Supplementary Material: 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

We provide details omitted in the main paper.

- Appendix A: details of the statistics in Figure 1 and Figure 7 (cf. section 1 and subsection 5.4 of the main paper).
- Appendix B: additional analysis of MFW (cf. subsection 3.3 of the main paper).
- Appendix C: additional details of the experimental setups (cf. subsection 5.1 and subsection 5.2 of the main paper).
- Appendix D: additional experimental results and analysis (cf. subsection 5.3 and subsection 5.4 of the main paper).

A. Details of the statistics of MFW

A.1. Classification ratio

We showed in Figure 1 (b) of the main paper the *classification ratio* per class, which is the ratio of "the number of *training* data that are classified into a class" (*e.g.*, class-1 or class-10) to "the number of *training* data which truly belong to that class." For instance, if class-10 has 50 training instances but there are 100 training instances (from all the classes) classified as class-10, then the ratio is $100/50 = 200$ (%). Concretely, we apply the learned models (at different epochs) to classify all training data, and compute the ratio of class c as follows

$$\frac{\sum_{n=1}^{N} \mathbf{1}\left[c = \arg\max_{c'} \boldsymbol{w}_{c'}^{\top} f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)\right]}{N_c}. \tag{13}$$

$\mathbf{1}[x]$ returns 1 if x is true and 0 otherwise. This ratio offers a measure of the training progress of each class. If more training data are classified into a particular class than the data that class truly has, the ratio would be high, indicating that the class has been fitted more than other classes.

In Figure 8, we compare the classification ratio of a minor and a major class using either a conventionally-trained neural network (using ERM; top) or the one trained with our MFW (bottom). No matter which method is used, the two classes attain nearly 100% ratio at the end of the training, which matches the fact that a neural network could ultimately fit the training data perfectly [75].

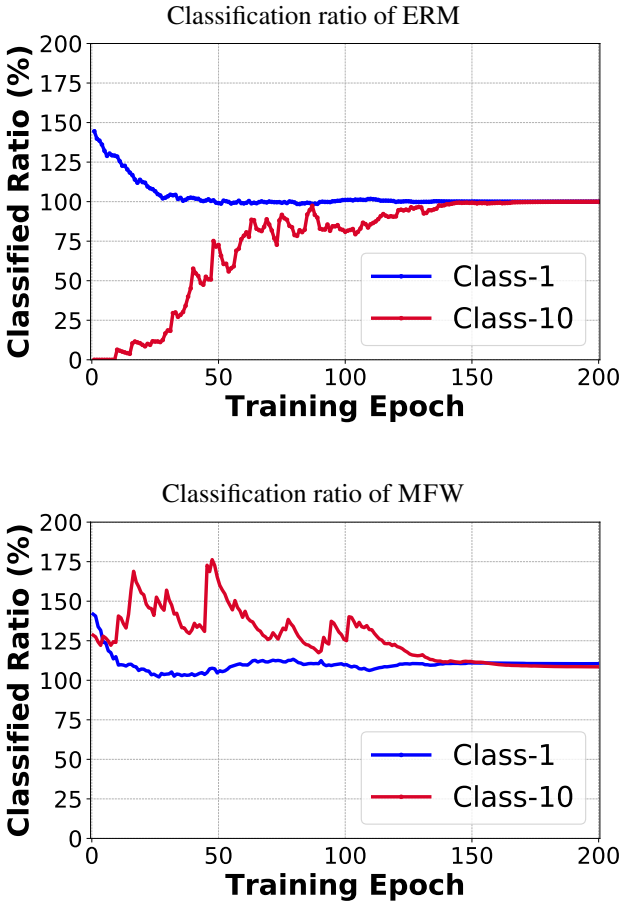

Figure 8. **The training progress of a neural network on class-imbalanced data.** We trained a ConvNet classifier using ResNet-32 [21] on a long-tailed CIFAR-10 data set [36], following [9]. We only showed two classes for clarity. Class $c = 1$ and $c = 10$ have 5000 and 50 training instances, respectively. The classification ratio, *i.e.*, the numbers of training data that are classified into a class divided by the number of training data that class truly has, is plotted along the epochs for both ERM (top) and MFW (bottom).

However, if we compare the ratios along the training process, we see that using ERM, the major class has a much

higher ratio (over 100%) in the beginning while the minor class has a much smaller ratio (around 0% in the beginning). This means that most of the training data are classified into major classes at the early epochs. In contrast, with MFW, the ratios become much balanced across classes, indicating a more balanced training progress.

A.2. Feature deviation

We showed the formula of computing the feature deviation per class between its training and test data in subsection 5.4 of the main paper. We followed [72] to perform multiple rounds of sub-sampling to compute the training mean. This is to reduce the influence of the estimation variance resulting from class sizes — even for the same major class, computing its mean using fewer data has a larger variance and therefore a larger deviation. To remove such a factor and to more faithfully reflect feature deviation across classes, we sub-sample the same number of training data per class to compute its training mean. Similar to [72], we observed a trend of increasing deviation from major to minor classes (in Figure 7 of the main paper), yet MFW leads to a much smaller deviation than ERM.

B. Analysis of MFW

B.1. $h_{\boldsymbol{\theta}}$ is linear

Following subsection 3.3 of the main paper, if $h_{\boldsymbol{\theta}}$ is linear, we have $f_{\boldsymbol{\theta}} = \boldsymbol{V}^\top g_{\boldsymbol{\theta}}$. When MFW is not performed (*i.e.*, $\lambda_1 = \lambda_2 = 0$), we have,

$$\begin{aligned}\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)}\ell &= (\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_1)) - y_1) \times \boldsymbol{V}\boldsymbol{w},\\ \nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell &= (\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times \boldsymbol{V}\boldsymbol{w}. \end{aligned} \tag{14}$$

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

$$\begin{aligned}\nabla_{\tilde{\boldsymbol{z}}_1}\ell &= (\sigma(\boldsymbol{w}^\top \boldsymbol{V}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times \boldsymbol{V}\boldsymbol{w},\\ \nabla_{\tilde{\boldsymbol{z}}_2}\ell &= (\sigma(\boldsymbol{w}^\top \boldsymbol{V}^\top \tilde{\boldsymbol{z}}_2) - y_2) \times \boldsymbol{V}\boldsymbol{w}, \end{aligned} \tag{15}$$

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

$$\begin{aligned}\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)}\ell =&(1-\lambda_1) \times (\sigma(\boldsymbol{w}^\top \boldsymbol{V}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times \boldsymbol{V}\boldsymbol{w},\\ \nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell =&(\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times \boldsymbol{V}\boldsymbol{w}\\ &+ \lambda_1 \times (\sigma(\boldsymbol{w}^\top \boldsymbol{V}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times \boldsymbol{V}\boldsymbol{w}. \end{aligned} \tag{16}$$

The second part of $\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell$ comes from $g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)$ being used to weaken $g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$. Now suppose $\boldsymbol{x}_2$ is not classified correctly by the current model, *i.e.* $\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) > 0.5$, we have

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

which means the norm of $\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell$ will be reduced with MFW[7], compared to Equation 14.

B.2. $h_{\boldsymbol{\theta}}$ is nonlinear

Let us define $\boldsymbol{z}_1 = g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$ and $\boldsymbol{z}_2 = g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)$. When $h_{\boldsymbol{\theta}}$ is nonlinear (*e.g.*, by a neural network block), Equation 16 becomes

$$\begin{aligned}\nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_1)}\ell =&(1-\lambda_1) \times (\sigma(\boldsymbol{w}^\top h_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_1)) - y_1) \times (\nabla_{\tilde{\boldsymbol{z}}_1}\boldsymbol{J})^\top \boldsymbol{w},\\ \nabla_{g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)}\ell =&(\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times (\nabla_{\boldsymbol{z}_2}\boldsymbol{J})^\top \boldsymbol{w}\\ &+ \lambda_1 \times (\sigma(\boldsymbol{w}^\top h_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_1)) - y_1) \times (\nabla_{\tilde{\boldsymbol{z}}_1}\boldsymbol{J})^\top \boldsymbol{w}, \end{aligned} \tag{17}$$

where $\nabla_{\tilde{\boldsymbol{z}}_1}\boldsymbol{J}$ is the Jacobian matrix of $h_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_1)$ w.r.t. $\tilde{\boldsymbol{z}}_1$ and $\nabla_{\boldsymbol{z}_2}\boldsymbol{J}$ is the Jacobian matrix of $h_{\boldsymbol{\theta}}(\boldsymbol{z}_2)$ w.r.t. $\boldsymbol{z}_2$. Suppose that $(\nabla_{\boldsymbol{z}_2}\boldsymbol{J})^\top \boldsymbol{w}$ and $(\nabla_{\tilde{\boldsymbol{z}}_1}\boldsymbol{J})^\top \boldsymbol{w}$ are pointing to similar directions (*e.g.* with a high cosine similarity), then the conclusion above on gradient reduction still holds.

B.3. Linear decision boundary w

Still following subsection 3.3 of the main paper and the notations defined above, when there is no MFW, the gradient to $\boldsymbol{w}$ with the two data instances is

$$\begin{aligned}\nabla_{\boldsymbol{w}}\ell =&(\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_1)) - y_1) \times f_{\boldsymbol{\theta}}(\boldsymbol{x}_1)\\ &+ (\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times f_{\boldsymbol{\theta}}(\boldsymbol{x}_2). \end{aligned} \tag{18}$$

When MFW is performed (with the same setting as above: $\lambda_1 \in [0, 0.5]$ and $\lambda_2 = 0$), the gradient to $\boldsymbol{w}$ becomes

$$\begin{aligned}\nabla_{\boldsymbol{w}}\ell =&(\sigma(\boldsymbol{w}^\top h_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_1)) - y_1) \times h_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_1)\\ &+ (\sigma(\boldsymbol{w}^\top f_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times f_{\boldsymbol{\theta}}(\boldsymbol{x}_2), \end{aligned} \tag{19}$$

in which the first term's gradient direction changes from $f_{\boldsymbol{\theta}}(\boldsymbol{x}_1) = h_{\boldsymbol{\theta}}(\boldsymbol{z}_1)$ to $h_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_1)$. In the case where $h_{\boldsymbol{\theta}}$ is an identity function, this means that

$$\begin{aligned}\nabla_{\boldsymbol{w}}\ell =&(\sigma(\boldsymbol{w}^\top \tilde{\boldsymbol{z}}_1) - y_1) \times ((1-\lambda) g_{\boldsymbol{\theta}}(\boldsymbol{x}_1) + \lambda g_{\boldsymbol{\theta}}(\boldsymbol{x}_2))\\ &+ (\sigma(\boldsymbol{w}^\top g_{\boldsymbol{\theta}}(\boldsymbol{x}_2)) - y_2) \times g_{\boldsymbol{\theta}}(\boldsymbol{x}_2). \end{aligned} \tag{20}$$

In other words, MFW also weakens the tendency of the linear classifier $\boldsymbol{w}$ to fit the major class data[8]. This is another reason why the overall training progress with MFW can be more balanced across classes.

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

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

C. Experimental Setups

C.1. Datasets

To study the imbalanced classification problems on balanced datasets (*e.g.*, CIFAR-10 [36], CIFAR-100 [36], Tiny-ImageNet [38]), we follow [5, 9] to create imbalanced versions by reducing the number of training instances, such that the numbers of instances per class follow a certain distribution. Specifically, the *long-tailed* imbalance follows an exponential distribution. We control the degree of dataset imbalance by the imbalance ratio $\rho = \frac{N_{\text{max}}}{N_{\text{min}}}$, where N_{max} (resp. N_{min}) is the number of training instances of the largest major (resp. smallest minor) class.

Tiered-ImageNet [50] is a subset of ImageNet [11] widely used in few-shot learning [67]. The images are 84×84. There are three splits with disjoint classes in Tiered-ImageNet: 351 classes for many-shot model training, 97 classes for few-shot model validation, and 160 classes for few-shot model testing. 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. We treat classes in the other two splits as minor classes: we randomly select 5 training instances per class. All the classes have 3 validation instances and 50 test instances. We tune hyper-parameters on the $351 + 97$ classes using their validation instances: the 351 classes are many-shot (*i.e.*, major) and the 97 are few-shot classes (*i.e.*, minor). We then re-train the model with selected hyper-parameters and test it on the $351 + 160$ classes using their test instances: the 351 classes are many-shot (*i.e.*, major) and the 160 are few-shot classes (*i.e.*, minor).

In this supplementary material, we further experiment on ImageNet-LT [45], which is also a subset of ImageNet [11]. ImageNet-LT is a *long-tailed* imbalance dataset with ratio $\rho = 256$. It contains 1,000 classes with 115.8K training images in total, 20 images per class for validation, and 50 images per class in the test set.

Except for the Tiered-ImageNet dataset described above, we follow [9, 5, 80] to report the accuracy on the test set for CIFAR-10 / CIFAR-100 / ImageNet-LT and the accuracy on the validation set for Tiny-ImageNet and iNaturalist.

C.2. Implementation details of MFW

For all the experiments, we use mini-batch stochastic gradient descent (SGD) with momentum = 0.9 as the optimization solver. The softmax cross-entropy loss is used for MFW. We apply MFW after the embedding of the second group of convolutional layers.

We tune the beta distribution parameter α based on the performance on the held-out set (except for Tiered-ImageNet, which has a separate validation set from the test set). Concretely, we split a held-out set from the training set, following [72]. We held out 3 images per class, creating a small balanced held-out set. If a class has fewer than 3 images, we ignore them in hyper-parameter tuning.

We keep the scale value in Equation 9 of the main paper as $\beta = 2$ and $\beta = 0.01$ for the long-tailed and step cases, respectively. Deferred Re-Weight (DRW) [5] is also applied to MFW near the end of the training process, to further improve the accuracy. In detail, we apply the vanilla softmax (with MFW) at first, and after completing 80% of the training progress, we apply a weighted version of softmax for optimization. The class-wise weights are set based on [9].

C.3. Training details for imbalanced CIFAR

We use ResNet-32 [21] for all the CIFAR [36] experiments, following [5, 9]. The batch size is 128, and the weight decay is 2×10^{-4}. The initial learning rate is linearly warmed up to 0.1 in the first five epochs, and decays with cosine annealing. The model is trained for 300 epochs[9]. We follow [5] to do data augmentation. The 32×32 CIFAR images are padded to 40×40 and randomly flipped horizontally, and then are randomly cropped back to 32×32.

C.4. Training details for Tiny-ImageNet

We use ResNet-18 [21], following [5]. It is trained for 200 epochs with a batch size of 128. The initial learning rate is 0.1 and decays with cosine annealing. Images are padded 8 pixels on each size and randomly flipped horizontally, and then are randomly cropped back to 64×64.

C.5. Training details for Tiered-ImageNet

We use ResNet-12 [39, 73]. It is trained for 180 epochs with a batch size of 512, following [30]. The initial learning rate is 0.2 and decays with cosine annealing. Images are padded 8 pixels on each size and randomly flipped horizontally, and then are randomly cropped back to 84×84.

C.6. Training details for ImageNet-LT

We use ResNet-10 [21] and ResNext-50 [71]. They are trained for 200 epochs with a batch size of 256. The initial learning rate is 0.1 and decays with cosine annealing. In training, the images are resized to 256×256 and flipped horizontally, and are randomly cropped back to 224×224.

C.7. Training details for iNaturalist

We follow [9, 80] to use ResNet-50 [21] on iNaturalist. We train the model for 90 and 180 epochs with a batch size of 128. The learning rate is 0.05 at first. It decays at the 60th (resp. 120th) and the 80th (resp. 160th) by 0.1 when training for 90 epochs (resp. 180 epochs). We follow [80] to apply the standard pre-processing and data augmentation

[9] On CIFAR, we found that for ERM and other baselines, training with 200 epochs converges. As MFW weakens major features, some cases need more epochs to converge and we train them for 300.

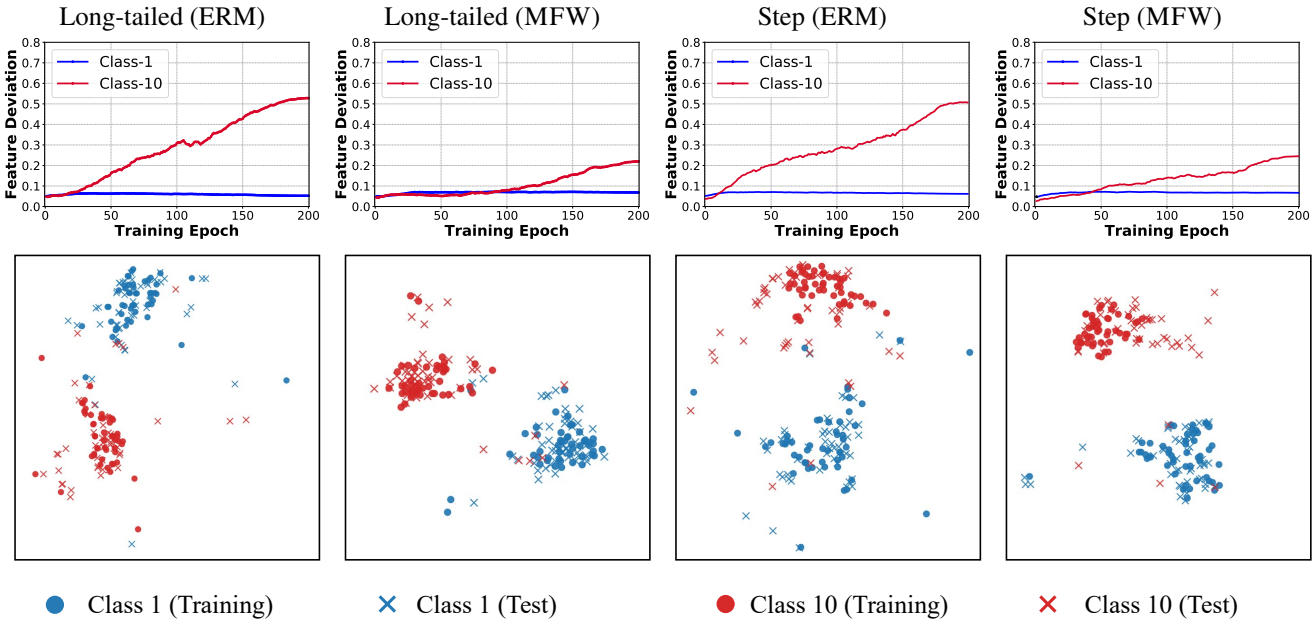

Figure 9. **Feature deviation between the training and test data per class along the training progress.** We experiment on CIFAR-10, using both the long-tailed and the step settings ($\rho = 100$). We only showed the most major ($c = 1$, with $5,000$ training samples) and minor classes ($c = 10$, with 50 training samples) for clarity. As the number of training epochs increases, the deviation increases, while MFW can achieve a much smaller deviation (top row). The bottom parts are the corresponding t-SNE embeddings of the training and test instances from the two classes (at the last epoch).

used for ImageNet [21]. We normalize the images by subtracting the RGB means computed on the training set. In training, the images are resized to 256×256 and flipped horizontally, and are randomly cropped back to 224×224.

D. Experimental Results and Analysis

We provide additional experimental results and analysis in this section.

D.1. The influence of the coefficient α

According to Algorithm 1 of the main paper, we sample a value λ_n from a beta distribution with a coefficient α to mix two features in MFW. With different α values, the beta distribution behaves differently. For example, the beta distribution with small α values (*i.e.*, $\alpha < 1$) favors sampling extreme λ_n values close to 0 or 1; large α increases the probability of sampling values near 0.5. We show the results when using different α in Table 6. We find that for the step case, a larger α is preferred; for the long-tailed case, a smaller α is preferred. We select α using the held-out set (or the validation set for Tiered-ImageNet). See subsection C.2 for details.

D.2. Ablation on the weight function.

We study different β in Equation 12 of the main paper. As shown in Figure 10, larger/smaller β (smoother/sharper weight changes) is preferred for long-tailed/step cases. We further apply Equation 12 to MixUp [76] and ReMix [8] but do not see improvements. We attribute this to the fundamental difference between MFW and them (cf. subsection 3.5): MFW does not change the labels, while MixUp does and ReMix mixes labels to favor minor classes.

Table 6. Test accuracy (%) on CIFAR-10/-100 with different values of α (for the beta distribution) in MFW.

	CIFAR-10		CIFAR-100	
α	Long-tailed	Step	Long-tailed	Step
0.1	75.9	73.3	43.1	40.8
0.2	76.8	74.3	44.7	42.6
0.5	77.1	75.2	43.6	44.0
1	78.5	77.6	42.1	46.5
5	77.4	80.1	41.2	46.9

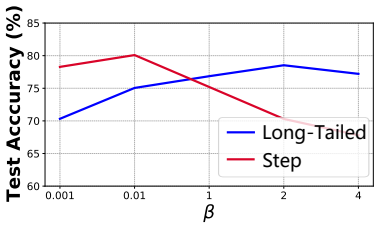

Figure 10. Ablation on the weight function.

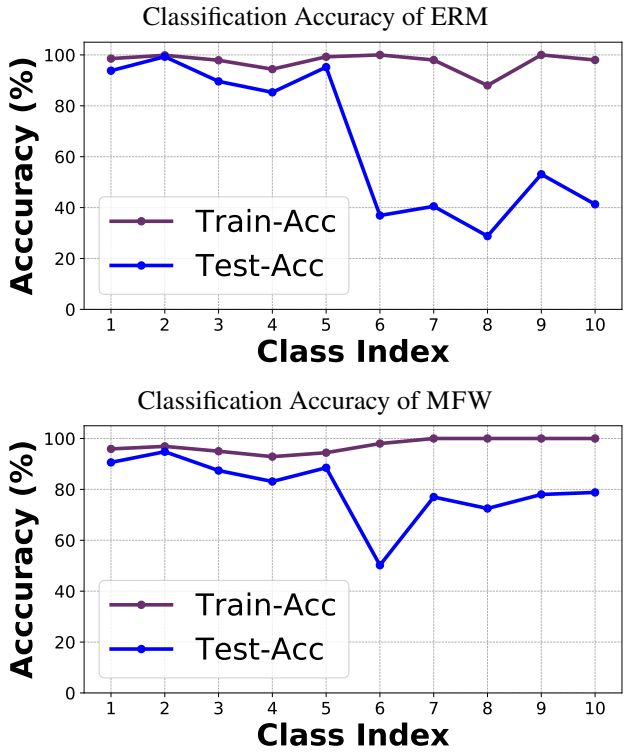

Figure 11. **The training and test accuracy per class of a neural network trained on class-imbalanced data.** We trained a ConvNet classifier using ResNet-32 [21] on a step imbalanced CIFAR-10 data set ($\rho = 100$) [36], following [9]. The first five and last five Classes have $5,000$ and 50 training instances, respectively. We compare training with ERM (top) and MFW (bottom). MFW leads to much higher test accuracy for the minor classes, with just a slight decrease of that for the major classes.

D.3. Whether MFW can reduce the feature deviation?

Following subsection 5.3 of the main paper, we further show the t-SNE [46] plots of the features to illustrate the feature deviation in Figure 9. We apply ERM and MFW on different imbalance configurations, including both long-tailed and step cases. The largest major class 1 and the smallest class 10 are selected for illustration.

In Figure 9, the top row shows the change of feature deviation values[10] along the training process, where MFW significantly reduces the feature deviation, justifying our claims that feature deviation is likely caused by the exaggerated gradients between major and minor classes. The bottom row shows the t-SNE visualization of $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ using the corresponding model (at the last epoch): circles indicate the training data and crosses indicate the test data. It can be seen that there exists feature deviation for class 10 (red) when applying ERM, which can be reduced using MFW. With MFW, the training and test features are clustered.

[10]Please see subsection 5.4 of the main paper and subsection A.2 for the definition.

Table 7. Test set accuracy (%) on imbalanced CIFAR-10 with larger imbalance ratio ρ.

Imbalance ratio ρ	500	1000
ERM	64.5	56.5
CDT [72]	66.1	59.0
MFW	65.6	62.0
MFW w/ DRW	67.3	63.0

Table 8. Test accuracy (%) on CIFAR-10/-100. We compare MFW 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
Mix-Up [76]	73.1	66.2	39.5	39.8
Manifold Mix-up [62]	73.0	65.4	38.3	39.4
MFW	**78.5**	**80.1**	**44.7**	**46.9**

D.4. Training and test accuracy across classes

We plot the training and test accuracy on the step imbalanced CIFAR-10 ($\rho = 100$) in Figure 11. We note that, in evaluating the training accuracy, we do not apply MFW—MFW is only applied in training the neural networks. We mention this in subsection 3.2 and subsection 3.4 in the main paper.

As shown in Figure 11, MFW leads to a much higher test accuracy for the minor classes (the average is over 60% in comparison to 40% by ERM), with just a slight decrease of that for the major classes. In summary, MFW can effectively facilitate class-imbalanced learning, especially for the challenging step imbalance cases.

D.5. Label mixing

We compare MFW to mixup [76] and manifold mixup [62]: they mix both the inputs (or features) and the labels. As shown in Table 8, MFW outperforms both of them, justifying that MFW is not regularizing the model predictions.

D.6. Performance on larger imbalance factors

We study the performance of MFW on a long-tailed scenario with larger imbalance ratio ρ on CIFAR-10 in Table 7. In detail, we construct the dataset in the same manner as [9, 5] and set $\rho = 500$ and $\rho = 1000$. Since the dataset is more imbalanced, it becomes more difficult. Compared to a strong baseline CDT [72], the gain by MFW increases as ρ increases. Overall, we find that MFW has advantages in tackling extreme imbalance, *e.g.*, step settings (all minor classes have $\frac{1}{\rho}$ training data than the major classes) or larger ρ in long-tailed settings.

Table 9. Top-1 test set accuracy (%) on ImageNet-LT.

ResNet-10		ResNeXt-50	
OLTR [45]	41.9	OLTR [45]	48.7
cRT [30]	41.8	cRT [30]	49.6
τ-Norm [30]	40.6	τ-Norm [30]	49.4
LWS [30]	41.4	LWS [30]	49.9
CDT [72]	41.4	De-confound [58]	48.6
SEQL [57]	36.4	De-confound-TDE [58]	51.8
MFW	40.1	MFW	50.5
MFW w/ DRW	42.0	MFW w/ DRW	51.9

D.7. Results on ImageNet-LT

We show the top-1 accuracy on ImageNet-LT in Table 9 with ResNet-10 and ResNext-50. Our MFW achieves promising results when compared with others.

D.8. Further comparisons on CIFAR

[57] mentioned that they used stronger data augmentations on CIFAR. Using the same augmentations, we have 47.8 on CIFAR-100 (ρ=200) vs. theirs at 43.4.

D.9. Deferred Re-Weight (DRW)

For step imbalance, MFW w/o DRW outperforms baselines in nearly all cases. (See tables in the main paper.) We include DRW mainly for a fair comparison to [8]. We note that, deferred scheduling (*e.g.*, DRW) is also included in the implementation of other methods, *e.g.*, [5, 26, 30, 34, 80].

D.10. MFW on other network architectures

We further study the generalizability of MFW to other network architectures. Using DenseNet-121 [25], MFW achieves $76.2/81.2/92.1$ for long-tailed CIFAR-10 at $\rho = 200/100/10$, higher than ERM which has $70.2/77.6/91.3$ (cf. Table 1 in the main paper).