Procrustean Training for Imbalanced Deep Learning

Han-Jia Ye De-Chuan Zhan
State Key Laboratory for Novel Software Technology, Nanjing University, China
{yehj, zhandc}@lamda.nju.edu.cn

Wei-Lun Chao
The Ohio State University, USA
chao.209@osu.edu

Abstract

Neural networks trained with class-imbalanced data are known to perform poorly on minor classes of scarce training data. Several recent works attribute this to over-fitting to minor classes. In this paper, we provide a novel explanation of this issue. We found that a neural network tends to first under-fit the minor classes by classifying most of their data into the major classes in early training epochs. To correct these wrong predictions, the neural network then must focus on pushing features of minor class data across the decision boundaries between major and minor classes, leading to much larger gradients for features of minor classes. We argue that such an under-fitting phase over-emphasizes the competition between major and minor classes, hinders the neural network from learning the discriminative knowledge that can be generalized to test data, and eventually results in over-fitting. To address this issue, we propose a novel learning strategy to equalize the training progress across classes. We mix features of the major class data with those of other data in a mini-batch, intentionally weakening their features to prevent a neural network from fitting them first. We show that this strategy can largely balance the training accuracy and feature gradients across classes, effectively mitigating the under-fitting then over-fitting problem for minor class data. On several benchmark datasets, our approach achieves the state-of-the-art accuracy, especially for the challenging step-imbalanced cases.

1. Introduction

Deep neural networks [21, 25, 37, 55, 56] trained with *class-imbalanced data*, of which we have ample data for some "major" classes and limited data for the other "minor" classes, are known to perform poorly on minor classes [3, 45, 61]. As many real-world data sets are class-imbalanced by nature, especially those for recognizing a large number of objects [17, 35, 44, 59, 60], this problem has attracted increasing attention recently in both the machine learning and computer vision communities [5, 9, 30, 58, 52, 66].

Many works have attempted to explain this problem and develop corresponding solutions. Some attribute this to the mismatch between training and test data distributions [5, 7, 20, 32, 29, 64, 57, 26, 49][1] or *under-fitting to minor classes* [40, 41]. Others attribute this to the poor feature learning [12, 13, 22, 23, 77, 19, 70, 79] or *over-fitting to minor classes* [1, 72, 33]. That is, a neural network can *easily* fit the limited amount of minor class data to obtain $\sim 100\%$ training accuracy, but just cannot generalize to the test data. One particular finding made by [72, 33] is the *feature deviation* between training and test data: for minor classes, the training and test features generated by the learned neural network[2] deviate from each other, making the learned decision boundary not applicable to test data.

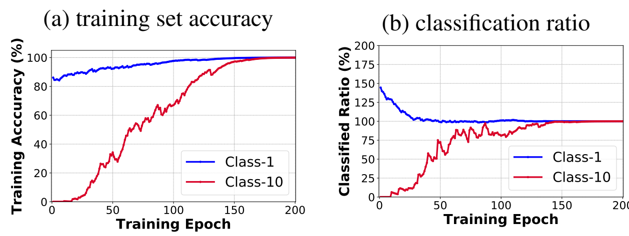

Figure 1. **The *training progress* of a neural network on class-imbalanced data.** We trained a ResNet-32 [21] on a long-tailed CIFAR-10 data set [36]. We only showed two classes for clarity. Class-1 and class-10 have 5000 and 50 training instances, respectively. **(a) training accuracy per class** along the epochs. **(b) classification ratio:** the numbers of training instances classified into a class *divided by* the number of training instances which truly belong to that class, along the epochs. Minor classes need longer time to achieve almost a 100% training accuracy, and most of their instances are classified into the major classes in early epochs.

In this paper, we present a novel finding that links these two lines of explanations. We analyzed the **training progress of a neural network** (*i.e.*, its training accuracy along the epochs; see Figure 1). We found that, while a neural network eventually fits the minor class data, it only happens when the training is about to converge. In fact, in most of the early training process, the neural network classifies most of the minor class training data into major classes, es-

[1]During testing, one usually assumes *class-balanced* test data or computes average per-class accuracy.

[2]Here the features mean the neural network's outputs before the last fully-connected layer.

sentially under-fitting the minor class data. To correct these wrong predictions, the neural network then must focus on pushing features of the minor class data across the decision boundaries between major and minor classes.

We argue that, this initial *under-fitting phase of minor classes* exaggerates the competition between major and minor classes, forcing the neural network to overly learn the discriminative knowledge that cannot be generalized and ultimately leading to feature deviation and over-fitting. Concretely, in training the feature extractor $f_{\boldsymbol{\theta}}(\boldsymbol{x})$[3] using a loss function ℓ, we found that the gradients assigned to features of minor class data (*i.e.*, $\nabla_{f_{\boldsymbol{\theta}}(\boldsymbol{x})}\ell$) are much higher than those for major class data (as will be analyzed in section 3). This finding provides an explanation to feature deviation [33, 72]: the exaggerated gradients push the features of minor class training data further away than where they need to be, thus deviating from the features of the test data.

To address this issue, we propose a novel, simple yet effective learning strategy to alleviate the *under-fitting then over-fitting* problem for minor class data. The core idea is to *equalize* the training progress between major and minor classes by suppressing a neural network's tendency to first fit major class data — hence making it *procrustean across classes*. We achieve this by *weakening* the features of major class data at every mini-batch. Concretely, for a major class datum, we mix its feature with that of another data (*i.e.*, a convex interpolation), such that the resulting features will likely move toward or even across the decision boundary (thus misclassified). We showed that, this learning strategy can effectively balance not only the training accuracy across classes, but also the gradients assigned to features. The resulting neural network therefore suffers less feature deviation and over-fitting for minor classes.

We validate our approach on several benchmark data sets, including CIFAR-10 [36], CIFAR-100 [36], TinyImageNet [38], and iNaturalist [60]. Our approach achieves the state-of-the-art results on many of the experimental settings, *especially for the more challenging step-imbalanced cases*. By analyzing the learned features, we also observe much smaller feature deviations, essentially resolving one fundamental issue in class-imbalanced deep learning.

2. Related Work

There are two mainstream methods of class-imbalanced learning: *re-sampling* based and *cost-sensitive* based.

Re-sampling based methods aims to change the training data distribution to match the balanced test data [14, 3, 61]. Exemplar ways are over-sampling the minor-class data [3, 4, 53] or under-sampling the major-class data [3, 20, 27, 48], directly from the training data. Some methods synthesized additional minor-class data to enlarge the diversity of a class [2, 7, 15, 63]. Some others transferred statistics from the major classes to the minor classes [18, 34, 45, 74]. [8] proposed Remix to synthesize new data by linearly interpolating two real data. Different from mixup [76], Remix used different mixing coefficients for the input data and labels (larger label coefficients for minor classes), essentially generating more data for minor classes. In contrast, we perform linear interpolation only in the input data. Our goal is to balance the training progress, not to augment more data for minor classes.

Cost-sensitive based methods adjust the cost of incorrect predictions according to the true class labels. One popular way is re-weighting, which gives each instance a weight based on its true label when computing the total loss. Setting the weights by (the square roots of) the reciprocal of the number of training instances per class has been widely used [22, 24, 47, 64, 69]. [9] proposed a principled way to set weights by computing the effective numbers of training instances. [6, 28, 51, 54, 68, 26, 49] explored dynamically adjusting the weights via meta-learning or curriculum learning. Instead of adjusting instance weights, [32] developed several instance loss functions that reflect class imbalance; [5, 72] forced minor-class instances to have large additive or multiplicative margins from the decision boundaries. [31] proposed to incorporate uncertainty of instances or classes in the loss function. The closest to ours are [42, 57], which introduced new loss functions to balance the gradients assigned to the last fully-connected layers (*i.e.*, linear classifiers) over classes. In contrast, we aim to balance the gradients assigned to data instances for better feature learning.

Learning feature embeddings with class-imbalanced data has also been studied, especially for face recognition [12, 13, 22, 23, 77]. [19, 70, 79] combined objective functions of classification and embedding learning to better exploit minor class data. [65, 10, 78] proposed two-stage training procedures to pre-train features with imbalanced data and fine-tune the classifier with balanced data; [30] systematically studied different training strategies for each stage. [80] introduced bilateral-branch networks to cumulatively make the two-stage transition. Our work also improves features, by reducing the unfavorable feature deviation [72, 33].

Empirical observations. Similar to ours, several recent works are built upon empirical analysis. [16, 30, 74, 33] found that the learned linear classifiers of a ConvNet tend to have larger norms for major classes, and proposed to force similar norms across classes in training or calibrate the norms in testing. [70] found that the feature norms of major-class and minor-class instances are different and proposed to regularize it by forcing similar norms. [72, 33] both found the phenomenon of feature deviation between the training and test data, especially for minor class data. Our work is different by presenting a novel finding of the network training progress. The observed larger gradients on

[3] $f_{\boldsymbol{\theta}}$ is the part of a neural network before the last fully-connected layer.

minor class data offer an explanation for feature deviation.

3. Approach

In this section, we introduce our approach, which we called Major Feature Weakening (MFW). We begin with the basic notation, followed by our algorithm. We then provide analysis of its properties, especially on how it balances feature gradients and the training progress across classes.

3.1. Background and notation

We denote a C-class neural network classifier by

$$\hat{y} = \arg\max_{c\in\{1,\cdots,C\}} \boldsymbol{w}_c^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}), \quad (1)$$

where $\boldsymbol{x}$ is the input, $f_{\boldsymbol{\theta}}(\cdot)$ is the feature extractor parameterized by $\boldsymbol{\theta}$, and $\{\boldsymbol{w}_c\}_{c=1}^C$ is the final fully-connected layer for linear classification. The feature extractor $f_{\boldsymbol{\theta}}(\cdot)$ can be further decomposed by $h_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\theta}}(\cdot) = h_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta}}(\cdot))$, where the output of $g_{\boldsymbol{\theta}}$ is the intermediate feature.

Given a training set $D_{\text{tr}} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, in which each class c has N_c instances, we normally train the classifier by **empirical risk minimization (ERM)**, using a loss function $\ell(y, \{\boldsymbol{w}_c^\top f_{\boldsymbol{\theta}}(\boldsymbol{x})\}_{c=1}^C)$,

$$\begin{aligned} &\min_{\boldsymbol{\theta}, \{\boldsymbol{w}_c\}_{c=1}^C} \sum_n \ell(y_n, \{\boldsymbol{w}_c^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)\}_{c=1}^C) \\ =& \min_{\boldsymbol{\theta}, \{\boldsymbol{w}_c\}_{c=1}^C} \sum_n \ell(y_n, \{\boldsymbol{w}_c^\top h_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta}}(\boldsymbol{x}_n))\}_{c=1}^C). \end{aligned} \quad (2)$$

One popular loss function is the cross-entropy loss,

$$\begin{aligned} \ell(y, \{\boldsymbol{w}_c^\top f_{\boldsymbol{\theta}}(\boldsymbol{x})\}_{c=1}^C) &= -\log p(y|\boldsymbol{x}; \boldsymbol{\theta}, \{\boldsymbol{w}_c\}_{c=1}^C) \\ &= -\log \frac{\exp(\boldsymbol{w}_y^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{\sum_c \exp(\boldsymbol{w}_c^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}))}. \end{aligned} \quad (3)$$

We apply stochastic gradient descent (SGD) for optimization, with uniformly sampled instances from D_{tr}.

For class-imbalanced learning, each class c will have a different number of training instances N_c.

3.2. Major feature weakening (MFW)

As mentioned in section 1 and Figure 1, a neural network trained with class-imbalanced data tends to fit major classes first, resulting in an inconsistent training progress across classes. To resolve this problem, we propose to weaken the features of major classes within each mini-batch.

Let $(\boldsymbol{x}_1, y_1)$ and $(\boldsymbol{x}_2, y_2)$ be two training data instances in a mini-batch, MFW performs the following operation to the intermediate feature $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$ of $\boldsymbol{x}_1$

$$\begin{aligned} \tilde{\boldsymbol{z}}_1 &= (1-\lambda_1) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_1) + \lambda_1 \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_2), \\ \tilde{y}_1 &= y_1, \end{aligned} \quad (4)$$

Algorithm 1: Major Feature Weakening (MFW): see subsection 3.2 for details.

Input : training data $D_{\text{tr}} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$; initial parameters $\boldsymbol{\theta}$, $\{\boldsymbol{w}_c\}_{c=1}^C$; weight function s; beta distribution coefficient α; batch size B
Model: $f_{\boldsymbol{\theta}} = h_{\boldsymbol{\theta}} \circ g_{\boldsymbol{\theta}}$
while *not converge* **do**
  Sample $D_1 = \{(\boldsymbol{x}_n^{(1)}, y_n^{(1)})\}_{n=1}^B$ from D_{tr}
  Permute D_1 to get $D_2 = \{(\boldsymbol{x}_n^{(2)}, y_n^{(2)})\}_{n=1}^B$
  for $n \in \{1, \cdots, B\}$ **do**
    $\lambda_n \sim \text{Beta}(\alpha, \alpha)$
    $\lambda_n \leftarrow s(N_{y_n^{(1)}}) \times \lambda_n$
    $\tilde{\boldsymbol{z}}_n = (1-\lambda_n) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_n^{(1)}) + \lambda_n \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_n^{(2)})$
    $\tilde{y}_n = y_n^{(1)}$
  end
  Optimize Equation 2 w.r.t. $\boldsymbol{\theta}$ and $\{\boldsymbol{w}_c\}_{c=1}^C$ using $\tilde{D} = \{(\tilde{\boldsymbol{z}}_n, \tilde{y}_n)\}_{n=1}^B$, where $g_{\boldsymbol{\theta}}(\boldsymbol{x}_n)$ in Equation 2 is replaced by $\tilde{\boldsymbol{z}}_n$.
end

which mixes (*i.e.*, convexly interpolates) the intermediate feature $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$ with $g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)$ to become the new intermediate feature $\tilde{\boldsymbol{z}}_1$ of $\boldsymbol{x}_1$. The label of $\boldsymbol{x}_1$ is kept intact. Thus, when $y_2 \neq y_1$, Equation 4 essentially moves $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$ toward features of other classes, hence *weakening* its feature. The resulting $(\tilde{\boldsymbol{z}}_1, \tilde{y}_1)$ is then fed into $h_{\boldsymbol{\theta}}$ to obtain the feature of $\boldsymbol{x}_1$ and calculate the loss.

Here, $\lambda_1 \in [0, 1]$ is sampled from a beta distribution $\text{Beta}(\alpha, \alpha)$ following [76], which is then multiplied with a class-dependent weight $s(N_{y_1})$. The weight function $s(\cdot)$ is monotonically increasing with the class size N_{y_1} and has a range $[0, 0.5]$, which gives major classes a larger weight to weaken their features. That is, $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$ will be weakened more with a larger λ_1 if y_1 is a major class. Nevertheless, the range of s and the support of the beta distribution ensure that $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$ is still the main ingredient of the new intermediate feature $\tilde{\boldsymbol{z}}_1$. Algorithm 1 summarizes the training procedure of MFW. We discuss how to set $s(\cdot)$ in Equation 12.

During evaluation, given a training or test example $\boldsymbol{x}$, we do not perform MFW but extract its feature by $h_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta}}(\boldsymbol{x}))$.

3.3. Why does MFW help imbalanced learning?

We now analyze why MFW can reduce the gradients assigned to features of minor classes and balance the training progress across classes. Without loss of generality, let us consider a binary classification problem, with $c = 1$ as the major class and $c = 0$ as the minor class. The classifier can be simplified as $\hat{y} = \frac{1}{2}[\text{sign}(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x})) + 1]$. The cross-entropy loss for a data instance $(\boldsymbol{x}, y)$ thus becomes

$$\begin{aligned} \ell = &- y \times \log \sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x})) \\ &- (1-y) \times \log(1 - \sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}))), \end{aligned} \quad (5)$$

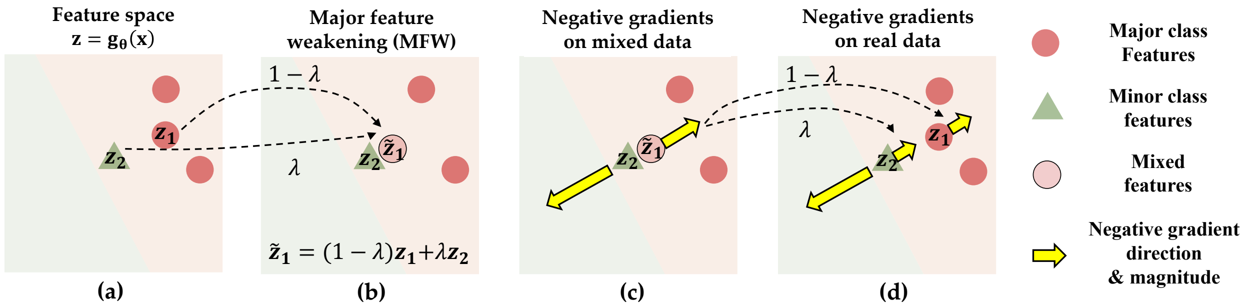

Figure 2. **Illustration of MFW.** (a) intermediate features $\boldsymbol{z} = g_{\boldsymbol{\theta}}(\boldsymbol{x})$; (b) mixed, weakened features $\tilde{\boldsymbol{z}}_1$ for a major class; (c) the gradients $\nabla_{\boldsymbol{z}_2}\ell$ and $\nabla_{\tilde{\boldsymbol{z}}_1}\ell$; (d) the gradients $\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell$ and $\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)}\ell$, where $\boldsymbol{x}_2$ and $\boldsymbol{x}_1$ are the pre-images of $\boldsymbol{z}_2$ and $\boldsymbol{z}_1$. From (c) to (d), $\nabla_{\tilde{\boldsymbol{z}}_1}\ell$ is separated into two parts, one of them will affect $\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell$. With MFW, $\|\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell\|_2$ (*i.e.*, to minor class data) can be reduced.

where $\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x})) = \dfrac{1}{1+\exp(-\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}))}$.

Reducing the gradients. Let $(\boldsymbol{x}_1, y_1 = 1)$ and $(\boldsymbol{x}_2, y_2 = 0)$ be the two data instances in a mini-batch of size two, where $\boldsymbol{x}_1$ is from the major class and $\boldsymbol{x}_2$ is from the minor class. Following Algorithm 1, we construct $(\tilde{\boldsymbol{z}}_1, \tilde{y}_1 = y_1 = 1)$ and $(\tilde{\boldsymbol{z}}_2, \tilde{y}_2 = y_2 = 0)$,

$$\begin{aligned} \tilde{\boldsymbol{z}}_1 &= (1-\lambda_1) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_1) + \lambda_1 \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_2), \\ \tilde{\boldsymbol{z}}_2 &= (1-\lambda_2) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_2) + \lambda_2 \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_1). \end{aligned} \tag{6}$$

Let us first consider $h_{\boldsymbol{\theta}}$ as an identify function: *i.e.*, $f_{\boldsymbol{\theta}} = g_{\boldsymbol{\theta}}$. When MFW is not applied (*i.e.*, $\lambda_1 = \lambda_2 = 0$), we have

$$\begin{aligned} \nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)}\ell &= (\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)) - y_1) \times \boldsymbol{w}, \\ \nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell &= (\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times \boldsymbol{w}, \end{aligned} \tag{7}$$

where misclassified data have larger gradients.

When MFW is applied and we have a weight function that gives $c = 1$ a weight 0.5 and $c = 0$ a weight 0 (so major classes have larger weights), we have $\lambda_1 \in [0, 0.5]$ while $\lambda_2 = 0$. This leads to

$$\begin{aligned} \nabla_{\tilde{\boldsymbol{z}}_1}\ell &= (\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times \boldsymbol{w}, \\ \nabla_{\tilde{\boldsymbol{z}}_2}\ell &= (\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_2) - y_2) \times \boldsymbol{w}, \end{aligned} \tag{8}$$

which, by passing the gradients back to $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$ and $g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)$ according to Equation 6 (note that, we set $\lambda_2 = 0$ already), then gives us

$$\begin{aligned} \nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)}\ell =& (1-\lambda_1) \times (\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times \boldsymbol{w}, \\ \nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell =& (\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times \boldsymbol{w} \\ &+ \lambda_1 \times (\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times \boldsymbol{w}. \end{aligned} \tag{9}$$

The second part of $\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell$ comes from $g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)$ being used to weaken $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$. Figure 2 gives an illustration.

Now suppose $\boldsymbol{x}_2$ is not classified correctly by the current model, *i.e.* $\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) > 0.5$, we have

$$\begin{aligned} &|(\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2)| \geq \\ &|(\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) + \lambda_1 \times (\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_1) - y_1)| \geq 0, \end{aligned}$$

which means the norm of $\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell$ will be reduced by MFW[4] in comparison to Equation 7.

Balancing the training progress. We now analyze the gradient w.r.t. the linear classifier $\boldsymbol{w}$. Without MFW, it is

$$\begin{aligned} \nabla_{\boldsymbol{w}}\ell =& (\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)) - y_1) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_1) \\ &+ (\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_2). \end{aligned} \tag{10}$$

With MFW (but $\lambda_2 = 0$), the gradient w.r.t. $\boldsymbol{w}$ becomes

$$\begin{aligned} \nabla_{\boldsymbol{w}}\ell =& (\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times \tilde{\boldsymbol{z}}_1 \\ &+ (\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_2) \\ =& (\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times ((1-\lambda) g_{\boldsymbol{\theta}}(\boldsymbol{x}_1) + \lambda g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) \\ &+ (\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_2). \end{aligned} \tag{11}$$

By comparing the first term in Equation 10 and Equation 11, MFW reduces the tendency of $\boldsymbol{w}$ to fit major class data[5]. In other words, besides weakening the features of major class training data, MFW also weakens their classifiers. Both can essentially balance the training progress across classes.

Further discussions. The gradient reduction for $g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)$ is governed by λ_1 that is affected by the class size of y_1. In theory, most of the data in a mini-batch come from major classes. Thus, very likely a minor class datum will be paired with a major class datum to get its gradient reduced.

For other $h_{\boldsymbol{\theta}}$, and for $\lambda_2 \neq 0$. When $h_{\boldsymbol{\theta}}$ is a linear mapping $\boldsymbol{V}$, the above conclusions still hold: the only difference is that $\boldsymbol{V}$ will be multiplied into the gradients. When $h_{\boldsymbol{\theta}}$ is a non-linear function, Jacobian matrices are needed for analysis. See the supplementary material for details. In practice, we found that even with a complex $h_{\boldsymbol{\theta}}$ (*e.g.*, several residual network blocks [21]), MFW still effectively improves class-imbalanced learning. We note that when $\lambda_2 \neq 0$, $\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)}\ell$ will be affected by $\lambda_2 \times (-\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times \boldsymbol{w}$.

3.4. Balanced training progress and gradients

We apply MFW to the same problem as in Figure 1. We train a classifier using ResNet-32 [21] on a long-tailed

[4] One can show this by plugging in $y_1 = 1$ and $y_2 = 0$, and consider $(\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - 0) > 0.5$ and $\lambda_1 \times (\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_1) - 1) \in [-0.5, 0.0]$.

[5] The 1st term moves $\boldsymbol{w}$ toward $(1-\lambda) g_{\boldsymbol{\theta}}(\boldsymbol{x}_1) + \lambda g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)$, not $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$.

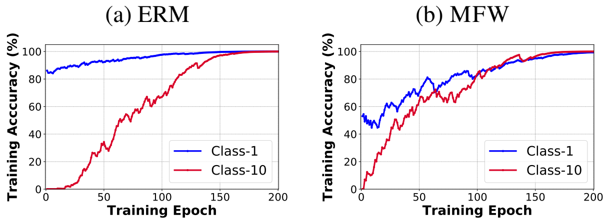

Figure 3. **Training set accuracy by learning with ERM (a) and MFW (b).** MFW makes the training progress of the major (*i.e.*, Class-1) and minor (*i.e.*, Class-10) classes more procrustean.

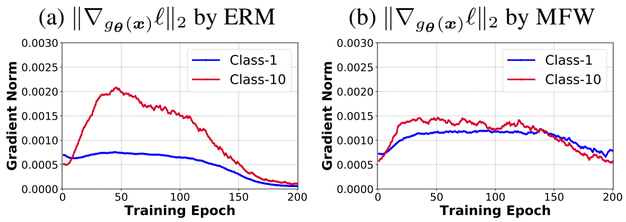

Figure 4. **Feature gradient norm $\|\nabla_{g_{\theta}(x)}\ell\|$ by learning with ERM (a) and MFW (b).** We show $\|\nabla_{g_{\theta}(x)}\ell\|$ after every training epoch, averaged over samples of each class. Through MFW, the gradient norm of the minor class (*i.e.*, Class-10) features decreases; gradient norms across classes are more balanced.

CIFAR-10 data set [36]. The most major class $c = 1$ has $5,000$ training examples while the most minor class $c = 10$ has 50 examples. We apply MFW after the second group of convolutional layers of the ResNet, using a sigmoid-shaped weight function $s(\cdot)$ such that $s(N_1) \approx 0.5$ and $s(N_{10}) \approx 0$ (see Equation 12). Please see section 5 for more details.

Figure 3 shows the training set accuracy: we evaluate this after each epoch, without altering the features. We include only the two extreme classes for clarity. With MFW, the accuracy is more balanced across classes. By comparing the gradient norm $\|\nabla_{g_{\theta}(x)}\ell\|$ averaged over samples per class in Figure 4, we see that learning without MFW (a) has a larger gap of gradient norms between classes, whereas learning with MFW (b) reduces the gap notably.

We note that, MFW reduces the gradients given the same network parameters. This does not imply that MFW has a smaller gradient norm of minor classes than ERM throughout the entire training process. Indeed, as will be seen in section 4, MFW has the effect of keeping samples not too far away from the decision boundary to prevent over-fitting. This means that the training loss at the final training epochs will be larger than ERM, leading to slightly larger gradients.

3.5. Comparison to mixup and Remix

Our Equation 4 is reminiscent of mixup [76] and [62], but with a notable difference: we do not mix the label. Thus, our work does not intend to regularize the neural network to favor a simple linear behavior between training examples.

A recent work Remix [8] proposed to use mixup for imbalanced learning by allowing the mixing coefficients of data and labels to be disentangled. Specifically, a higher label mixing coefficient is assigned to minor classes. Their method thus can be seen as re-sampling or data augmentation for minor classes: increasing the minor class examples with linearly interpolated data. In contrast, MFW does not change the class distribution and hence is not a re-sampling method. Besides, for minor class data, MFW tends to perform no mixing (no feature weakening). Therefore, MFW is hardly a data augmentation method for minor classes, but an effective and mathematically sound way to balance the training progress and gradient norms across classes.

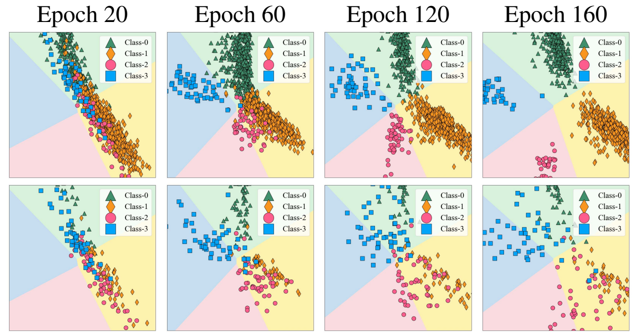

Figure 5. **The training (top) and test (bottom) feature distributions along the training process, using ERM.** We study a four class imbalanced task, with different classes denoted by different colors/shapes (Class-0, Class-1 are major classes). There is a clear feature deviation (hence over-fitting) between training and testing.

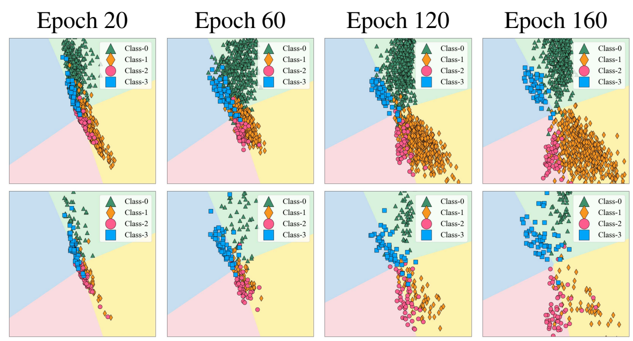

Figure 6. **The training (top) and test (bottom) feature distributions along the training process, using MFW.** We study a four class imbalanced task, with different classes denoted by different colors/shapes (Class-0, Class-1 are major classes). The feature deviation (over-fitting) between training and testing is reduced.

4. Illustrative experiments

To showcase the effect of MFW, we conduct another experiment. We select four classes from CIFAR-10 [36], and make their training data to be $5000, 5000, 50, 50$ per class: *i.e.*, two major and two minor classes. The test data per class are $1,000$ samples. We use a ResNet-32 [21] but add an additional linear projection layer to make the final feature dimension (*i.e.*, right before the last fully-connected layer) to be 2 for visualization. We then train the ResNet-32 with MFW or with ERM, using the cross-entropy loss (cf. Equation 3) for 200 epochs. The initial learning rate is 0.1 and it is decreased based on the cosine annealing rule.

For MFW, we mix the intermediate features after the second group of convolutional layers. We set $s = 0.5$ for major classes and $s = 0$ for minor classes, and set $\alpha = 2.0$ for the beta distribution. In other words, the minor class features will not be weakened. After each training epoch, we plot the final two-dimensional features and the decision boundaries for both training and test data. Due to the page limit, only the results of 20, 60, 120, and 60 epochs are shown in Figure 5 (for ERM) and Figure 6 (for MFW). We also sub-sample data in the figures to make it less crowded.

From Figure 5 (for ERM), we see that at epoch 20, the training data (top row) of the minor classes are 100% misclassified into the major classes. It thus leads to large gradients that try to push the minor class data away into their own territories. At the end of training, we see a nearly perfect separation for the training data. This is, however, not the case for the test data (bottom row). Specifically, at epoch 160, most of the test features of the minor classes are close to the boundaries between major and minor classes or even wrongly classified, essentially a case of *feature deviations*.

Let us now look at Figure 6 (for MFW). There are four notable differences from Figure 5. First, at epoch 60, most of the minor class training data are correctly classified (or close to be); the major class features are kept close to the boundaries. The training progress is thus more equalized. Second, with the *gradient reduction* by MFW, the competition between the major and minor classes are reduced. Even at epoch 160, the minor class training data are not overly pushed away from the boundaries. Third, the feature distributions of the training and testing data are much closer, indicating a smaller feature deviation. Finally, compared to Figure 5, a larger portion of the minor class test data are correctly classified at the end. As a result, the final model by MFW outperforms that by ERM (83.65% vs. 78.30%).

5. Experiment

5.1. Setup

Datasets. We validate MFW on five datasets. **CIFAR-10** and **CIFAR-100** [36] are for image classification with 32×32 images. There are $50,000$ training and $10,000$ test images from 10 and 100 classes, respectively. **Tiny-ImageNet** [38] has 200 classes. Each class has 500 training and 50 validation images of 64×64 pixels. **iNaturalist** [60] (2018 version) is a natural large-scale long-tailed dataset containing 437,513 training images from 8,142 classes and there are 3 validation images per class. The image resolution is 224×224. **Tiered-ImageNet** [50] is a subset of ImageNet [11] widely used in few-shot learning. The image resolution is 84×84. We use Tiered-ImageNet to synthesize a large-scale step imbalanced dataset. We treat the 351 many-shot classes as the major classes, each with around $1,000$ training images, and the 160 few-shot classes as the minor classes, each with 5 training instances. All the classes have 50 test instances. See the supplementary for details.

Setup. We follow [5, 9] to create *imbalanced* CIFAR-10, CIFAR-100, and Tiny-ImageNet with different imbalance ratios $\rho = N_{\text{max}}/N_{\text{min}} \in \{10, 100, 200\}$. Two types of imbalanced data are investigated, *i.e.*, the *long-tailed* (LT) imbalance where the number of training instances exponentially decayed per class, and the *step* imbalance where the size of the second half of classes are proportional to a fixed ratio w.r.t. the first half head classes. The test and validation sets remain unchanged and balanced. We re-index classes so that smaller indices have more training instances.

We follow existing works [5, 9, 80, 39] to use ResNet [21]: ResNet-32 for CIFAR, ResNet-18 for Tiny-ImageNet, ResNet-12 for Tiered-ImageNet, and ResNet-50 for iNaturalist. We adapt the code from [5, 9] for CIFAR, from [39] for Tiered-ImageNet, and from [80] for iNaturalist. Please see the supplementary material for details.

Following [9, 5, 80], the test set accuracy (validation set accuracy on Tiny-ImageNet and iNaturalist) are reported for evaluation. See the supplementary material for details.

5.2. Implementation of MFW

We apply the cross-entropy loss in Equation 3 to train ResNet with MFW. Considering a residual block and the convolutional layers before the first residual block each as a group of convolutional layers, we apply MFW after the second groups of convolutions, unless stated otherwise.

We design the weight function s in Algorithm 1 as follows. First, the weight should be monotonically increasing from minor to major classes. Second, major and minor classes get weights around 0.5 (more weakening) and 0.0 (no weakening), respectively. To take the number of instances per class N_c into account, we define s as follows

$$s(N_c) = 0.5 \times \sigma(\frac{N_c - \mu}{\beta \cdot \gamma}), \qquad (12)$$

where $\sigma(a) = \frac{1}{1+\exp(-a)}$ is the sigmoid function, which is widely used to squash real values into the range $[0, 1]$. μ and γ are the geometric mean and standard deviation over $\{N_c\}_{c=1}^{C}$, which normalize the weights with respect to all classes[6]. We use the geometric mean because it is less sensitive to extremely large N_c than the arithmetic mean, making the weight stable in both long-tailed and step settings. The scale β controls the softness of weights. We set $\beta = 2.0$ for the long-tailed case and $\beta = 0.01$ for the step case. We tune the beta distribution coefficient α on a small held-out set from the training data. Details are in the supplementary.

[6] Subtracting the mean and dividing by the standard deviation are common practices to normalize function inputs (*e.g.*, z-score).

Table 1. Test set accuracy (%) on imbalanced CIFAR-10/-100. The best result of each setting (column) is in bold font.

	CIFAR-10						CIFAR-100					
	Long-tailed			Step			Long-tailed			Step		
Imbalance ratio ρ	200	100	10	200	100	10	200	100	10	200	100	10
ERM [9]	65.6	71.1	87.2	60.0	65.3	85.1	35.9	40.1	56.9	38.7	39.9	54.6
Focal [43]	65.3	70.4	86.8	-	63.9	83.6	35.6	38.7	55.8	-	38.6	53.5
CB [9]	68.9	74.6	87.5	-	61.9	84.6	36.2	39.6	58.0	-	33.8	53.1
LDAM-DRW [5]	74.6	77.0	88.2	73.6	76.9	87.8	39.5	42.0	58.7	42.4	45.4	59.5
τ-Norm [30]	70.3	75.1	87.8	68.8	73.0	87.3	39.3	43.6	57.4	**43.2**	45.2	57.7
CDT [72]	74.7	79.4	89.4	70.3	76.5	88.8	40.5	44.3	58.9	40.0	47.0	59.6
BBN [80]	-	79.8	88.3	-	-		-	42.6	59.1	-	-	-
M2M [34]	-	79.1	87.5	-	-	-		43.5	57.6	-	-	-
Meta-Weight [54]	67.2	73.6	87.6	-	-		36.6	41.6	58.9	-	-	-
DA [26]	70.7	76.4	88.9	-	-	-	39.3	43.4	59.6	-	-	-
Remix-DRW [8]	-	79.8	89.0	-	77.9	88.3	-	**46.8**	**61.2**	-	46.8	60.4
De-confound-TDE [58]	-	**80.6**	88.5	-	-	-	-	44.1	59.6	-	-	-
MFW	73.2	78.5	**89.8**	75.4	80.1	89.6	40.7	44.7	60.1	42.5	46.9	61.2
MFW w/ DRW	**75.0**	79.8	89.7	**78.8**	**81.6**	**89.9**	**41.4**	46.0	59.1	43.0	**48.4**	**61.6**

Table 2. Top-1/-5 validation set accuracy (%) on imbalanced Tiny-ImageNet. The best result of each setting (column) is in bold font.

	Long-tailed				Step			
Imbalance ratio ρ	100 Top-1	100 Top-5	10 Top-1	10 Top-5	100 Top-1	100 Top-5	10 Top-1	10 Top-5
ERM [5]	33.8	57.4	49.7	73.3	36.2	55.9	49.1	72.9
CB [10]	27.3	47.4	48.4	71.1	25.1	40.9	45.5	66.8
LDAM-DRW [5]	37.5	60.9	**52.8**	**76.2**	39.4	61.9	52.6	**76.7**
τ-Norm [30]	36.4	59.8	49.6	72.8	40.0	61.9	51.7	75.2
CDT [72]	**37.9**	**61.4**	52.7	75.6	39.6	61.5	53.3	76.2
MFW	35.4	59.2	51.0	73.4	**40.4**	**62.9**	52.9	76.3
MFW w/ DRW	36.2	59.8	**52.8**	74.5	40.0	61.2	**54.3**	**76.7**

Inspired by [5, 8], we also apply Deferred Re-Weight (DRW) after training 80% of epochs. DRW applies an instance-specific weights over the loss function, which stresses the optimization for minor classes. We follows the same weighting strategy as [5].

5.3. Results

CIFAR. We extensively examine CIFAR-10 and CIFAR-100 with imbalance ratios $\rho \in \{10, 100, 200\}$ on both long-tailed and step cases. Results are shown in Table 1. MFW (without DRW) is on a par with or better than the compared approaches. With DRW, MFW achieves the best performance in most of the settings.

It is worth noting that MFW *obtains particularly high accuracy on the* ***step imbalance scenario*** *that most of the other methods struggled.* In contrast to the long-tailed setting where there is a smooth transition of class sizes between the major and minor classes, the step setting only has two extremes (*i.e.*, major or minor classes), whose sizes are different by a ratio ρ. We attribute the superior performance of MFW to its inner working, which aims to resolve the *over-competition between the major and minor classes*.

Table 3. Top-1/-5 test accuracy (%) on Step Tiered-ImageNet.

	Top1	Top5
ERM	41.7	58.1
LDAM [5]	42.3	63.1
CDT [72]	43.8	64.2
cRT [30]	44.4	66.5
τ-norm [30]	43.0	62.4
LWS [30]	42.6	58.8
Remix [8]	43.4	62.6
MFW	46.1	67.5
MFW w/ DRW	**46.4**	**67.8**

Such a competition is stronger in the step case than in the long-tailed case, and that is why MFW excels.

Tiny-ImageNet. (Table 2) Our MFW again performs particularly well in the challenging step setting.

Step Tiered-ImageNet. (Table 3) Tiered-ImageNet is a large-scale step imbalanced dataset (511 classes) whose imbalanced ratio $\rho > 200$. We re-implement all the baselines. Our MFW outperforms all methods, which demonstrate the strong ability of MFW on step imbalanced cases.

iNaturalist. (Table 4) iNaturalist has 8,142 classes and

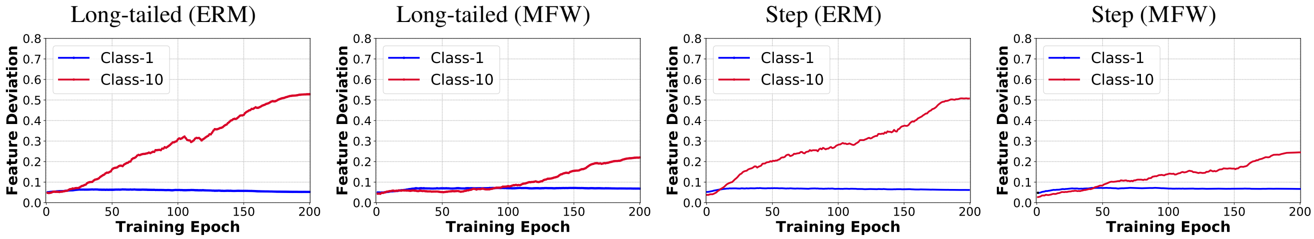

Figure 7. **Feature deviation between the training and test data per class along the training progress.** We experiment on CIFAR-10, using both the long-tailed and the step settings ($\rho = 100$). We only showed the most major ($c = 1$, with $5,000$ training samples) and minor classes ($c = 10$, with 50 training samples) for clarity. The formulation of deviation is in subsection 5.4. As the number of training epochs increases, the deviation increases, while MFW can achieve a much smaller deviation.

Table 4. Top-1/-5 validation accuracy (%) on iNaturalist. We present results by training with 90/180 epochs in the form "A/B".

	Top1	Top5
ERM	58.8 / 64.3	80.1 / 84.5
CB [9]	61.5 / -	80.9 / -
LDAM [5]	64.6 / 66.1	83.5 / -
LDAM-DRW [5]	68.0 / 68.6	85.2 / 85.3
CDT [72]	63.7 / 69.5	82.5 / 86.8
BBN [80]	66.3 / 69.6	-
cRT [30]	65.2 / 67.6	-
τ-norm [30]	65.6 / 69.3	-
Remix-DRW [8]	- / 70.5	- / 87.3
MFW	65.5 / 67.3	85.3 / 85.8
MFW w/ DRW	66.7 / 69.6	85.5 / 86.1

many of them have scarce instances, making it particularly challenging. Our MFW outperforms nearly all but the Remix approach (worse by $\sim 1\%$). As large-scale training is sensitive to batch size, the fact that Remix uses a batch size of 256 while we use 128 (due to the computational constraint) might contribute to the difference.

5.4. Ablation Studies

We conduct further analysis on CIFAR-10 (for $\rho = 100$).

Can MFW reduce the feature deviation? We follow [72] to compute the feature deviation. We extract ℓ_2-normalized features, compute feature means in training and test data for each class c, and calculate their Euclidean distance $dis(c) =$

$$\frac{1}{R}\sum_{r=1}^{R} \|\text{mean}(S_K(\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\text{train}}^{(c)})\})) - \text{mean}(\{f_{\boldsymbol{\theta}}(\boldsymbol{x}_{\text{test}}^{(c)})\})\|_2.$$

S_K is a subsampling (of K examples) over training samples per class before computing the mean, which is to alleviate the estimation variance resulting from different class sizes. We follow [72] to perform $R = 1,000$ subsampling rounds and set K as the minor class size. The larger the $dis(c)$ is, the larger the feature deviation is, which implies severer over-fitting (thus worse accuracy). Figure 7 shows the result. MFW notably reduces the feature deviation, justifying our claims that feature deviation is likely caused by the exaggerated feature gradients on the minor classes.

Table 5. Test accuracy (%) on CIFAR-10/-100. We apply MFW after each convolutional group and compare to [76, 62].

	CIFAR-10		CIFAR-100	
Imbalance ratio $\rho = 100$	Long-tailed	Step	Long-tailed	Step
ERM	71.1	65.8	40.1	39.9
MFW (Input layer)	76.3	68.1	41.2	43.1
MFW (1st Group)	77.3	72.4	42.3	44.3
MFW (2nd Group)	**78.5**	**80.1**	**44.7**	**46.9**
MFW (3rd Group)	77.1	69.8	43.1	46.3
MFW (4th Group)	75.0	67.4	40.3	42.2

Which layer for MFW? (Table 5) We apply MFW after different groups of convolutions. Intermediate layers yield the highest accuracy. All of them outperform ERM.

Label mixing. We compare MFW to mixup [76] and manifold mixup [62]. Details are in the supplementary.

6. Conclusion

Class-imbalanced deep learning is a fundamental problem and a practical issue to resolve in computer vision. In this paper, we take a new perspective to tackle it, which is to investigate how imbalanced data affect the training progress of a neural network. Namely, how major classes and minor classes are fitted along the epochs. We found a huge discrepancy: a network tends to fit the major classes first and then the minor ones, resulting in large gradients for the minor class data that may eventually lead to feature deviation and over-fitting. We propose a fairly simple yet mathematically sound approach MFW to effectively balance the training progress and reduce the minor class gradients. MFW performs well on multiple benchmark datasets, especially on the challenging step imbalanced scenario.

Acknowledgement. The research was supported by National Key R&D Program of China (2020AAA0109401) and NSFC (61773198, 61921006, 62006112), NSF of Jiangsu Province (BK20200313). We are thankful for the generous support by Ohio Supercomputer Center and AWS Cloud Credits for Research.

References

[1] Mustakim Al Helal, Mohammad Salman Haydar, and Seraj Al Mahmud Mostafa. Algorithms efficiency measurement on imbalanced data using geometric mean and cross validation. In *International Workshop on Computational Intelligence*, 2016. 1

[2] Anonymous. Imbalanced classification via adversarial minority over-sampling. *OpenReview.net*, 2019. 2

[3] 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. 1, 2

[4] Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *ICML*, 2019. 2

[5] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019. 1, 2, 6, 7, 8, 14, 16, 17

[6] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *NIPS*, 2017. 2

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *JAIR*, 16:321–357, 2002. 1, 2

[8] Hsin-Ping Chou, Shih-Chieh Chang, Jia-Yu Pan, Wei Wei, and Da-Cheng Juan. Remix: Rebalanced mixup. 2020. 2, 5, 7, 8, 15, 17

[9] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019. 1, 2, 6, 7, 8, 12, 14, 16

[10] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 2018. 2, 7

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6, 14

[12] Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *ICCV*, 2017. 1, 2

[13] Qi Dong, Shaogang Gong, and Xiatian Zhu. Imbalanced deep learning by minority class incremental rectification. *TPAMI*, 41(6):1367–1381, 2018. 1, 2

[14] 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, pages 1–8, 2003. 2

[15] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *JAIR*, 61:863–905, 2018. 2

[16] Yandong Guo and Lei Zhang. One-shot face recognition by promoting underrepresented classes. *CoRR*, abs/1707.05574, 2017. 2

[17] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *ECCV*, 2016. 1

[18] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017. 2

[19] Munawar Hayat, Salman Khan, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Gaussian affinity for max-margin class imbalanced learning. In *ICCV*, 2019. 1, 2

[20] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *TKDE*, 21(9):1263–1284, 2009. 1, 2

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4, 5, 6, 12, 14, 15, 16

[22] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016. 1, 2

[23] Chen Huang, Yining Li, Change Loy Chen, and Xiaoou Tang. Deep imbalanced learning for face recognition and attribute prediction. *TPAMI*, 42(11):2781–2794, 2020. 1, 2

[24] Chen Huang, Chen Change Loy, and Xiaoou Tang. Discriminative sparse neighbor approximation for imbalanced learning. *TNNLS*, 29(5):1503–1513, 2017. 2

[25] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1, 17

[26] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *CVPR*, 2020. 1, 2, 7, 17

[27] Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In *International Conference on Artificial Intelligence*, 2000. 2

[28] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018. 2

[29] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, 2019. 1

[30] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *ICLR*, 2020. 1, 2, 7, 8, 14, 17

[31] Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *CVPR*, 2019. 2

[32] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *TNNLS*, 29(8):3573–3587, 2017. 1, 2

[33] Byungju Kim and Junmo Kim. Adjusting decision boundary for class imbalanced learning. *IEEE Access*, 8:81674–81685, 2020. 1, 2

[34] Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. M2m: Imbalanced classification via major-to-minor translation. In *CVPR*, 2020. 2, 7, 17

[35] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome:

Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73, 2017. 1

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 1, 2, 5, 6, 12, 14, 16

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[38] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *Stanford CS 231N*, 2015. 2, 6, 14

[39] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019. 6, 14

[40] Jinyan Li, Simon Fong, Sabah Mohammed, Jinan Fiaidhi, Qian Chen, and Zhen Tan. Solving the under-fitting problem for decision tree algorithms by incremental swarm optimization in rare-event healthcare classification. *Journal of Medical Imaging and Health Informatics*, 6(4):1102–1110, 2016. 1

[41] Jinyan Li, Simon Fong, Raymond K Wong, and Victor W Chu. Adaptive multi-objective swarm fusion for imbalanced data classification. *Information Fusion*, 39:1–24, 2018. 1

[42] Yu Li, Tao Wang, Bingyi Kang, Sheng Tang, Chunfeng Wang, Jintao Li, and Jiashi Feng. Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In *CVPR*, 2020. 2

[43] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 7

[44] 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 *ECCV*, 2014. 1

[45] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019. 1, 2, 14, 17

[46] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008. 16

[47] 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 *ECCV*, 2018. 2

[48] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *CVPR*, 2016. 2

[49] Jiawei Ren, Cunjun Yu, Shunan Sheng, Xiao Ma, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Balanced meta-softmax for long-tailed visual recognition. In *NeurIPS*, 2020. 1, 2

[50] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018. 6, 14

[51] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018. 2

[52] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *CoRR*, abs/1911.08731, 2019. 1

[53] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 2

[54] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019. 2, 7

[55] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[57] Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *CVPR*, 2020. 1, 2, 17

[58] Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. In *NeurIPS*, 2020. 1, 7, 17

[59] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 1

[60] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 1, 2, 6

[61] Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. *CoRR*, abs/1709.01450, 2017. 1, 2

[62] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019. 5, 8, 16

[63] Qingfeng Wang, Xuehai Zhou, Chao Wang, Zhiqin Liu, Jun Huang, Ying Zhou, Changlong Li, Hang Zhuang, and Jie-Zhi Cheng. Wgan-based synthetic minority over-sampling technique: Improving semantic fine-grained classification for lung nodules in ct images. *IEEE Access*, 7:18450–18463, 2019. 2

[64] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. Training deep neural networks on imbalanced data sets. In *IJCNN*, 2016. 1, 2

[65] Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. In *ICML*, 2020. 2

[66] Xudong Wang, Long Lian, Zhongqi Miao, et al. Long-tailed recognition by routing diverse distribution-aware experts. In *ICLR*, 2021. 1

[67] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *CoRR*, abs/1911.04623, 2019. 14

[68] Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, and Junjie Yan. Dynamic curriculum learning for imbalanced data classification. In *ICCV*, 2019. 2

[69] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *NIPS*, 2017. 2

[70] Yue Wu, Hongfu Liu, Jun Li, and Yun Fu. Deep face recognition with center invariant loss. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, 2017. 1, 2

[71] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 14

[72] Han-Jia Ye, Hong-You Chen, De-Chuan Zhan, and Wei-Lun Chao. Identifying and compensating for feature deviation in imbalanced deep learning. *CoRR*, abs/2001.01385, 2020. 1, 2, 7, 8, 13, 14, 16, 17

[73] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*, 2020. 14

[74] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for face recognition with under-represented data. In *CVPR*, 2019. 2

[75] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021. 12

[76] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 2, 3, 5, 8, 15, 16

[77] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *ICCV*, 2017. 1, 2

[78] Yubo Zhang, Pavel Tokmakov, Martial Hebert, and Cordelia Schmid. A study on action detection in the wild. *CoRR*, abs/1904.12993, 2019. 2

[79] Yaoyao Zhong, Weihong Deng, Mei Wang, Jiani Hu, Jianteng Peng, Xunqiang Tao, and Yaohai Huang. Unequal-training for deep face recognition with long-tailed noisy data. In *CVPR*, 2019. 1, 2

[80] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. BBN: bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, 2020. 2, 6, 7, 8, 14, 17