Learnable Logit Adjustment for Imbalanced Semi-Supervised Learning under Class Distribution Mismatch

Supplementary Material

A. Additional Related Works

A.1. Class-imbalanced fully supervised learning

Class-imbalanced learning (CIL) algorithms are used to train unbiased classifiers when the class distribution of the training set is class-imbalanced. Resampling techniques [1, 7, 18, 25] are used to re-balance the class distribution by oversampling minority class samples or undersampling majority class samples. Reweighting techniques [12, 21, 24, 48, 64] re-balance the gradients for each class by assigning higher weights for minority classes. Loss functions introduced by Cao et al. [6] and Ren et al. [52] aim to minimize a bound on generalization error. Kim et al. [29] and Yin et al. [70] focused on transferring knowledge from majority class data to minority class data. Kang et al. [27] proposed decoupling the learning of features and classifiers, while Menon et al. [47] introduced post-hoc logit-adjustment to minimize the balanced error. Recently, class-balanced distillation [23, 45], contrastive learning [11, 26, 46, 58], and multi-expert learning [5, 41, 60, 68, 74, 75] have been used for CIL.

A.2. Semi-supervised learning

SSL algorithms aim to improve classification performance by utilizing unlabeled samples for training. One SSL technique, called entropy minimization [16], encourages the classifier to produce confident class predictions for unlabeled samples by employing pseudo-labels [38]. Another SSL approach, consistency regularization [49, 51, 55], enforces consistency in the class predictions for two augmented versions of the same unlabeled sample. FixMatch [54] and ReMixMatch [3] incorporate both entropy minimization and consistency regularization into their frameworks, leveraging strong data augmentation techniques [10, 13]. In addition, ReMixMatch also incorporates Mixup regularization [2, 57] and rotation-based self-supervised learning [15]. Recently, CoMatch [42] introduced graph-based contrastive learning, and FlexMatch [73] introduced curriculum pseudo-labeling. FreeMatch [62] and SoftMatch [8] extended curriculum pseudo-labeling by using exponential moving averages of prediction confidence and truncated Gaussian functions, respectively.

B. Time complexity of the proposed algorithm

To verify that learning λ adds negligible time complexity compared to solely training the base SSL algorithm, we measured the FLoating point OPerations per Second (FLOPS) for the training of the base SSL algorithm (FixMatch and ReMixMatch) and the base SSL algorithm+LLA. We conducted experiments on CIFAR-10-LT with Nvidia 3090ti. The results are summarized in Tab. 6. We can observe that learning λ with LLA has a negligible impact on time complexity.

CIFAR-10-LT	
Algorithm	iteration/sec
FixMatch	18.52
FixMatch+LLA	18.38
ReMixMatch	6.80
ReMixMatch+LLA	6.76

Table 6. FLOPS for the training of FixMatch, FixMatch+LLA, ReMixMatch, and ReMixMatch+LLA on CIFAR-10-LT.

C. Illustration of the test phase

As we described in the main paper, the proposed algorithm also adjusts the class predictions on test samples. The test phase of the proposed algorithm is illustrated in Fig. 5.

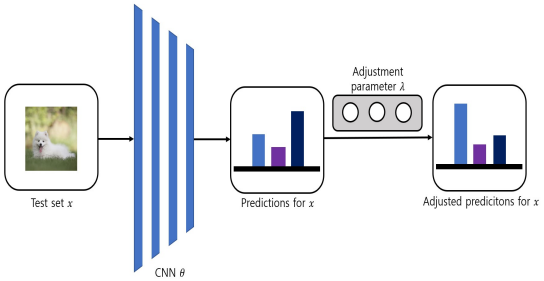


Figure 5. Test phase of the proposed algorithm

D. Pseudo code of the proposed algorithm

Algorithm 1 provides the pseudo-code of the proposed algorithm including both training and test phases.

E. Further details of datasets

CIFAR-10-LT and CIFAR-100-LT are artificially generated long-tailed datasets that sample images from CIFAR-10 and CIFAR-100 [33], respectively. The numbers of

Algorithm 1 Pseudo code of the proposed algorithm

Input: Labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$, test set $\mathcal{X}$, network parameters θ, parameters for LLA λ
Output: Adjusted class predictions for test samples $f_\theta^*\left(x_t^{test}\right)$ for $t = 1, \ldots, T$
while training **do**
  Generate minibatches $\mathcal{M}_\mathcal{L} = \{(x_b, y_b) : b \in (1, \ldots, B)\} \subset \mathcal{L}$ and $\mathcal{M}_\mathcal{U} = \{(u_b) : b \in (1, \ldots, \mu B)\} \subset \mathcal{U}$
  Calculate logits for weakly augmented unlabeled images $g_\theta\left(\alpha\left(u_b\right)\right) = \omega\left(\psi\left(\alpha\left(u_b\right)\right)\right)$ for $b = 1, \ldots, \mu B$
  Adjust logits for weakly augmented unlabeled images $g_\theta^*\left(\alpha\left(u_b\right)\right) = g_\theta\left(\alpha\left(u_b\right)\right) - \log\phi\left(\lambda\right)$ for $b = 1, \ldots, \mu B$
  Generate refined pseudo-labels $q_b^* = \phi\left(g_\theta^*\left(\alpha\left(u_b\right)\right)\right)$ for $b = 1, \ldots, \mu B$
  Calculate logits for weakly augmented labeled images $g_\theta\left(\alpha\left(x_b\right)\right) = \omega\left(\tau_{y_b} \times \psi\left(\alpha\left(x_b\right)\right)\right)$ for $b = 1, \ldots, B$
  Calculate logits for strongly augmented unlabeled samples $g_\theta\left(\mathcal{A}\left(u_b\right)\right) = \omega\left(\nu_{q_b^*} \times \psi\left(\mathcal{A}\left(u_b\right)\right)\right)$ for $b = 1, \ldots, \mu B$
  Adjust logits for weakly augmented labeled images $g_\theta^*\left(\alpha\left(x_b\right)\right) = g_\theta\left(\alpha\left(x_b\right)\right) - \log\phi\left(\lambda\right)$ for $b = 1, \ldots, B$
  Adjust logits for strongly augmented unlabeled images $g_\theta^*\left(\mathcal{A}\left(u_b\right)\right) = g_\theta\left(\mathcal{A}\left(u_b\right)\right) - \log\phi\left(\lambda\right)$ for $b = 1, \ldots, \mu B$
  Compute the class-averaged loss for labeled minibatch $L_l = \frac{1}{K}\Sigma_k \frac{\Sigma_b p_{b,k} \log g_\theta^*(\alpha(x_b))_k}{\Sigma_b p_{b,k} + \epsilon}$
  Compute the class-averaged loss for unlabeled minibatch $L_u = \frac{1}{K}\Sigma_k \frac{\Sigma_b q_{b,k}^* \log g_\theta^*(\mathcal{A}(u_b))_k}{\Sigma_b q_{b,k}^* + \epsilon}$
  Compute the training loss of the base SSL algorithm L_{base} with the refined pseudo-labels q_b^* for $b = 1, \ldots, \mu B$
  Compute the total training loss $L = L_{base} + L_c = L_{base} + L_l + L_u$
  $\Delta\boldsymbol{\theta} \propto \nabla_{\boldsymbol{\theta}} L_{base}, \quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  $\Delta\boldsymbol{\lambda} \propto \nabla_{\boldsymbol{\lambda}} L_c, \quad \boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}$
end while
Calculate logits for test set $g_\theta\left(x_t^{test}\right) = \omega\left(\psi\left(x_t^{test}\right)\right)$ for $t = 1, \ldots, T$
Adjust logits for test set $g_\theta^*\left(x_t^{test}\right) = g_\theta\left(x_t^{test}\right) - \log\phi\left(\lambda\right)$ for $t = 1, \ldots, T$
Obtain the adjusted class predictions $f_\theta^*\left(x_t^{test}\right) = \arg\max_k g_\theta^*\left(x_t^{test}\right)_k$ for $t = 1, \ldots, T$

labeled and unlabeled images of the kth class, denoted as N_k and M_k, respectively, can be expressed as $N_k = N_1 \times (N_K/N_1)^{\frac{k-1}{K-1}}$ and $M_k = M_1 \times (M_K/M_1)^{\frac{k-1}{K-1}}$. For CIFAR-10-LT, we set N_1 as 1500 and M_1 as 3000. We first conducted experiments under $\gamma_l = \gamma_u$, while varying the imbalance ratio as 50, 100, and 150. Then, we conducted experiments where γ_u is unknown and $\gamma_l \neq \gamma_u$, while setting $\gamma_l = 100$ and varying γ_u as 1, 50, and 150. For CIFAR-100-LT, we set N_1 as 150 and M_1 as 300. We conducted experiments under $\gamma_l = \gamma_u$, while varying the imbalance ratio as 20, 50, and 100.

STL-10-LT is an artificially generated long-tailed dataset that samples images from STL-10 [9], where the number of labeled images of the kth class, denoted as N_k, can be expressed as $N_k = N_1 \times (N_K/N_1)^{\frac{k-1}{K-1}}$. Note that the class distribution of the unlabeled set of STL-10-LT is unknown. We set N_1 as 450 and used all 100,000 unlabeled images for training. We conducted experiments while varying γ_l to 10 and 20.

Small-ImageNet-127 is a down-sampled variant of ImageNet-127 [22], which was created by categorizing the ImageNet [53] into 127 classes according to the WordNet hierarchy. The training set consists of 1,281,167 images and exhibits class imbalance, with an imbalance ratio of 286. Fan et al. [14] down-sampled the images to 32×32 and 64×64 and used 10% of the training set as a labeled set. Following [14, 66], we conducted experiments on Small-ImageNet-127 using only FixMatch because training ReMixMatch requires an excessive training cost. Note that the test set of Small-ImageNet-127 also has imbalanced class distribution.

F. Further details of experimental setup and baseline algorithms

We used the Adam optimizer [30] for training with a learning rate set to 2×10^{-3}. We used exponential moving average (EMA) of the model parameters at each iteration with a decay factor of 0.999, to assess the classification performance on the test set. Wide ResNet-28-2 [72] was used as a deep CNN for the experiments on CIFAR-10-LT, CIFAR-100-LT, and STL-10-LT, while ResNet-50 [19] was used for Small-ImageNet-127. For CIFAR-100, we set the parameter of weight decay to 0.08 because CIFAR-100 has relatively many classes. For other datasets, we set the parameter of weight decay to 0.04 when $N + M < 3 \times 10^4$, while we set it to 0.01 for FixMatch and 0.02 for ReMixMatch when $N + M \geq 3 \times 10^4$. This is due to the reduced effectiveness of weight decay with larger training set sizes. Similarly, we set the hyperparameter δ of EFCC to 0.1 when $N + M < 3 \times 10^4$ and set it to 0.2 otherwise. For the experiments using FixMatch, we trained the proposed algorithm for 500 epochs (1 epoch=500 iterations) with both labeled and unlabeled minibatches set to a size of 64. We did not use confidence threshold η to employ every unlabeled sam-

ple. Instead, we reduced the risk associated with the use of incorrect pseudo-labels by employing soft pseudo-labels. For the experiments using ReMixMatch, we trained the proposed algorithm for 300 epochs with labeled minibatches of size 64 and unlabeled minibatches of size 128. When the class distribution of the unlabeled set is unknown, we did not employ the distribution alignment technique because it relies on the class distribution of the labeled set as an approximation for the unlabeled set. Instead, we included a classification loss for weakly augmented labeled images in the L_{base}.

For CIFAR-10-LT, CIFAR-100-LT, and STL-10-LT, we conducted experiments three times, varying the random seed each time. We used Nvidia Tesla-V100 and 3090ti for the GPU server, and PyTorch 1.11.0 and 1.12.1 for the deep learning library. The experimental results in the main paper can be reproduced by implementing the code provided in the supplementary material. We compared the classification performance of LLA with that of various baseline algorithms. **1. For SSL algorithms**, we used FixMatch [54] and ReMixMatch [3] as baseline algorithms. **2. For CISSL algorithms**, we used DARP, DARP+LA, DARP+cRT [28], CReST, CReST+LA [66], ABC [40], CoSSL [14], SAW, SAW+LA, SAW+cRT [35], UDAL [36], L2AC [59], Adsh [17], and DebiasPL [61] as baseline algorithms. Every CISSL algorithm was combined with either FixMatch or ReMixMatch. We initially conducted experiments using baseline methods based on the official codes available on GitHub. If we were able to reproduce the experimental results from the original article, we reported the results from the original article. Otherwise, we reported the results from our re-implementation. To clarify the source of experimental results, we will add the above details in Section 5.1 of the main paper.

G. Comparison with DASO, DebiasPL, UDAL, L2AC and ACR

DASO [50], DebiasPL [61], UDAL [36], L2AC [59], and ACR [67] measured performance under different settings compared to ours. To compare the classification performance of these algorithms and the proposed algorithm, we conducted experiments using the official codes of the algorithms available on GitHub. In Tab. 7, Tab. 8, and Tab. 9, classification performances of DASO and the proposed algorithm are summarized. From the tables, we can observe that LLA achieves better classification performance than DASO. Additionally, from Tab. 10, Tab. 11 and Tab. 12, we can observe that LLA achieves better performance than DebiasPL, UDAL, L2AC and ACR.

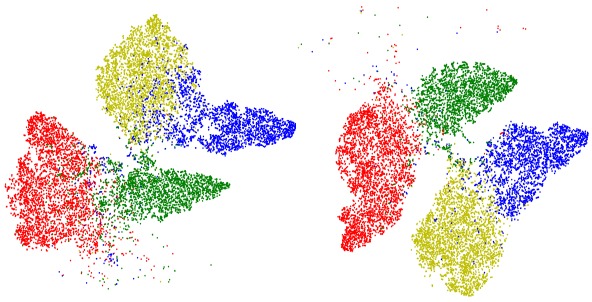
H. Fine-grained experimental results of LLA

To demonstrate that the proposed algorithm achieves higher classification performance for minority classes than baseline algorithms, we conducted experiments using FixMatch/ReMixMatch, FixMatch/ReMixMatch+SAW [35], FixMatch/ReMixMatch+SAW+cRT [27], and FixMatch/ReMixMatch+LLA on CIFAR-10-LT ($\gamma_l = 100$ and $\gamma_u = 1$) and measured the classification accuracy for each of the Many/Medium/Few groups. For CIFAR-10-LT, we categorized the first three classes as the "many" group, the subsequent four classes as the "medium" group, and the final three classes as the "few" group. From Tab. 13, we can observe that LLA achieves higher classification accuracy for the "few" group than the baseline algorithms.

I. Experimental results using FreeMatch as the base SSL algorithm.

To validate the compatibility of LLA with a recent SSL algorithm, we conducted experiments on CIFAR-10-LT under $\gamma_l = \gamma_u = 100$ and $\gamma_l = 100$ & $\gamma_u = 1$ by using FreeMatch [63] as the base SSL algorithm. We compared the classification performance of FreeMatch+LLA against FreeMatch, FreeMatch+SAW+cRT, and FreeMatch+CoSSL. The results in Tab. 14 indicate that FreeMatch+LLA significantly outperforms the compared algorithms

J. t-SNE visualizations for training set of CIFAR-10-LT



(a) Without EFCC (b) With EFCC

Figure 6. t-SNE visualizations of features from the CIFAR-10 training set, calculated using ReMixMatch+LLA without/ with EFCC

In Fig. 6, we visualize the features of the two most dominant classes (by yellow and red points) and two least dominant classes (blue and green points) of the CIFAR-10-LT ($\gamma_l = 100$ and $\gamma_u = 1$) training set using t-SNE [56]. EFCC enhances the separability of minority class features.

CIFAR-10-LT ($\gamma = \gamma_l = \gamma_u$)			
Algorithm	$\gamma = 50$	$\gamma = 100$	$\gamma = 150$
FixMatch+DASO	81.8/ 81.0	75.7/ 74.0	72.0/ 68.9
FixMatch+DASO+LA	84.1/ 83.7	79.4/ 78.8	76.5/ 75.5
FixMatch+LLA (Ours)	**88.1/87.8**	**84.8/84.5**	**82.2/81.5**
ReMixMatch+DASO	82.5/ 81.9	76.0/ 73.9	70.8/ 66.5
ReMixMatch+DASO+LA	85.9/ 85.7	82.8/ 82.4	79.0/ 78.4
ReMixMatch+LLA (Ours)	**89.0/ 88.8**	**85.8/ 85.6**	**83.2/ 82.9**

Table 7. bACC/GM of LLA and DASO on CIFAR-10-LT under $\gamma = \gamma_l = \gamma_u$.

	CIFAR-10-LT ($\gamma_l = 100$)			STL-10-LT	
Algorithm	$\gamma_u = 1$	$\gamma_u = 50$	$\gamma_u = 150$	$\gamma_l = 10$	$\gamma_l = 20$
FixMatch+DASO	86.4/ 86.0	79.1/ 78.2	74.2/ 71.6	68.4/ 65.3	62.1/ 58.9
FixMatch+DASO+LA	86.2/ 85.8	81.7/ 81.2	78.0/ 77.0	68.9/ 66.3	66.0/ 64.6
FixMatch+LLA	**88.6/ 88.4**	**86.0/ 85.7**	**83.2/ 82.7**	**82.6/ 81.9**	**79.5/ 78.6**
ReMixMatch+DASO	89.6/ 89.3	79.6/ 77.8	72.3/ 69.0	75.1/ 73.6	66.8/ 61.8
ReMixMatch+DASO+LA	80.6/ 77.7	84.8/ 84.5	79.7/ 79.2	78.1/ 77.3	75.3/ 74.0
ReMixMatch+LLA	**90.4/ 90.2**	**87.2/ 87.0**	**83.8/ 83.6**	**84.0/ 83.2**	**82.1/ 81.1**

Table 8. bACC/GM of LLA and DASO on CIFAR-10-LT and STL-10-LT under $\gamma_l \neq \gamma_u$.

CIFAR-100-LT ($\gamma = \gamma_l = \gamma_u$)			
Algorithm	$\gamma = 20$	$\gamma = 50$	$\gamma = 100$
FixMatch+DASO	45.8	39.2	33.9
FixMatch+DASO+LA	46.2	39.9	34.5
FixMatch+LLA	**54.7**	**49.2**	**44.5**
ReMixMatch+DASO	51.5	43.0	38.2
ReMixMatch+DASO+LA	52.8	45.5	40.3
ReMixMatch+LLA	**57.0**	**50.9**	**45.7**

Table 9. bACC of LLA and DASO on CIFAR-100-LT.

CIFAR-10-LT ($\gamma = \gamma_l = \gamma_u$)			
Algorithm	$\gamma = 50$	$\gamma = 100$	$\gamma = 150$
FixMatch+DebiasPL	85.9/ 85.3	80.6/ 79.8	76.1/ 74.4
FixMatch+UDAL	86.5/ 86.1	81.4/ 80.9	77.9/ 76.5
FixMatch+L2AC	87.4/ 87.0	82.1/ 81.5	77.6/ 75.8
FixMatch+ACR	86.2/ 85.9	81.8/ 81.4	79.7/ 78.5
FixMatch+LLA	**88.1/ 87.8**	**84.8/ 84.5**	**82.2/ 81.5**

Table 10. bACC/GM of LLA and compared algorithms on CIFAR-10-LT under $\gamma = \gamma_l = \gamma_u$.

	CIFAR-10-LT ($\gamma_l = 100$)			STL-10-LT	
Algorithm	$\gamma_u = 1$	$\gamma_u = 50$	$\gamma_u = 150$	$\gamma_l = 10$	$\gamma_l = 20$
FixMatch+L2AC	88.1/ 87.9	82.6/ 82.1	77.0/ 76.1	79.9/ 79.1	77.0/ 75.8
FixMatch+ACR	85.6/ 85.3	82.4/ 82.0	78.6/ 78.0	81.1/ 80.5	77.5/ 76.4
FixMatch+LLA	**88.6/ 88.4**	**86.0/ 85.7**	**83.2/ 82.7**	**82.6/ 81.9**	**79.5/ 78.6**

Table 11. bACC/GM of LLA and compared algorithms under $\gamma_l \neq \gamma_u$.

CIFAR-100-LT ($\gamma = \gamma_l = \gamma_u$)			
Algorithm	$\gamma = 20$	$\gamma = 50$	$\gamma = 100$
FixMatch+DebiasPL	52.5	46.2	41.0
FixMatch+UDAL	53.6	48.0	43.7
FixMatch+L2AC	52.6	45.9	40.7
FixMatch+ACR	52.2	46.0	41.1
FixMatch+LLA	**54.7**	**49.2**	**44.5**

Table 12. bACC of LLA and compared algorithms on CIFAR-100-LT.

CIFAR-10-LT ($\gamma_l = 100$, $\gamma_u = 1$)				
Algorithm	Overall	Many	Medium	Few
FixMatch	70.2	96.3	77.7	34.0
w/ SAW	81.2	95.6	82.9	64.5
w/ SAW+cRT	84.6	87.8	85.5	80.2
w/ LLA	88.6	94.9	87.7	83.4
ReMixMatch	65.4	96.6	70.8	27.0
w/ SAW	87.0	96.8	86.4	78.0
w/ SAW+cRT	88.8	94.5	87.8	84.4
w/ LLA	90.4	94.2	88.6	89.0

Table 13. Fine-grained classification performance on CIFAR-10-LT under $\gamma_l = 100$, and $\gamma_u = 1$.

	CIFAR-10-LT	
Algorithm	$\gamma_l = \gamma_u = 100$	$\gamma_l = 100$, $\gamma_u = 1$
FreeMatch	75.4/72.9	74.2/69.5
w/ CoSSL	81.7/81.1	87.9/87.6
w/ SAW+cRT	82.8/82.3	86.4/86.2
w/ LLA	**85.1/84.8**	**89.5/89.3**

Table 14. Experimental results with using FreeMatch [63] as the base SSL algorithm.