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

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

\textsuperscript{1}School of Computer Science and Engineering, Southeast University
\textsuperscript{2}Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
\textsuperscript{3}Shenzhen Key Laboratory of Advanced Machine Learning and Applications, School of Mathematical Science, Shenzhen University, Shenzhen, China
\textsuperscript{4}Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen, China
\textsuperscript{5}Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China

\textsuperscript{*Corresponding author.}

\textsuperscript{1,2}{\{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.

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 is a long-tailed distribution.

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.
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 stemmed 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 (H¨ullermeier and Beringer 2006; Nguyen and Caruana 2008) first dynamically maintains a prototype feature for estimation of class distribution, and then uses logit-adjustment (Menon et al. 2021) to remedy the bias stemmed from long tail effect. 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 $\theta$, 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 $P^h = [p^h_1, \cdots, p^h_n]^T = [p^h_i]_{n \times C}$ and $P^t = [p^t_1, \cdots, p^t_n]^T = [p^t_i]_{n \times C}$ to denote the probability prediction matrix of the head classifier and tail classifier, where $p^h_i = \text{softmax}(z^h(x_i))$ and $p^t_i = \text{softmax}(z^t(x_i))$. For convenience, a batch of samples $B$ of size $n_b$ is...
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.,

$$P_{\text{pseudo}}(y = j) = \frac{1}{n} \sum_{i=1}^{n} q_{ij}, \quad (3)$$

where $P_{\text{train}}(y = j)$ 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 shown 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

$$\mathcal{L}_{\text{pseudo}}^h = \frac{1}{n_b} \sum_{i=1}^{n_b} \mathcal{H}(p_i^h, q_i), \quad (4)$$
$$\mathcal{L}_{\text{pseudo}}^t = \frac{1}{n_b} \sum_{i=1}^{n_b} \mathcal{H}(p_i^t, q_i). \quad (5)$$

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 accord-
ing to the estimated distribution of the training data) to select high-confidence samples. Specifically, we use \( r_h^j = \alpha(k) \cdot \mathbb{P}_{train}(y = j) \cdot n_h \) 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). \( \alpha(k) = \alpha_x + (\alpha_e - \alpha_x) \cdot \min(k, 1) \) is the ratio function of the epoch number \( k \), \( \alpha_x \), \( \alpha_e \) 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 \( B_j^h = \{ x_i | j = \arg \max_{a \in S_i} p_{ih}^{adj}, \forall x_i \in B \} \) 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

\[
B_{re}^h = \bigcup_{j=1}^{C} BottomK(B_j^h, r_j^h),
\]

where \( BottomK(B_j^h, r_j^h) \) means selecting the \( r_j^h \) samples with the smallest loss \( l_i^h = \mathcal{H}(p_i^h, q_i) \) from \( B_j^h \). 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 \( x_i \in B_{re}^h \) with another sample \( x_j \in B_{re}^h \) randomly: \( x_i^{m} = \phi x_i + (1 - \phi) x_j \), \( q_i^{m} = \phi q_i + (1 - \phi) q_j \), where \( \phi \sim \text{Beta}(\zeta, \zeta) \) and \( \zeta \) is a hyper-parameter that controls the mixup ratio. Then we define mixup loss as the cross-entropy of predictions of \( x^m \) and \( q^m \):

\[
\mathcal{L}_{mixup}^h = \frac{1}{|B_{re}^h|} \sum_{i=1}^{|B_{re}^h|} \mathcal{H}(p_i^{mh}, q_i^{m}),
\]

where \( p_i^{mh} = \text{softmax}(z^h(x_i^{m})) \) denotes the head classifier prediction for \( x_i^{m} \). Putting them together, the overall training loss for the head classifier becomes

\[
\mathcal{L}_h = \mathcal{L}_{pseudo}^h + \beta(k) \cdot \mathcal{L}_{mixup}^h,
\]

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 \( r_j^t = \alpha(k) \cdot \frac{1}{r_h^j} \cdot n_t \). 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

\[
p_{ij}^{adj} = \frac{\mathbb{I}(j \in S_i) \exp(z^h(x_i)) - \tau \log \mathbb{P}_{train}(y = j))}{\sum_{a \in S_i} \exp(z^h_a(x_i)) - \tau \log \mathbb{P}_{train}(y = a))}.
\]

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 \( B_j^t = \{ x_i | j = \arg \max_{a \in S_j} p_{ij}^{adj}, \forall x_i \in B \} \) for the \( j \)-th class, and select \( r_j^t \) samples with the smallest loss \( l_i^t = \mathcal{H}(p_i^t, p_i^{adj}) \) to construct the reliable sample set

\[
B_{re}^t = \bigcup_{j=1}^{C} BottomK(B_j^t, r_j^t).
\]

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

\[
\mathcal{L}_{ce}^t = \frac{1}{|B_{re}^t|} \sum_{i=1}^{|B_{re}^t|} \mathcal{H}(p_i^t, p_i^{adj}),
\]

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 \( x_i^{mt} = \phi x_i + (1 - \phi) x_j \) and corresponding labels \( p_i^{mt} = \phi p_i^{adj} + (1 - \phi) p_j^{adj} \). Then the mixup loss for the tail classifier is

\[
\mathcal{L}_{mixup}^t = \frac{1}{|B_{re}^t|} \sum_{i=1}^{|B_{re}^t|} \mathcal{H}(p_i^{mt}, p_i^{adj}),
\]

where \( p_i^{mt} = \text{softmax}(z^t(x_i^{mt})) \) denotes the tail classifier prediction of \( x_i^{mt} \). Finally, for the tail classifier, the overall loss is

\[
\mathcal{L}_t = \mathcal{L}_{pseudo}^t + \beta(k) \cdot \left( \mathcal{L}_{ce}^t + \mathcal{L}_{mixup}^t \right),
\]

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

\[
\hat{p}_{ij}^h = \frac{\exp(z_j^h(x_i)) - \log \mathbb{P}_{train}(y = j))}{\sum_{k=1}^{C} \exp(z_j^h(x_i)) - \log \mathbb{P}_{train}(y = k))},
\]

where \( \hat{p}_{ij}^h \) denotes calibrated prediction. We concatenate the predictions of the two classifier to get \( [p_i^h, \hat{p}_i^h] \in \mathbb{R}^{1 \times 2C} \), and the parameters of the CWE module are represented by
Algorithm 1: Training Process the Proposed Method

Input: Training dataset \( D \), 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 \( = 1, 2, \cdots, n_b \) 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

\( W^f \in \mathbb{R}^{2C \times 2} \). Through an activation function LeakyReLU (Maas, Hannun, and Ng 2013), the weights of two classifiers are denoted as

\[
[w^h_i || w^t_i] = \text{softmax}(\text{LeakyReLU}([p^h_i || p^t_i]W^f)),
\]

where \( w^h_i \) and \( w^t_i \) are the weights of head classifier and tail classifier on sample \( x_i \), respectively, and \( \text{softmax} \) is used to normalize the weights. We then minimize the cross-entropy of the fused prediction and the soft-pseudo-labels

\[
\mathcal{L}_f = \frac{1}{n_b} \sum_{i=1}^{n_b} \mathcal{H}(p^f_i, q_i),
\]

to infer the learnable parameters of the CEW module, where \( p^f_i = w^h_i \cdot \hat{p}^h_i + w^t_i \cdot \hat{p}^t_i \) denotes the fused prediction.

Joint Optimization

By adding these losses together, the final loss becomes:

\[
\mathcal{L} = \mathcal{L}_h + \mathcal{L}_t + \mathcal{L}_f.
\]

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 \( p^t = \text{softmax}(z^t(x)) \), 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 \( y_{\text{pred}} = \arg \max p \).

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. For the long-tailed sets, \( n_j \) samples for the \( j \)-th class in the CIFAR training sets were randomly selected according to a pre-determined imbalance ratio \( \gamma = \frac{n_j}{n_C} \), where \( n_j \) and \( n_C \) are 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 \( \gamma \in \{0.1, 0.5, 1.0, 2.0, 3.0\} \) for CIFAR10 and \( \gamma \in \{0.05, 0.1, 0.3\} \) 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 \( \alpha \) from 0.2 to 0.6 in first 50 epochs, dynamic balancing coefficient \( \beta \) from 0 to 0.9 in first 50 epochs for all experiments. When imbalance ratio \( \gamma \geq 200 \), the logit adjustment coefficient \( \tau \) was set to 1.2, otherwise, \( \tau = 2 \). RECORDS is a 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 \( \eta \) and different imbalance ratios \( \gamma \), our method outperforms all the compared methods by a large margin. For example, on CIFAR10 with \( \eta = 0.3 \), our method improves upon the best compared method by
Table 1: Accuracy comparisons on CIFAR10 and CIFAR100 under various flipping probability $\eta$ and imbalance ratio $\gamma$. Bold and underlined indicate the best and second best results, respectively. Accuracies are presented in percentage ($\%$) form.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR10 ($\eta = 0.3$)</th>
<th>CIFAR100 ($\eta = 0.1$, $\gamma = 50$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\gamma = 10$</td>
<td>$\gamma = 150$</td>
</tr>
<tr>
<td>PLL</td>
<td>38.98 $\pm$ 3.39</td>
<td>41.00 $\pm$ 0.34</td>
</tr>
<tr>
<td></td>
<td>60.45 $\pm$ 1.06</td>
<td>54.96 $\pm$ 3.54</td>
</tr>
<tr>
<td></td>
<td>54.70 $\pm$ 1.07</td>
<td>51.00 $\pm$ 1.45</td>
</tr>
<tr>
<td>PLL</td>
<td>39.51 $\pm$ 0.27</td>
<td>39.09 $\pm$ 0.61</td>
</tr>
<tr>
<td></td>
<td>72.81 $\pm$ 0.37</td>
<td>69.29 $\pm$ 0.99</td>
</tr>
<tr>
<td>LT-PLL</td>
<td>77.95 $\pm$ 0.36</td>
<td>73.65 $\pm$ 0.57</td>
</tr>
<tr>
<td>RECORDS</td>
<td>79.49 $\pm$ 0.50</td>
<td>74.59 $\pm$ 0.63</td>
</tr>
<tr>
<td>SoLar</td>
<td>85.66 $\pm$ 1.44</td>
<td>82.46 $\pm$ 1.09</td>
</tr>
<tr>
<td>Ours</td>
<td>86.56 $\pm$ 1.44</td>
<td>82.46 $\pm$ 1.09</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR10 ($\eta = 0.5$, $\gamma = 150$)</th>
<th>CIFAR100 ($\eta = 0.1$, $\gamma = 50$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overall</td>
<td>Many</td>
</tr>
<tr>
<td>PLL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LW</td>
<td>20.87</td>
<td>50.95</td>
</tr>
<tr>
<td>CC</td>
<td>43.56</td>
<td>79.93</td>
</tr>
<tr>
<td>PRODEN</td>
<td>40.01</td>
<td>79.12</td>
</tr>
<tr>
<td>CAVL</td>
<td>36.43</td>
<td>80.25</td>
</tr>
<tr>
<td>PiCO</td>
<td>64.75</td>
<td>80.37</td>
</tr>
<tr>
<td>LT-PLL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORDS</td>
<td>67.10</td>
<td>88.17</td>
</tr>
<tr>
<td>SoLar</td>
<td>66.85</td>
<td>90.50</td>
</tr>
<tr>
<td>Ours</td>
<td>82.22</td>
<td>91.58</td>
</tr>
</tbody>
</table>

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 different classes of 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
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.
 Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant 62106044, Grant 62176160, and Grant 62225602), in part by the Natural Science Foundation of Jiangsu Province under Grant BK20210221 and in part by the Guangdong Basic and Applied Basic Research Foundation (Grant 2022A1515010791).

References


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*.


