposed. Based on this conjecture, a new method is introduced to learn to perturb logits (LPL) during DNN training. Two key components of LPL including category-set split and boundary calculation are investigated. The differences between our proposed LPL and two existing methods are analyzed. Extensive experiments on data augmentation (for balanced classification) and long-tail classification are conducted. LPL achieves the best performances in both situations under different basic networks. Existing methods with logit perturbation (e.g. LA) can also be improved by using our method.

References

Cao, K.; Wei, C.; Gaidon, A.; Arechiga, N.; and Ma, T. 2019. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 1567–1578.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2180–2188.

Cui, Y.; Jia, M.; Lin, T.-Y.; Song, Y.; and Belongie, S. 2019. Class-balanced loss based on effective number of samples. In *CVPR*, 9268–9277.

Cui, Y.; Song, Y.; Sun, C.; Howard, A.; and Belongie, S. 2018. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 4109–4118.

Fan, Y.; Lyu, S.; Ying, Y.; and Hu, B.-G. 2017. Learning with average top-k loss. In *NeurIPS*, 497–505.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *CVPR*, 7132–7141.

Hu, Z.; Tan, B.; Salakhutdinov, R.; Mitchell, T.; and Xing, E. P. 2019. Learning data manipulation for augmentation and weighting. In *NeurIPS*, 15738–15749.

Jin, D.; Jin, Z.; Zhou, J. T.; and Szolovits, P. 2020. Is bert really robust? A strong baseline for natural language attack on text classification and entailment. In *AAAI*, 8018–8025.

Li, S.; Gong, K.; Liu, C. H.; Wang, Y.; Qiao, F.; and Cheng, X. 2021. MetaSAug: Meta semantic augmentation for Long-tailed visual recognition. In *CVPR*, 5212–5221.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *ICCV*, 2980–2988.

Liu, W.; Wen, Y.; Yu, Z.; and Yang, M. 2016. Large-margin softmax loss for convolutional neural networks. In *ICML*, 507–516.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.

Menon, A. K.; Jayasumana, S.; Rawat, A. S.; Jain, H.; Veit, A.; and Kumar, S. 2021. Long-tail learning via logit adjustment. In *ICLR*.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. arXiv:1411.1784.

Odena, A.; Olah, C.; and Shlens, J. 2017. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2642–2651.

Patrini, G.; Rozza, A.; Krishna Menon, A.; Nock, R.; and Qu, L. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 1944–1952.

Reed, S.; Lee, H.; Anguelov, D.; Szegedy, C.; Erhan, D.; and Rabinovich, A. 2015. Training deep neural networks on noisy labels with bootstrapping. In *ICLR*.

Shu, J.; Xie, Q.; Yi, L.; Zhao, Q.; Zhou, S.; Xu, Z.; and Meng, D. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 1917–1928.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*, 2818–2826.

Wang, X.; Hua, Y.; Kodirov, E.; Clifton, D. A.; and Robertson, N. M. 2021. Proselflc: Progressive self label correction for training robust deep neural networks. In *CVPR*, 752–761.

Wang, Y.; Pan, X.; Song, S.; Zhang, H.; Huang, G.; and Wu, C. 2019. Implicit semantic data augmentation for deep networks. In *NeurIPS*, 12635–12644.

Wen, Y.; Zhang, K.; Li, Z.; and Qiao, Y. 2016. A discriminative feature learning approach for deep face recognition. In *ECCV*, 499–515. Springer.

Wu, T.; Liu, Z.; Huang, Q.; Wang, Y.; and Lin, D. 2021a. Adversarial robustness under long-tailed distribution. In *CVPR*, 8659–8668.

Wu, Y.; Shu, J.; Xie, Q.; Zhao, Q.; and Meng, D. 2021b. Learning to purify noisy labels via meta soft label corrector. In *AAAI*, 10388–10396.

Xie, C.; Tan, M.; Gong, B.; Wang, J.; Yuille, A. L.; and Le, Q. V. 2020. Adversarial examples improve image recognition. In *CVPR*, 819–828.

Xie, L.; Wang, J.; Wei, Z.; Wang, M.; and Tian, Q. 2016. Disturblabel: Regularizing cnn on the loss layer. In *CVPR*, 4753–4762.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. In *BMVC*, 87.1–87.12.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. Mixup: Beyond empirical risk minimization. In *ICLR*.

Zhang, Z.; and Sabuncu, M. R. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 8778–8788.

Zhou, B.; Cui, Q.; Wei, X.-S.; and Chen, Z.-M. 2020. BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, 9719–9728.

Supplementary material to the paper "Logit Perturbation"

S.1. Section "Methodology"

1) More details and typo-corrections to Eqs. (15)-(17)

We propose a more specific optimization method called PGD-like optimization based on PGD. According to the derivative of the cross-entropy loss function with respect to logit vector, our PGD-like optimization method can be implemented simply. There are typos in Eqs. (15)-(17) in the submission. They are corrected and the inference details are described below.

First, we have

$$\begin{aligned} l(\text{softmax}(u_i + \tilde{\delta}_{y_i}), y_i) &= -\log \frac{\exp(u_{i,y_i} + \tilde{\delta}_{y_i,y_i})}{\sum_{c'} \exp(u_{i,c'} + \tilde{\delta}_{y_i,c'})} \\ &= \log \textstyle\sum_{c'} \exp(u_{i,c'} + \tilde{\delta}_{y_i,c'}) - (u_{i,y_i} + \tilde{\delta}_{y_i,y_i}). \end{aligned} \tag{s-1}$$

Then we get the following Eq. (s-2):

$$\begin{aligned} &\frac{\partial l(\text{softmax}(u_i + \tilde{\delta}_{y_i}), y_i)}{\partial \tilde{\delta}_{y_i}} \\ &= [\frac{\exp(u_{i,1} + \tilde{\delta}_{y_i,1})}{\sum_{c'} \exp(u_{i,c'} + \tilde{\delta}_{y_i,c'})}, ..., \frac{\exp(u_{i,y_i} + \tilde{\delta}_{y_i,y_i})}{\sum_{c'} \exp(u_{i,c'} + \tilde{\delta}_{y_i,c'})} - 1, ..., \frac{\exp(u_{i,C} + \tilde{\delta}_{y_i,C})}{\sum_{c'} \exp(u_{i,c'} + \tilde{\delta}_{y_i,c'})}]^T \\ &= \text{softmax}(u_i + \tilde{\delta}_{y_i}) - \hat{y}_i. \end{aligned} \tag{s-2}$$

According to Eq. (s-2) and Gradient Descent, we can solve the maximization and minimization in Eq. (12) and Eq. (14). Let u_i^k be the logit vector after perturbation at the step k. In the maximization of Eqs.(12) and (14), $\tilde{\delta}_{y_i}^{k+1}$ is updated by

$$\begin{aligned} \tilde{\delta}_{y_i}^{k+1} &= \frac{\lambda}{N_{y_i}} \sum_{j:y_j=y_i} \frac{\partial l(\text{softmax}(u_j^k + \tilde{\delta}_{y_i}), y_j)}{\partial \tilde{\delta}_{y_i}}\Big|_{\mathbf{0}} \\ &= \frac{\lambda}{N_{y_i}} \sum_{j:y_j=y_i} (\text{softmax}(u_j^k) - \hat{y}_j), \end{aligned} \tag{s-3}$$

where λ is the hyper-parameter. In the minimization of Eqs. (12) and (14), $\tilde{\delta}_{y_i}^{k+1}$ is updated by

$$\begin{aligned} \tilde{\delta}_{y_i}^{k+1} &= -\frac{\lambda}{N_{y_i}} \sum_{j:y_j=y_i} \frac{\partial l(\text{softmax}(u_j^k + \tilde{\delta}_{y_i}), y_j)}{\partial \tilde{\delta}_{y_i}}\Big|_{\mathbf{0}} \\ &= -\frac{\lambda}{N_{y_i}} \sum_{j:y_j=y_i} (\text{softmax}(u_j^k) - \hat{y}_j). \end{aligned} \tag{s-4}$$

Then, in the next update, $u_i^{k+1} = u_i^k + \tilde{\delta}_{y_i}^{k+1}$. Eqs.(15)-(17) are revised to the following equations:

$$\frac{\partial l(\text{softmax}(u_i' + \tilde{\delta}_{y_i}), y_i)}{\partial \tilde{\delta}_{y_i}}\Big|_{\mathbf{0}} = \text{softmax}(u_i') - \hat{y}_i. \tag{15'}$$

$$\tilde{\delta}'_{y_i} = \frac{\lambda}{N_{y_i}} \sum_{j:y_j=y_i} (\text{softmax}(u'_j) - \hat{y}_j). \tag{16'}$$

$$\tilde{\delta}'_{y_i} = -\frac{\lambda}{N_{y_i}} \sum_{j:y_j=y_i} (\text{softmax}(u'_j) - \hat{y}_j). \tag{17'}$$

2) The optimization in Step 6 in Algorithm 1

First, we calculate the perturbation bound ϵ_c according to Eq. (18), where c is the category. Let α be the step size. Then, the number of steps (#steps) for category c is calculated by

$$K_c = \lfloor \frac{\epsilon_c}{\alpha} \rfloor. \tag{s-5}$$

With step size α and #steps, the PGD-like optimization is detailed in Algorithm S1.

Algorithm S1 PGD-like Optimization

Input: The logit vectors (u_i) for the cth category in the current mini-batch, ϵ_c, and α.

1: Let $u_i^0 = u_i$ for the input vectors;
2: Calculate K_c according to Eq. (s-5);
3: **for** k = 0 to $K_c - 1$ **do**
4: Calculate $\frac{\partial l(\text{softmax}(u_i^k + \tilde{\delta}_c), c)}{\partial \tilde{\delta}_c}\Big|_{\mathbf{0}} = \text{softmax}(u_i^k) - \hat{c}$.
5: Calculate $\tilde{\delta}_{y_i}^{k+1}$ according to Eq. (s-3) for maximization and Eq. (s-4) for minimization;
6: $u_i^{k+1} := u_i^k + \tilde{\delta}_{y_i}^{k+1}$.
7: **end for**=0

Output: $\delta_c = u_i^{K_c} - u_i$

3) The proof of Lemma 1

Lemma 1. Assume that there is an average sample x_{avg} that satisfies $\text{softmax}(u_{avg}) = \{\pi_1, \cdots, \pi_C\}$. The perturbation term on the logit vector of x_i in LA equals to λu_{avg}.

Proof: Assume that there is an average sample x_{avg} who satisfies

$$\begin{aligned} \text{softmax}(u_{avg}) &= [\frac{\exp(u_{avg,1})}{\sum_{c'} \exp(u_{avg,c'})}, ..., \frac{\exp(u_{avg,c})}{\sum_{c'} \exp(u_{avg,c'})}, ..., \frac{\exp(u_{avg,C})}{\sum_{c'} \exp(u_{avg,c'})}]^T \\ &= [\pi_1, ..., \pi_c, ..., \pi_C]^T. \end{aligned} \tag{s-6}$$

Therefore, $u_{avg,c} = \log\pi_c + \Delta$, where Δ is an arbitrary constant. We have the following equation:

$$\frac{\exp(u_{i,c} + u_{avg,c} + \Delta)}{\sum_{c'} \exp(u_{i,c'} + u_{avg,c'} + \Delta)} = \frac{\exp(u_{i,c} + u_{avg,c})}{\sum_{c'} \exp(u_{i,c'} + u_{avg,c'})}. \tag{s-7}$$

The arbitrary constant Δ does not affect the loss value. Therefore, Δ is set as zero and we have

$$\{u_{avg,1}, \cdots, u_{avg,C}\} = \{\log\pi_1, \cdots, \log\pi_C\}. \tag{s-8}$$

We further assume that x_{avg} is the output of the last feature encoding layer. Therefore,

$$W^T(x_i + \lambda x_{avg})) + b_i + \lambda b_{avg} = u_i + \lambda u_{avg}. \tag{s-9}$$

According to Eqs. (s-8) and (s-9) and LA, the perturbation term on the logit vector of x_i equals to λu_{avg}. The proof is done.

4) Explanation of why x_{avg} locates in the first head category.

In our submission, we assume $\pi_1 > \pi_2 > ... > \pi_C$ for the long-tail problem. For the average sample, we have softmax$(u_{avg}) = \{\pi_1, ..., \pi_C\}$. Naturally, x_{avg} locates in the (first) head category as π_1 is the largest in the prediction.

According to Lemma 1, we have following proposition 1.

Proposition 1: If $\lambda \to +\infty$, then each training sample will be classified into the (first) head category.

Proof: According to Lemma 1, we get the u_i' as follows:

$$u_i' = u_i + \lambda u_{avg}. \tag{s-11}$$

Then we calculate the softmax value of u_i' as follows:

$$\text{softmax}(u_i') = [\frac{\exp(u_{i,1} + \lambda u_{avg,1})}{\sum_{c'} \exp(u_{i,c'} + \lambda u_{avg,c'})}, ..., \frac{\exp(u_{i,c} + \lambda u_{avg,c})}{\sum_{c'} \exp(u_{i,c'} + \lambda u_{avg,c'})}, ..., \frac{\exp(u_{i,C} + \lambda u_{avg,C})}{\sum_{c'} \exp(u_{i,c'} + \lambda u_{avg,c'})}]. \tag{s-12}$$

Then we calculate the limits of softmax(u_i') as follows:

$$\begin{aligned}
&\lim_{\lambda \to +\infty} \frac{\exp(u_{i,1} + \lambda u_{avg,1})}{\sum_{c'} \exp(u_{i,c'} + \lambda u_{avg,c'})} \\
&= \lim_{\lambda \to +\infty} \frac{1}{\sum_{c'} \exp(u_{i,c'} - u_{i,1} + \lambda(u_{avg,c'} - u_{avg,1}))} \\
&= \lim_{\lambda \to +\infty} \frac{1}{1 + \sum_{c' \geq 2} \exp(u_{i,c'} - u_{i,1} + \lambda(u_{avg,c'} - u_{avg,1}))} \\
&= 1.
\end{aligned} \tag{s-13}$$

However, $\forall c \geq 2$, we get the limits as follows:

$$\begin{aligned}
&\lim_{\lambda \to +\infty} \frac{\exp(u_{i,c} + \lambda u_{avg,c})}{\sum_{c'} \exp(u_{i,c'} + \lambda u_{avg,c'})} \\
&= \lim_{\lambda \to +\infty} \frac{1}{\sum_{c'} \exp(u_{i,c'} - u_{i,c} + \lambda(u_{avg,c'} - u_{avg,c}))} \\
&= \lim_{\lambda \to +\infty} \frac{1}{1 + \sum_{c' > c} \exp(u_{i,c'} - u_{i,c} + \lambda(u_{avg,c'} - u_{avg,c})) + \sum_{c' < c} \exp(u_{i,c'} - u_{i,c} + \lambda(u_{avg,c'} - u_{avg,c}))} \\
&= 0.
\end{aligned} \tag{s-14}$$

Therefore, when $\lambda \to +\infty$, x_i will be classified into the (first) head category. In summary, the proof is completed.

S.2. Subsection "Experiments on Data Augmentation"

1) Explanation for using varied #steps instead of varied ϵ_c

If varied ϵ_c is used, the implementation is complicated and the time consumption is relatively large according to our empirical observations. It is simplier and more efficient to control the perturbation bound directly based on the varied #steps according to Eq. (s-5). In our future work, we will explore how to directly set the varied ϵ_c in the codes.

2) Parameter settings for PGD-like optimization. The step size is searched in $\{0.01, 0.02, 0.03\}$.

3) Results with varied thresholds.

Fig. S-1 shows the results under different thresholds on CIFAR10 and CIFAR100. The thresholds are $\{$0.8mean, 0.9mean, mean, 1.1mean, 1.2mean$\}$. ResNet32 is used. The performances are stable under the five thresholds. On CIFAR10, when the threshold is 0.9mean, the performance is the best.

S.3. Subsection "Experiments on Long-tail Classification"

1) The parameters step size and steps for the PGD-like optimization The step size is searched in $\{0.1, 0.2, 0.3\}$ because the augmentation extent should be large in long-tail classification.

2) The parameter setting for the combination method LA+LPL The value of λ is searched in $\{0.1, 1, 2\}$. The threshold τ is set as 4 and 40 on CIFAR10 and CIFAR100, respectively. Other parameters follow the setting in the previous experiments.

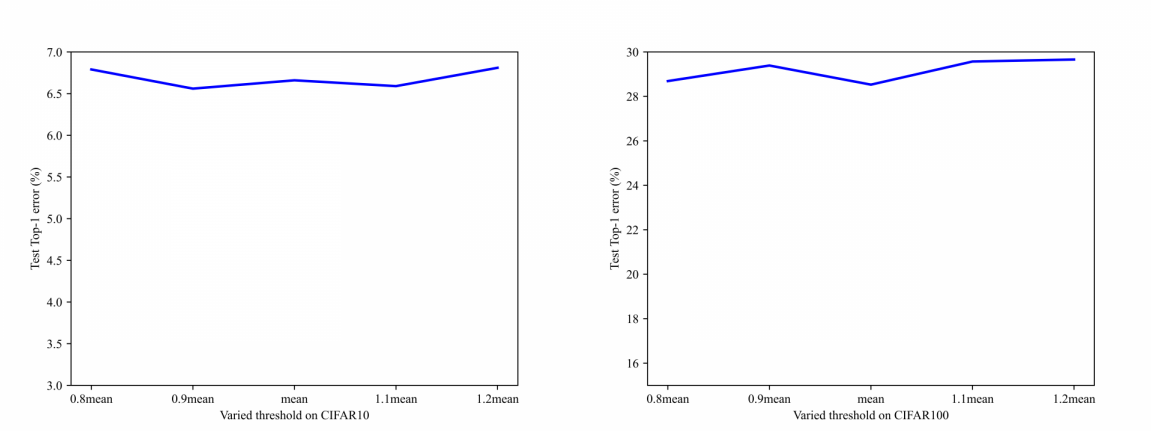


Figure S-1. Test Top-1 errors under different thresholds on CIFAR10 (left) and CIFAR100 (right).

model	CIFAR10	CIFAR100
ResNet32+LA	15.18s	16.17s
ResNet32+LPL	17.56s	18.41s
ResNet32+ISDA	26.85s	27.72s

Table S-1. Average time consumption in an epoch.

S.4. Subsection “More Analysis for Our Method”

Table S-1 shows the average time consumption in an epoch of the three methods ResNet32, ResNet32+LTP, and ResNet32+ISDA on the two data sets of CIFAR10 and CIFAR100. On CIFAR10 and CIFAR100, the time costs of LPL are 34.59% and 33.58% lower than ISDA, respectively. The time costs of LPL are 15.67% and 13.8% larger than the LA algorithm, respectively.

S.5. Section “Experiments”

All codes can be found in the ”code.zip” compressed package.