Long-tailed Partial Label Learning by Head Classifier and Tail Classifier Cooperation

Yuheng Jia^{1,2}, Xiaorui Peng^{1,2}, Ran Wang^{3,4*}, Min-Ling Zhang^{1,5*}

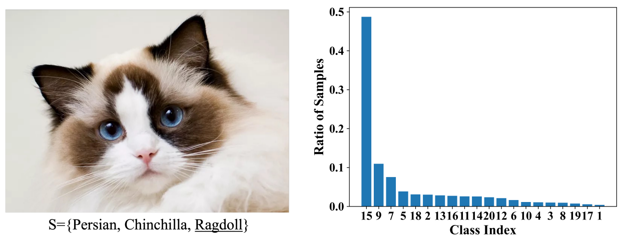
^{1}School of Computer Science and Engineering, Southeast University
^{2}Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
^{3}Shenzhen Key Laboratory of Advanced Machine Learning and Applications, School of Mathematical Science, Shenzhen University, Shenzhen, China
^{4}Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen, China
^{5}Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China
{yhjia, xiaorui, zhangml}@seu.edu.cn; wangran@szu.edu.cn

Abstract

In partial label learning (PLL), each instance is associated with a set of candidate labels, among which only one is correct. The traditional PLL almost all implicitly assume that the distribution of the classes is balanced. However, in real-world applications, the distribution of the classes is imbalanced or long-tailed, leading to the long-tailed partial label learning problem. The previous methods solve this problem mainly by ameliorating the ability to learn in the tail classes, which will sacrifice the performance of the head classes. While keeping the performance of the head classes may degrade the performance of the tail classes. Therefore, in this paper, we construct two classifiers, i.e., a head classifier for keeping the performance of dominant classes and a tail classifier for improving the performance of the tail classes. Then, we propose a classifier weight estimation module to automatically estimate the shot belongingness (head class or tail class) of the samples and allocate the weights for the head classifier and tail classifier when making prediction. This cooperation improves the prediction ability for both the head classes and the tail classes. The experiments on the benchmarks demonstrate the proposed approach improves the accuracy of the SOTA methods by a substantial margin. Code and data are available at : https://github.com/pruirui/HTC-LTPLL.

Introduction

Labeling ambiguity occurs in many real-world scenarios. For example, as shown in Figure 1(a), a non-expert annotator usually cannot accurately judge whether it is a Persian cat, a Chinchilla cat, or a Ragdoll cat. A common-used strategy is to treat all the possible labels as the candidate labels, which leads to the partial label learning (PLL) problems (Cour, Sapp, and Taskar 2011). Formally speaking, PLL is a weakly supervised learning paradigm that allows samples to be associated with a set of candidate labels, of which only one is the ground truth label. PLL is more friendly to non-expert annotators and can greatly reduce the cost of annotation, which has been applied to many applications, such as automatic face annotation (Chen, Patel, and Chellappa 2018), web mining (Luo and Orabona 2010), and face age estimation (Panis and Lanitis 2014). The key to PLL is label disambiguation, i.e., finding the correct label from the candidate label set. For instance, (Wang, Zhang, and Li 2022; Jia, Jiang, and Wang 2023; Jia, Si, and Zhang 2023) used the similarity of features for disambiguation i.e., if two samples are similar to each other in the features, they are likely to share the same ground-truth label. (Lv et al. 2020; Jia, Yang, and Dong 2023) perform disambiguation by dynamically adjusting the confidence of the candidate labels. PiCO (Wang et al. 2022b) introduces contrastive learning (He et al. 2020) to keep prototypes for disambiguation.



(a) Sample of PLL (b) Class distribution of "VOC"

Figure 1: (a) An image with three candidate labels S={Persian, Chinchilla, Ragdoll}, where the ground truth label is Ragdoll. (b) The class distribution of the real-world PLL dataset PASCAL VOC is a long-tailed distribution.

Despite the promise, these PLL methods all explicitly or implicitly assume that the training samples follow an approximate class-balanced distribution, i.e., each class has a roughly similar numbers of training samples. However, the class distribution collected from the real-world applications usually obeys a long-tailed distribution (Zhang et al. 2023; Kang et al. 2020; Menon et al. 2021) i.e., the leading classes have much more samples than the non-leading classes. As shown in Figure 1(b), the class distribution of the real-world PLL dataset PASCAL VOC (Hong et al. 2023) follows a long-tailed distribution. *Therefore, the long-tailed partial label learning (LT-PLL) is a more practical setting.* The long-tailed class distribution makes the current PLL bi-

*Corresponding author.

ases towards the dominant classes, and performs poorly on the tail classes, which further increase the difficulty for label disambiguation. Moreover, most methods designed for long-tailed distribution learning cannot be used directly for LT-PLL because they need the class distribution of the training set, which is unavailable in PLL due to label ambiguities.

So far, only few works were proposed to solve the under-explored LT-PTT problem. For example, SoLar (Wang et al. 2022a) performs long-tailed label disambiguation by optimal transport based on an estimated class distribution. RECORDS (Hong et al. 2023) first dynamically maintains a prototype feature for estimation of class distribution, and then uses logit-adjustment (Menon et al. 2021) to remedy the bias stemed from long tail effect. *Although the above methods achieve great performance, they tend to fall into a dilemma of trade-offs: focusing more on the tail classes would degrade the performance of the head classes, while focusing more on the head classes would neglect the learning of the tail classes.*

To relieve this issue, we propose a model comprised of two classifiers, where the head classifier and the tail classifier take the responsibility for the samples from the head classes and the tail classes, respectively, leading to high-quality prediction on all the class shots. Specifically, we first disambiguate the candidate label set and estimate the class distribution by exploring the information from the moving average of the model's outputs. Then, we construct the head classifier and the tail classifier by an imbalanced sampling strategy and a balanced sampling strategy based on the estimated class distribution. On this basis, we propose a classifier weight estimation (CWE) module, which can perceive the class belongingness (i.e., head class or tail class) of a sample and then adaptively weight and fuse the outputs of the two classifiers to produce more accurate prediction. We extensively evaluate our method on several benchmarks and real-world TL-PLL datasets, which demonstrates that our method largely outperforms SOTA methods, e.g., on the CIFAR10-LT dataset we method improves the classification accuracy by **13.46%** compared with the best comparison.

Related Work

Partial Label Learning (PLL)

In PLL, each sample is associated with a candidate label set in which the ground truth label is hidden, and the goal is to learn a multi-class classifier from the ambiguous candidate label set (Hüllermeier and Beringer 2006; Nguyen and Caruana 2008). The average-based methods (Hüllermeier and Beringer 2006; Cour, Sapp, and Taskar 2011) treat all the candidate labels as true labels equally, while the identification based methods (Lv et al. 2020; Zhang et al. 2022; Wen et al. 2021) aim to find the ground-truth label from the candidate labels. Recently, deep learning-based PLL methods (Feng et al. 2020; Wen et al. 2021) have become popular due to their excellent performance. For example, PRODEN (Lv et al. 2020) normalizes the model output over the candidate label set as the disambiguated pseudo labels. PiCO (Wang et al. 2022b) introduces contrastive learning into PLL and performs disambiguation by maintaining class prototypes. CORR (Wu, Wang, and Zhang 2022) performs consistency regularization on candidate labels for disambiguation. Despite the promise, these methods neglect that in real-world scenarios data is usually long-tailed distributed.

Long-tailed Partial Label Learning (LT-PLL)

There have been few works done on the LT-PLL problem. For example, (Wang and Zhang 2018; Liu et al. 2021) addresses the LT-PLL problem by over-sampling and regularization constraints. SoLar (Wang et al. 2022a) formulates the LT-PLL as an optimal transport problem and uses the Sinkhorn-Knopp algorithm to get a fast approximation. The key idea of SoLar is to alleviate the pseudo label bias towards head classes by constraining the pseudo labels to satisfy the estimated class distribution priors, which are dynamically generated by the model. Almost at the same time, RECORDS (Hong et al. 2023) proposes a dynamic rebalancing auxiliary strategy for LT-PLL, which dynamically recovers the class distribution priors by maintaining prototypes, and then performs logit adjustment on the output of model.

Although these methods have achieved some results, they tend to overemphasize the tail classes (resp. head class) and neglect the accuracy of the head classes (resp. tail class). As a single classifier cannot always balance the performance on the head and tail classes, we propose to construct two classifiers that pay more attention to the head and tail classes, respectively. Through the cooperation of these two classifiers, the proposed model achieve excellent performance on both the head classes and the tail classes.

Proposed Method

In this section, we will first introduce how to disambiguate the candidate label set and estimate the class distribution of training samples, then we present how to construct the head classifier and the tail classifier. Furthermore, we propose the classifier weight estimation (CWE) module to allocate weights for the head classifier and the tail classifier, and finally introduce how to make prediction based on the cooperation of two classifiers. Before presenting our model, we briefly introduce the used notations.

Let $\mathcal{X} = \mathbb{R}^d$ be the d-dimensional input space and $\mathcal{Y} = \{1, \cdots, C\}$ be the label space, and C is the number of classes. We denote $\mathcal{D} = \{(x_i, S_i)\}_{i=1}^n$ the training dataset with n examples, and each tuple in $\mathcal{D}$ comprises of a feature vector $x_i \in \mathcal{X}$ and a candidate label set $S_i \subset \mathcal{Y}$. In PLL, the ground truth label y_i is concealed in candidate label set i.e. $y_i \in S_i$ (Wang et al. 2022b). Our goal is to train a multi-class classification model using the LT-PLL dataset $\mathcal{D}$. Let $f(x;\theta)$ denote a deep feature extractor parameterized by θ, which transforms x to into an embedding vector. Then the output logits of x on head classifier and tail classifier are given by $z^h(x) = g^h(f(x;\theta); W^h)$ and $z^t(x) = g^t(f(x;\theta); W^t)$ respectively. We use $P^h = [p_1^h, \cdots, p_n^h]^T = [p_{ij}^h]_{n\times C}$ and $P^t = [p_1^t, \cdots, p_n^t]^T = [p_{ij}^t]_{n\times C}$ to denote the probability prediction matrix of the head classifier and tail classifier, where $p_i^h = softmax(z^h(x_i))$ and $p_i^t = softmax(z^t(x_i))$. For convenience, a batch of samples B of size n_b is

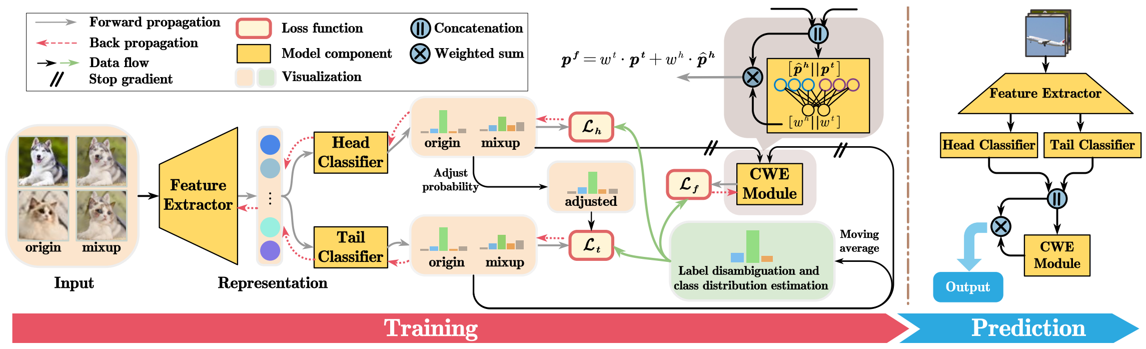


Figure 2: Illustration for our method. The Head Classifier pays more attention to the head classes, while the Tail Classifier focuses on the tail classes. Classifier Weight Estimation (CWE) Module is a shallow network that learns how to judge whether a sample belongs to the head or tail classes based on the outputs of the two classifiers, and assigns corresponding weights to the two classifiers for prediction.

used for the subsequent formula description. $\mathcal{H}(p_i, q_i) = \sum_{j=1}^{C} q_{ij} log p_{ij}$ denotes cross-entropy loss.

Candidate Label Set Disambiguation and Class Distribution Estimation

LT-PLL faces two kinds of challenges, where the first one is how to find the correct label from the candidate label set (a.k.a. label disambiguation), and the second one is to estimate the class distribution of the training set that is critical to handle the long-tailed learning problem. To this end, we propose to use the output of the model itself to solve those two challenges simultaneously. Specifically, as the ground-truth label only exits in the candidate label set, we first correct the tail classifier's prediction, making its probability on the non-candidate label set zero, i.e.,

$$p_{ij}^c = \frac{\mathbb{I}(j \in S_i) p_{ij}^t}{\sum_{u \in S_i} p_{iu}^t}, \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function that outputs 1 if the condition satisfies while 0 otherwise, p_{ij}^t is probability prediction of tail classifier for the sample x_i on the j-th label and p_{ij}^c is corrected prediction of tail classifier. As the prediction on one epoch may be not reliable, we propose to use the moving average of p_{ij}^c to build the soft-pseudo label, i.e.,

$$q_i \leftarrow \mu q_i + (1 - \mu) p_i^c, \tag{2}$$

where $\mu \in [0, 1]$ is a predefined scalar and $q_i = [q_{ij}, \cdots, q_{iC}]$ denotes the soft-pseudo label vector for x_i and all the soft-pseudo labels constitute the soft-pseudo label matrix $Q = [q_1, \cdots, q_n]^T \in [0, 1]^{n \times C}$. We chose the output of the tail classifier rather than the head classifier to construct the soft-pseudo label because in the LT-PLL scenario, the pseudo-labeling will be biased toward the head classes (Hong et al. 2023), especially for the output of the head classifier, as it does not include any special design to deal with the long-tailed problem. In contrast, the prediction of the tail classifier is much more balanced. The soft-pseudo label vector can be regarded as the disambiguated candidate label confidence vector to further improve the network. To initialize the i-th soft-pseudo label, we set $q_{ij} = \frac{1}{|S_i|}$ if the labels belong to the candidate label set, i.e., $j \in S_i$, and $q_{ij} = 0$ otherwise.

Moreover, as q_i assesses the label confidence for the i-th sample, we add that of all the samples together, which could act as the estimated class distribution, i.e.,

$$\mathbb{P}_{train}(y = j) = \frac{1}{n} \sum_{i=1}^{n} q_{ij}, \tag{3}$$

where $P_{train}(y = j)$ records the sample size ratio for the j-th class. As a summary, we use the moving average of model's output to achieve candidate label disambiguation and class distribution estimation synchronously.

Head Classifier and Tail Classifier Construction

To handle the long-tailed distribution of the training samples, as shwon in Figure 2, we propose to construct two classifiers, where the head classifier (resp. tail classifier) is adept in predicting samples in the head class shot (resp. tail class shot). Specifically, we let the head classifier and the tail classifier share the same feature extractor. Then, we minimize the cross-entropies between the output probabilities P^h and P^t of the two classifiers and the corresponding soft-pseudo labels Q, i.e., on a batch of size n_b, the preliminary loss functions of the head classifier and the tail classifier are respectively

$$\mathcal{L}_{pseudo}^h = \frac{1}{n_b} \sum_{i=1}^{n_b} \mathcal{H}(p_i^h, q_i), \tag{4}$$

$$\mathcal{L}_{pseudo}^t = \frac{1}{n_b} \sum_{i=1}^{n_b} \mathcal{H}(p_i^t, q_i). \tag{5}$$

To enable the head classifier (resp. tail classifier) to be good at predicting the samples from the head classes (resp. tail classes), we adopt two different high-confidence sampling strategies for further training of the two classifiers.

Head Classifier Construction. For the head classifier, we use a long-tailed sampling strategy (i.e., sampling according to the estimated distribution of the training data) to select high-confidence samples. Specifically, we use $r_j^h =$

$\alpha(k) \cdot \mathbb{P}_{train}(y = j) \cdot n_b$ to represent the sampling number of the j-th class on a batch, where $\mathbb{P}_{train}(y = j)$ is the estimated sample size ratio by Eq. (3). $\alpha(k) = \alpha_s + (\alpha_e - \alpha_s) \cdot \min(\frac{k}{E}, 1)$ is the ratio function of the epoch number k. α_s, α_e and E are hyper-parameters. As the model training progresses, the output of the model becomes more reliable and the ratio $\alpha(k)$ increases. For each class j, we construct $B^h_j = \{x_i | j = \arg\max_{u \in S_i} q_{iu}, \forall x_i \in B\}$ to represent the samples belonging to class j, where q_i is the soft-pseudo label of x_i. Then the reliable samples set selected for the head classifier is

$$B^h_{re} = \bigcup_{j=1}^{C} BottomK(B^h_j, r^h_j), \tag{6}$$

where $BottomK(B^h_j, r^h_j)$ means selecting the r^h_j samples with the smallest loss $l^h_i = \mathcal{H}(p^h_i, q_i)$ from B^h_j. For these reliable samples, we use mixup (Zhang et al. 2018) to enhance the model's memory of the correspondence between features and labels. Specifically, we construct new training samples by linearly interpolating a sample $x_i \in B^h_{re}$ with another sample $x_j \in B^h_{re}$ randomly: $x^m_i = \phi x_i + (1-\phi)x_j$, $q^m_i = \phi q_i + (1 - \phi)q_j$, where $\phi \sim Beta(\zeta, \zeta)$ and ζ is a hyper-parameter that controls the mixup ratio. Then we define mixup loss as the cross-entropy of predictions of x^m and q^m:

$$\mathcal{L}^h_{mixup} = \frac{1}{|B^h_{re}|} \sum_{i=1}^{|B^h_{re}|} \mathcal{H}(p^{mh}_i, q^m_i), \tag{7}$$

where $p^{mh}_i = softmax(z^h(x^m_i))$ denotes the head classifier predication for x^m_i. Putting them together, the overall training loss for the head classifier becomes

$$\mathcal{L}_h = \mathcal{L}^h_{pseudo} + \beta(k) \cdot \mathcal{L}^h_{mixup}, \tag{8}$$

where a dynamic balancing function $\beta(k) = \min(k/E, 1)$ w.r.t. the epoch number k is employed where E is a preset epoch number, because the output is not reliable in the early stages of training .

Tail Classifier Construction. Different from the head classifier, we adopt a uniform sampling strategy for selecting the high-confidence samples for the tail classifier. Specifically, the sampling number of the j-th class on a batch is $r^t_j = \alpha(k) \cdot \frac{1}{C} \cdot n_b$. To further increase the ability of the tail classifier in handling the samples in the tail classes, we perform logit adjustment (Menon et al. 2021) on the output of the head classifier to guide the selection of high-confidence samples and the training of the tail classifier. The adjusted prediction of head classifier for the i-th sample on the j-th class is

$$p^{adj}_{ij} = \frac{\mathbb{I}(j \in S_i) exp(z^h_j(x_i) - \tau \log \mathbb{P}_{train}(y = j))}{\sum_{u \in S_i} exp(z^h_u(x_i) - \tau \log \mathbb{P}_{train}(y = u))}. \tag{9}$$

To bias the adjusted probabilities towards the tail classes to some extent, we set $\tau > 1$ as the adjustment coefficient. $\mathbb{I}(\cdot)$ is the indicator function that outputs 1 if the condition satisfies while 0, otherwise, which ensures the probabilities on the non-candidate label set are zero. Similarly, we construct the sample set $B^t_j = \{x_i | j = \arg\max_{u \in S_i} p^{adj}_{iu}, \forall x_i \in B\}$ for the j-th class, and select r^t_j samples with the smallest loss $l^t_i = \mathcal{H}(p^t_i, p^{adj}_i)$ to construct the reliable sample set

$$B^t_{re} = \bigcup_{j=1}^{C} BottomK(B^t_j, r^t_j). \tag{10}$$

For the selected reliable samples, we use the cross-entropy loss

$$\mathcal{L}^t_{ce} = \frac{1}{|B^t_{re}|} \sum_{i=1}^{|B^t_{re}|} \mathcal{H}(p^t_i, p^{adj}_i) \tag{11}$$

to make the tail classifier pay more attention to the tail classes. Mixup is also used for reliable samples to ensure the classifier to memorize the connection between features and labels. By the same way, we obtain new samples $x^m_i = \phi x_i + (1 - \phi)x_j$ and corresponding labels $p^{madj}_i = \phi p^{adj}_i + (1 - \phi)p^{adj}_j$. Then the mixup loss for the tail classifier is

$$\mathcal{L}^t_{mixup} = \frac{1}{|B^t_{re}|} \sum_{i=1}^{|B^t_{re}|} \mathcal{H}(p^{mt}_i, p^{madj}_i), \tag{12}$$

where $p^{mt} = softmax(z^t(x^m))$ denotes the tail classifier predication of x^m. Finally, for the tail classifier, the overall loss is

$$\mathcal{L}_t = \mathcal{L}^t_{pseudo} + \beta(k) \cdot (\mathcal{L}^t_{ce} + \mathcal{L}^t_{mixup}), \tag{13}$$

where $\beta(k)$ is identical to that of the head classifier.

Classifier Weight Estimation Module

So far, we have constructed two classifiers with different strengths, so we need to fuse the results of the two classifiers together to get a reasonable prediction on the test set. An intuitive approach is to use the head classifier (resp. tail classifier) to predict the samples from the head classes, (resp. tail classes), but unfortunately the shot belongingness of a sample is agnostic. To solve this problem, we propose a classifier weight estimation (CWE) module that can automatically estimate the shot belongingness (head classes or tail classes) of the samples and allocate the weights for the head classifier and tail classifier to achieve cooperation when making prediction. Specifically, the CWE module takes the predictions of the two classifiers on each sample as input, and computes the corresponding weights of the two classifiers. Since the head classifier is too biased towards the head classes, we calibrate the output of it by

$$\hat{p}^h_{ij} = \frac{exp(z^h_j(x_i) - \log \mathbb{P}_{train}(y = j))}{\sum_{k=1}^{C} exp(z^h_k(x_i) - \log \mathbb{P}_{train}(y = k))}, \tag{14}$$

where $\hat{p}^h_i$ denotes calibrated prediction. We concatenate the predications of the two classifier to get $[\hat{p}^h_i || p^t_i] \in \mathbb{R}^{1 \times 2C}$, and the parameters of the CWE module are represented by $W^f \in \mathbb{R}^{2C \times 2}$. Through an activation function LeakyReLU

Algorithm 1: Training Process the Proposed Method

```
Input: Training dataset $\mathcal{D}$, hyper-parameters $\tau$, $\zeta$ and $\mu$.
Output: Feature extractor $f$, head classifier $g^h$, tail classifier $g^t$, and parameters of CWE module $W^f$.
1:  Initialize the parameters of $f$, $g^h$, $g^t$, and parameters of CWE module $W^f$ randomly;
2:  for $epoch = 1, 2, \cdots$ do
3:      for $batch = 1, 2, \cdots$ do
4:          Get head classifier prediction $P^h$ and tail classifier prediction $P^t$ on a batch of data $B$ of size $n_b$;
5:          Calculate pseudo loss by (4) and (5);
6:          Select reliable samples for head classifier by (6);
7:          Calculate loss of head classifier by (8);
8:          Select reliable samples for tail classifier by (10);
9:          Calculate loss of tail classifier by (13);
10:         Calculate weights of the two classifiers for each sample by (15);
11:         Calculate loss of CWE module by (16);
12:         Obtain the overall loss by summing up the three losses by (17);
13:         Update network parameter via gradient descent;
14:         Update soft-pseudo labels by (2);
15:     end for
16:     Calculate class distribution by (3);
17: end for
```

(Maas, Hannun, and Ng 2013), the weights of two classifiers are denoted as

$$[w_i^h || w_i^t] = softmax(LeakyReLU([\hat{p}_i^h || p_i^t] W^f)), \quad (15)$$

where w_i^h and w_i^t are the weights of head classifier and tail classifier on sample x_i respectively, and $softmax$ is used to normalize the weights. We then minimize the cross-entropy of the fused prediction and the soft pseudo-labels

$$\mathcal{L}_f = \frac{1}{n_b} \sum_{i=1}^{n_b} \mathcal{H}(p_i^f, q_i), \quad (16)$$

to infer the learnable parameters of the CEW module, where $p_i^f = w_i^h \cdot \hat{p}_i^h + w_i^t \cdot p_i^t$ denotes the fused prediction.

Joint Optimization

By adding these losses together, the final loss becomes:

$$\mathcal{L} = \mathcal{L}_h + \mathcal{L}_t + \mathcal{L}_f. \quad (17)$$

Figure 2 shows the overall framework of our model with gradient forward and backpropagation processes. The pseudocode is shown in Algorithm 1.

Prediction

For a new test sample x, we first compute the calibrated prediction $\hat{p}^h$ of head classifier with (14). Then, through $\hat{p}^h$ and the prediction of the tail classifier $p^t = softmax(z^t(x))$, we compute the corresponding weights w^h and w^t by the proposed CWE module in (15), and we have $p = w^h \cdot \hat{p}^h + w^t \cdot p^t$. Finally, the prediction for x is obtained by $y_{pred} = \arg\max p$.

Experiments and Analysis

Experiment Setup

Datasets. Following the previous works (Wang et al. 2022a; Hong et al. 2023), we evaluated our method on the long-tailed versions of CIFAR10 and CIFAR100 and a real-world LT-PLL dataset PASCAL VOC. To generate the long-tailed training sets, n_j samples for the j-th class in the CIFAR10 and CIFAR100 training sets were randomly selected according to a pre-determined imbalance ratio $\gamma = \frac{n_1}{n_C}$, where $n_j = n_1 \cdot \gamma^{\frac{j-1}{C-1}}$ and C is the number of classes. Specifically, $n_1 = 5000$ and $C = 10$ for CIFAR10, and $n_1 = 500$ and $C = 100$ for CIFAR100. We applied different imbalance ratios to evaluate our method, with $\gamma \in \{100, 150, 200, 250\}$ for CIFAR10 and $\gamma \in \{20, 50, 100, 150\}$ for CIFAR100. Following (Wang et al. 2022a,b), we constructed the candidate label set for each sample by manually flipping negative labels $\overline{y} \neq y$ to false-positive labels with probability $\eta = P(\overline{y} \in S_i | \overline{y} \neq y)$. We considered $\eta \in \{0.3, 0.5\}$ for CIFAR10 and $\eta \in \{0.05, 0.1\}$ for CIFAR100. The real-world LT-PLL dataset PASCAL VOC is constructed from PASCAL VOC 2007 (Everingham et al. 2010) by RECORDS (Hong et al. 2023).

Compared methods. We compared our method with five SOTA PLL methods including LW (Wen et al. 2021), CC (Feng et al. 2020), PRODEN (Lv et al. 2020), CAVL (Zhang et al. 2022) and PiCO (Wang et al. 2022b), and two recent LT-PLL methods including Solar (Wang et al. 2022a) and RECORDS (Hong et al. 2023). The hyper-parameters of the compared methods were set according to the original papers.

Implementation details. We used the 18-layer ResNet as the backbone. For all methods, they were trained for 800 epochs using SGD as the optimizer with momentum of 0.9 and weight decay of 0.001. The initial learning rate was set to 0.01, and divided by 10 after 700 epochs. We set batch size to 256 for CIFAR10 and CIFAR100 and 64 for PASCAL VOC. For all methods on PASCAL VOC, we loaded the pre-trained weights from ImageNet for the feature extractor to improve the training efficiency. For our method, we fixed the mixup hyper-parameter $\zeta = 4$, $\mu = 0.6$ and linearly ramp up reliable sample ratio α from 0.2 to 0.6 in first 50 epochs, dynamic balancing coefficient β from 0 to 0.9 in first 50 epochs for all experiments. When imbalance ratio $\gamma \geq 200$, the logit adjustment coefficient τ was set to 1.2, otherwise, $\tau = 2$. RECORDS is an plug-in approach for PLL to deal with LT-PLL. We used the best setting provided by the original paper, i.e., CORR+RECORDS+Mixup, for all RECORDS experiments in this paper. All the experiments were conducted three times under the same random seed, and the model with the best performance on the validation dataset is taken as the final model.

Main Results

Our method achieves SOTA classification accuracy. As shown in Table 1, on the CIFAR10 and CIFAR100 datasets with different flipping probability η and different imbalance ratios γ, our method outperforms all the compared methods by a large margin. For example, on CIFAR10 with $\eta = 0.3$,

Method		CIFAR10							
		$\eta = 0.3$				$\eta = 0.5$			
		$\gamma = 100$	$\gamma = 150$	$\gamma = 200$	$\gamma = 250$	$\gamma = 100$	$\gamma = 150$	$\gamma = 200$	$\gamma = 250$
PLL	LW (ICML 2021)	38.98 ± 3.39	41.00 ± 0.34	38.58 ± 3.63	36.04 ± 3.09	20.22 ± 1.42	20.45 ± 0.93	19.72 ± 0.58	19.99 ± 0.59
	CC (NeurIPS 2020)	60.45 ± 1.06	54.96 ± 3.54	51.87 ± 0.57	50.69 ± 0.98	47.64 ± 1.07	44.20 ± 0.89	43.02 ± 0.42	40.22 ± 0.75
	PRODEN (ICML 2020)	54.70 ± 1.07	51.00 ± 1.45	48.49 ± 0.41	45.19 ± 1.40	42.90 ± 1.49	39.33 ± 0.97	38.18 ± 0.87	36.77 ± 0.92
	CAVL (ICLR 2022)	39.51 ± 0.27	39.09 ± 0.61	39.16 ± 0.75	38.27 ± 0.12	37.51 ± 1.07	34.61 ± 2.27	34.81 ± 2.02	33.28 ± 1.74
	PiCO (ICLR 2022)	72.81 ± 0.37	69.29 ± 0.99	65.05 ± 1.44	64.04 ± 1.45	67.50 ± 2.87	62.95 ± 2.83	57.55 ± 0.73	53.38 ± 3.45
LT-PLL	RECORDS (ICLR 2023)	77.95 ± 0.36	73.63 ± 0.55	$\underline{71.67 \pm 0.57}$	66.73 ± 0.68	74.05 ± 1.11	67.74 ± 1.30	63.75 ± 0.47	58.61 ± 2.35
	SoLar (NeurIPS 2022)	$\underline{79.49 \pm 0.50}$	$\underline{74.59 \pm 0.63}$	70.74 ± 0.47	$\underline{68.24 \pm 1.15}$	$\underline{75.71 \pm 1.26}$	$\underline{70.21 \pm 2.92}$	$\underline{64.25 \pm 2.02}$	$\underline{60.57 \pm 2.75}$
	Ours	**85.66 ± 1.44**	**82.46 ± 1.09**	**80.57 ± 1.40**	**78.11 ± 2.28**	**83.35 ± 2.24**	**79.79 ± 2.53**	**77.71 ± 1.12**	**72.37 ± 1.74**

Method		CIFAR100							
		$\eta = 0.05$				$\eta = 0.1$			
		$\gamma = 20$	$\gamma = 50$	$\gamma = 100$	$\gamma = 150$	$\gamma = 20$	$\gamma = 50$	$\gamma = 100$	$\gamma = 150$
PLL	LW (ICML 2021)	36.58 ± 0.48	28.93 ± 0.48	25.41 ± 0.53	23.50 ± 0.86	27.14 ± 0.34	22.89 ± 0.72	20.06 ± 1.08	17.72 ± 0.09
	CC (NeurIPS 2020)	41.40 ± 1.34	32.92 ± 0.73	27.81 ± 0.76	25.94 ± 0.25	32.50 ± 1.33	26.42 ± 0.84	22.68 ± 0.33	20.37 ± 1.14
	PRODEN (ICML 2020)	38.68 ± 0.18	31.80 ± 0.18	27.68 ± 0.03	24.97 ± 0.03	30.99 ± 0.29	24.18 ± 0.07	21.69 ± 0.14	19.81 ± 0.01
	CAVL (ICLR 2022)	27.55 ± 0.46	23.49 ± 0.79	19.94 ± 0.20	19.40 ± 0.55	18.29 ± 0.99	15.32 ± 0.26	14.42 ± 0.81	14.36 ± 0.50
	PiCO (ICLR 2022)	48.25 ± 1.52	40.12 ± 0.37	35.45 ± 0.85	33.09 ± 0.33	39.64 ± 0.71	33.75 ± 1.77	29.40 ± 0.55	28.15 ± 0.52
LT-PLL	RECORDS (ICLR 2023)	$\underline{57.60 \pm 1.99}$	$\underline{49.04 \pm 1.57}$	$\underline{43.36 \pm 1.95}$	$\underline{39.83 \pm 0.34}$	$\underline{54.73 \pm 0.80}$	$\underline{45.47 \pm 0.74}$	$\underline{40.48 \pm 0.23}$	$\underline{37.37 \pm 1.21}$
	SoLar (NeurIPS 2022)	57.13 ± 1.46	47.53 ± 1.22	41.94 ± 1.92	39.13 ± 0.30	52.56 ± 1.62	42.49 ± 0.59	36.36 ± 1.48	33.81 ± 0.43
	Ours	**61.13 ± 1.15**	**53.26 ± 1.90**	**47.49 ± 0.63**	**44.83 ± 1.28**	**60.53 ± 1.45**	**51.26 ± 1.31**	**46.24 ± 1.70**	**42.63 ± 1.45**

Table 1: Accuracy comparisons on CIFAR10 and CIFAR100 under various flipping probability η and imbalance ratio γ. Bold and underlined indicate the best and second best results, respectively. Accuracies are presented in percentage (%) form.

Method		CIFAR10 ($\eta = 0.5, \gamma = 150$)				CIFAR100 ($\eta = 0.1, \gamma = 50$)			
		Overall	Many	Medium	Few	Overall	Many	Medium	Few
PLL	LW (ICML 2021)	20.87	50.95	0.00	0.00	22.17	55.70	11.15	0.00
	CC (NeurIPS 2020)	43.56	79.93	37.6	0.43	26.63	57.36	21.53	1.15
	PRODEN (ICML 2020)	40.01	79.12	27.87	0.00	24.14	53.64	18.41	0.55
	CAVL (ICLR 2022)	36.43	80.25	14.43	0.00	15.47	45.33	1.50	0.00
	PiCO (ICLR 2022)	64.75	80.37	71.73	36.93	34.87	62.79	34.24	7.61
LT-PLL	RECORDS (ICLR 2023)	$\underline{67.10}$	88.17	$\underline{71.10}$	$\underline{35.00}$	$\underline{46.31}$	$\underline{76.09}$	$\underline{45.65}$	8.52
	SoLar (NeurIPS 2022)	66.85	$\underline{90.50}$	68.73	33.43	41.81	68.85	41.53	$\underline{15.06}$
	Ours	**82.22**	**91.58**	**80.83**	**71.13**	**50.52**	**76.21**	**53.21**	**22.06**

Table 2: Fine-grained analysis on CIFAR10 with $\eta = 0.5$ and $\gamma = 150$ and CIFAR100 with $\eta = 0.1$ and $\gamma = 50$. Many/Medium/Few corresponds to three partitions on the long-tailed data. Bold and underlined indicate the best and second best results, respectively. Accuracies are presented in percentage (%) form.

our method improves upon the best compared method by **6.17%**, **7.87%**, **8.90%** and **9.87%** on four different imbalance ratios $\gamma = 100, 150, 200$ and 250 respectively. Especially, on CIFAR10 with $\eta = 0.5$ and $\gamma = 200$, our method performs **13.46%** better than the best compared method.

Our method achieves SOTA performance on different class shots. To compare the performance of different methods on classes of different shots, we divide the CIFAR10 ($\eta = 0.5$, $\gamma = 150$) and the CIFAR100 ($\eta = 0.1$, $\gamma = 50$) into three groups according to the number of samples per class: Many/Medium/Few. Many denotes the classes with the top $\frac{1}{3}$ of samples, Few are the classes with the bottom $\frac{1}{3}$ of samples, and the remaining classes belong to Medium. As shown in Table 2, our method outperforms the compared methods not only on the overall accuracy but also on all the class shots. It is worth noting that on CIFAR100 ($\eta = 0.1$, $\gamma = 50$), both SoLar and RECORDS fall into a trade-off dilemma: SoLar is skilled in the tail classes and has lower performance on the Many group and Medium group than RECORDS by 7.24% and 4.12%, respectively, while RECORDS pays more attention to the head classes and has lower accuracy than SoLar by 6.54% on the Few group. However, our method overcomes this difficulty and maintains high accuracy on both the tail classes (53.21% on Medium group and 22.06% on Few group) and the head classes (76.21% on Many group). Similar phenomenon also occurs on CIFAR10 ($\eta = 0.5$, $\gamma = 150$). Those observations explain why our method far exceeds the other compared methods in terms of overall accuracy.

Further Analysis

Skilled in real-world LT-PLL. To further verify the superiority of our method, we evaluated different methods on one real-world LT-PLL dataset PASCAL VOC. As indicated in Table 3, our method achieves the highest accuracy, outperforming the best compared method by **5.62%** on the whole dataset, and by **11.79%** on the Few group.

More accurate class distribution estimation. Figure 3 compares the class distribution estimation ability of all the LT-PLL methods, where we can find that our method gives

Method		Result on PASCAL VOC			
		Overall	Many	Medium	Few
PLL	LW (ICML 2021)	18.70	50.75	9.93	0.00
	CC (NeurIPS 2020)	35.78	68.17	28.00	15.79
	PRODEN (ICML 2020)	38.30	63.17	32.93	22.36
	CAVL (ICLR 2022)	30.12	57.92	26.79	9.64
	PiCO (ICLR 2022)	49.05	70.50	45.79	33.93
LT-PLL	RECORDS (ICLR 2023)	59.70	68.92	<u>68.57</u>	42.93
	SoLar (NeurIPS 2022)	<u>61.83</u>	**76.33**	65.64	<u>45.57</u>
	Ours	**67.45**	<u>75.83</u>	**70.36**	**57.36**

Table 3: Classification accuracy of each method on real world TL-PLL dataset PASCAL VOC. Many/Medium/Few corresponds to three partitions on the long-tailed data. Bold and underlined indicate the best and second best results, respectively. Accuracies are presented in percentage (%) form.

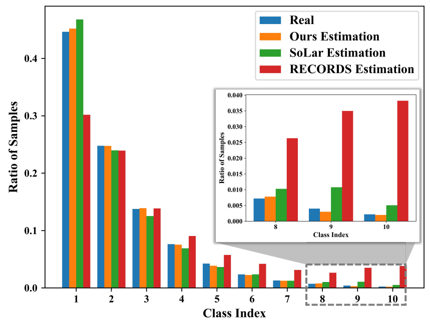


Figure 3: The real/estimated class distribution on CIFAR10 dataset ($\eta = 0.3, \gamma = 200$). The class distribution estimated by our method is very close to the real one, while SoLar and RECORDS fails to recover real class distribution especially in tail classes.

a more accurate estimation of the real class distribution, especially for the tail classes.

CWE correctly estimates the weights for two classifiers. Figure 4 shows our model allocates more weights for the head classifier than the tail classifier when predicting samples from the head classes, and the opposite is true for tail classes, indicating that the proposed CWE module can roughly estimates the class group of the samples and assign the corresponding weights for two classifiers.

Classifiers cooperation is important. We compared our method with two variants:1) *Ours with Only Tail (V1)* which only keeps the tail classifier and 2) *Ours with Only Head (V2)* which only keeps the head classifier. As depicted in Table 4, V1 and V2 are biased towards certain class shot leading to suboptimal overall performance (V1: 69.31%, V2: 68.12%). The CWE module assigns appropriate weights to classifiers based on sample characteristics, achieving 12.66% and 13.85% accuracy improvement on V1 and V2 respectively.

All the components contribute to the proposed model. We conducted additional experiments to assess the effectiveness of each part. Specifically, we tested *Ours w/o Pesudo* by removing the soft-pseudo label losses in Eq. (4) and Eq. (5); *Ours w/o Mixup* by eliminating all terms that used mixup; *Ours w/o Logit Adjustment* by removing the logit adjustment and using the soft-pseudo labels instead; and *Ours w/o Sample Selection* by using all samples as reliable samples without any reliable sample selection for the two classifiers. From Table 4, we can see that all the ingredients of our method contribute to the performance improvement. Especially in the ablation experiment *Ours w/o Pesudo*, the overall performance has a severe decline of 18.48%, highlighting the importance of the soft-pseudo labels for guiding the model training.

Method	CIFAR10 ($\eta = 0.3, \gamma = 200$)			
	Overall	Many	Medium	Few
Ours	81.97	92.43	78.10	76.67
Ours with Only Tail (V1)	69.31	44.00	77.90	83.17
Ours with Only Head (V2)	68.12	96.20	66.85	41.73
Ours w/o Pesudo (V3)	63.49	57.10	76.97	51.90
Ours w/o Mixup (V4)	72.32	93.83	67.50	57.23
Ours w/o Logit Adjustment (V5)	73.79	82.43	74.22	64.57
Ours w/o Sample Selection (V6)	74.25	83.67	73.03	66.47

Table 4: Ablation study of our method on LT-PLL datasets CIFAR10 ($\eta = 0.3, \gamma = 200$). Accuracies are presented in percentage (%) form.

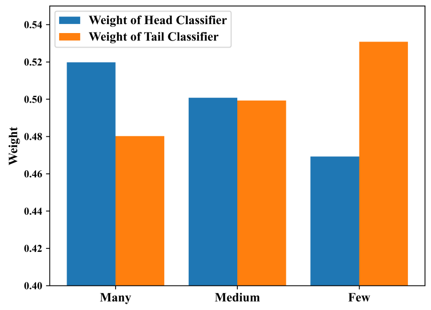


Figure 4: The mean weights of *head classifier* and *tail classifier* for the testing samples of CIFAR10 ($\eta = 0.3, \gamma = 200$) estimated by the CWE module in different shots. During the prediction, the samples belonging to the head classes have larger weights in the head classifier, while the samples belonging to the tail classes have larger weights in the tail classifier.

Conclusion

In this work, we have present a novel model to solve the performance dilemma of the previous LT-PLL methods. Specifically, we trained two classifiers that excel in different class shots and combine their predictions automatically by an effective CWE module. We conducted extensive experiments on both synthetic and real-world LT-PLL datasets, suggesting the salient performance advantage of our method. Moreover, the class distribution estimated by our method is more accurate than the SOTA methods, and the classifier weights estimated by CWE module can reflect the class shot belongingness of samples.

References

Chen, C.; Patel, V. M.; and Chellappa, R. 2018. Learning from Ambiguously Labeled Face Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(7): 1653–1667.

Cour, T.; Sapp, B.; and Taskar, B. 2011. Learning from Partial Labels. *Journal of Machine Learning Research*, 12: 1501–1536.

Everingham, M.; Gool, L. V.; Williams, C. K. I.; Winn, J. M.; and Zisserman, A. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2): 303–338.

Feng, L.; Lv, J.; Han, B.; Xu, M.; Niu, G.; Geng, X.; An, B.; and Sugiyama, M. 2020. Provably Consistent Partial-Label Learning. In *Advances in Neural Information Processing Systems 33*, 10948–10960.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. B. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9726–9735.

Hong, F.; Yao, J.; Zhou, Z.; Zhang, Y.; and Wang, Y. 2023. Long-Tailed Partial Label Learning via Dynamic Rebalancing. In *The Eleventh International Conference on Learning Representations*.

Hüllermeier, E.; and Beringer, J. 2006. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 10(5): 419–439.

Jia, Y.; Jiang, J.; and Wang, Y. 2023. Semantic Dissimilarity Guided Locality Preserving Projections for Partial Label Dimensionality Reduction. In *29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 964–973.

Jia, Y.; Si, C.; and Zhang, M. 2023. Complementary Classifier Induced Partial Label Learning. In *29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Jia, Y.; Yang, F.; and Dong, Y. 2023. Partial Label Learning with Dissimilarity Propagation guided Candidate Label Shrinkage. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Kang, B.; Xie, S.; Rohrbach, M.; Yan, Z.; Gordo, A.; Feng, J.; and Kalantidis, Y. 2020. Decoupling Representation and Classifier for Long-Tailed Recognition. In *8th International Conference on Learning Representations*.

Liu, W.; Wang, L.; Chen, J.; Zhou, Y.; Zheng, R.; and He, J. 2021. A Partial Label Metric Learning Algorithm for Class Imbalanced Data. In *Asian Conference on Machine Learning*, volume 157, 1413–1428.

Luo, J.; and Orabona, F. 2010. Learning from Candidate Labeling Sets. In *Advances in Neural Information Processing Systems 23*, 1504–1512.

Lv, J.; Xu, M.; Feng, L.; Niu, G.; Geng, X.; and Sugiyama, M. 2020. Progressive Identification of True Labels for Partial-Label Learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, 6500–6510.

Maas, A. L.; Hannun, A. Y.; and Ng, A. Y. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning*, volume 30.

Menon, A. K.; Jayasumana, S.; Rawat, A. S.; Jain, H.; Veit, A.; and Kumar, S. 2021. Long-tail learning via logit adjustment. In *9th International Conference on Learning Representations*.

Nguyen, N.; and Caruana, R. 2008. Classification with partial labels. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 551–559.

Panis, G.; and Lanitis, A. 2014. An Overview of Research Activities in Facial Age Estimation Using the FG-NET Aging Database. In *European Conference on Computer Vision Workshops*, 737–750.

Wang, D.; Zhang, M.; and Li, L. 2022. Adaptive Graph Guided Disambiguation for Partial Label Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12): 8796–8811.

Wang, H.; Xia, M.; Li, Y.; Mao, Y.; Feng, L.; Chen, G.; and Zhao, J. 2022a. SoLar: Sinkhorn Label Refinery for Imbalanced Partial-Label Learning. In *Advances in Neural Information Processing Systems 35*, volume 35, 8104–8117.

Wang, H.; Xiao, R.; Li, Y.; Feng, L.; Niu, G.; Chen, G.; and Zhao, J. 2022b. PiCO: Contrastive Label Disambiguation for Partial Label Learning. In *The Tenth International Conference on Learning Representations*.

Wang, J.; and Zhang, M. 2018. Towards Mitigating the Class-Imbalance Problem for Partial Label Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2427–2436.

Wen, H.; Cui, J.; Hang, H.; Liu, J.; Wang, Y.; and Lin, Z. 2021. Leveraged Weighted Loss for Partial Label Learning. In *Proceedings of the 38th International Conference on Machine Learning*, 11091–11100.

Wu, D.; Wang, D.; and Zhang, M. 2022. Revisiting Consistency Regularization for Deep Partial Label Learning. In *International Conference on Machine Learning*, 24212–24225.

Zhang, F.; Feng, L.; Han, B.; Liu, T.; Niu, G.; Qin, T.; and Sugiyama, M. 2022. Exploiting Class Activation Value for Partial-Label Learning. In *The Tenth International Conference on Learning Representations*.

Zhang, H.; Cissé, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. In *6th International Conference on Learning Representations*.

Zhang, Y.; Kang, B.; Hooi, B.; Yan, S.; and Feng, J. 2023. Deep long-tailed learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9): 10795 – 10816.