SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2025, Vol. 68, Iss. 5, 152101:1–152101:16
https://doi.org/10.1007/s11432-023-4068-1

AsyCo: an asymmetric dual-task co-training model for partial-label learning

Beibei LI[1], Yiyuan ZHENG[2,3], Beihong JIN[2,3], Tao XIANG[1], Haobo WANG[4] & Lei FENG[5*]

[1]*College of Computer Science, Chongqing University, Chongqing 400044, China*
[2]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100044, China*
[3]*University of Chinese Academy of Sciences, Beijing 100044, China*
[4]*School of Software Technology, Zhejiang University, Hangzhou 310027, China*
[5]*School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore*

Received 16 July 2023/Revised 4 October 2023/Accepted 8 March 2024/Published online 16 January 2025

Abstract Partial-label learning (PLL) is a typical problem of weakly supervised learning, where each training instance is annotated with a set of candidate labels. Self-training PLL models achieve state-of-the-art performance but suffer from error accumulation problems caused by mistakenly disambiguated instances. Although co-training can alleviate this issue by training two networks simultaneously and allowing them to interact with each other, most existing co-training methods train two structurally identical networks with the same task, i.e., are symmetric, rendering it insufficient for them to correct each other due to their similar limitations. Therefore, in this paper, we propose an asymmetric dual-task co-training PLL model called AsyCo, which forces its two networks, i.e., a disambiguation network and an auxiliary network, to learn from different views explicitly by optimizing distinct tasks. Specifically, the disambiguation network is trained with a self-training PLL task to learn label confidence, while the auxiliary network is trained in a supervised learning paradigm to learn from the noisy pairwise similarity labels that are constructed according to the learned label confidence. Finally, the error accumulation problem is mitigated via information distillation and confidence refinement. Extensive experiments on both uniform and instance-dependent partially labeled datasets demonstrate the effectiveness of AsyCo.

Keywords machine learning, weakly supervised learning, partial-label learning, co-training models, candidate label sets

Citation Li B B, Zheng Y Y, Jin B H, et al. AsyCo: an asymmetric dual-task co-training model for partial-label learning. Sci China Inf Sci, 2025, 68(5): 152101, https://doi.org/10.1007/s11432-023-4068-1

1 Introduction

Training deep neural networks via supervised learning requires massive accurately-annotated data, which are, however, expensive to collect. To overcome this problem, weakly supervised learning [1–4] has been widely studied in recent years. Partial-label learning (PLL) [5, 6] is a typical type of weakly supervised learning with inaccurate supervision, which assumes that each training instance is annotated with a candidate label set that contains the ground-truth label. As shown in Figure 1, the visual resemblance between raccoons and Ailurus fulgens makes it challenging for annotators to confidently pinpoint the exact animal depicted in the images. As a result, they assign multiple candidate labels to each image, leading to partially labeled instances. Since label ambiguity is pervasive in data annotations, PLL has been widely applied in various real-world applications, such as automatic image annotation [7] and multimedia content analysis [8].

Recent research on PLL has primarily concentrated on identification-based methods, which regard the ground-truth label as a latent variable and try to recognize the ground-truth label by conducting label disambiguation. To this end, various techniques have been employed, such as maximum margin [9], graph models [10–13], expectation-maximum algorithm [14], contrastive learning [15], and consistency regularization [16]. Among these, self-training deep models [17–19] have emerged as a promising approach, which learns label confidence vectors and trains the models with them iteratively, and achieved state-of-the-art performance [16].

* Corresponding author (email: lfengqaq@gmail.com)

Figure 1 (Color online) Two examples of partially labeled instances. Due to the visual similarity between Ailurus fulgens and raccoons, the two images are both annotated with "Ailurus fulgens" and "raccoons".

However, self-training PLL models suffer from a problem of error accumulation, because complicated instances are difficult to classify and easy to be mistakenly disambiguated, which could further mislead the model with false positive labels and cause performance degradation. The co-training strategy [20,21], which trains two networks simultaneously and makes them interact with each other, is a feasible solution to mitigate error accumulation. While the co-training strategy has been extensively explored in noisy label learning (NLL) [22–24], its usage in PLL remains understudied. Recently, Yao et al. [25] proposed a novel approach called NCPD for PLL based on co-training. NCPD transforms partially labeled datasets into highly noisy datasets via data duplication and adopts a typical NLL method called co-teaching [24]. However, NCPD not only causes extremely high time and space complexity but also obtains limited performance.

Moreover, the majority of existing co-training models, including NCPD for PLL, are symmetric, i.e., their two branches of networks have the same structure and are trained with the same input data and loss functions. They assume that different parameter initializations enable the two structurally identical networks trained with the same task to obtain distinct capabilities so that they are able to provide mutual corrections for each other. Nevertheless, being trained in a symmetric paradigm makes the two networks fall into the same limitations easily, e.g., both of them are hard to recognize complicated instances correctly. Consequently, they are easier to reach a consensus and cannot correct errors for each other effectively.

Therefore, we argue that training the two networks in asymmetric paradigm by constructing different structures for each branch of the network carefully or training them with distinct input data or loss functions from different views, can explicitly enable them to capture disparate information and enhance the possibility to get complementary capabilities, which benefits error correction. Intuitively, in PLL, the partially labeled dataset can be transformed into an exactly labeled dataset by annotating each instance with a pseudo label corresponding to the maximum confidence. Then, the two networks can be trained via a partial-label learning task and a supervised learning task, respectively. However, training with the generated exactly labeled dataset is challenging since mistakenly disambiguated instances could be annotated with noisy class labels, which could be harmful to model learning. Fortunately, according to Wu et al. [26], under mild conditions, when the number of classes $c \leqslant 8$, the noise rate of noisy pairwise similarity labels, i.e., labels indicating whether or not two instances belong to the same class, is lower than that of the noisy class labels. Thus, converting the noisy class labels into noisy similarity labels can reduce the influence of noisy class labels.

In the light of the above motivations, we propose an asymmetric dual-task co-training PLL model AsyCo. AsyCo comprises two networks that share identical structures but are trained by distinct tasks. The first network is designed as a disambiguation network that focuses on resolving label ambiguity and is trained using a self-training method, for the PLL task. According to its learned confidence, we generate pseudo class labels for instances by annotating each training instance with the most confident label, and further transform the noisy pseudo class labels into noisy pairwise similarity labels, of which the noise rates are much lower. Then, the second network, referred to as an auxiliary network, is then trained using the generated noisy similarity labels in a supervised learning paradigm. With the clarified

information provided by the disambiguation network, the auxiliary network is trained with higher-quality data, improving the probability of classifying complicated instances correctly. Besides, the auxiliary network utilizes different data and loss functions from the disambiguation network for training and makes it easier to obtain the complementary classification capabilities of the disambiguation network. Therefore, we leverage the prediction of the auxiliary network to conduct error correction for the disambiguation network and boost classification accuracy.

Overall, our contributions can be summarized as follows:

(1) We explore asymmetric co-training for PLL and propose a novel deep dual-task PLL model AsyCo that trains two structurally identical networks with distinct tasks collaboratively.

(2) As an integral part of AsyCo, an effective supervised learning auxiliary network is proposed, which utilizes the pseudo labels identified by the disambiguation network for training and mitigates the error accumulation problem via distillation and refinement in turn.

(3) Extensive experimental results on benchmark datasets demonstrate the superior performance of AsyCo on both uniform and instance-dependent partially labeled data.

The rest of the paper is organized as follows. First, some necessary preliminary knowledge of PLL is illustrated in Section 2. Next, we present the proposed AsyCo model in Section 3 and report experiments in Section 4, respectively. Then, related work is briefly reviewed in Section 5. Lastly, we conclude this paper in Section 6.

2 Preliminaries

2.1 Problem settings

Let $\mathcal{X} \subset \mathbb{R}^d$ denote the d-dimensional feature space and $\mathcal{Y} \subset \{1, 2, \ldots, m\}$ denote the label space. In partially labeled datasets, each training instance $\boldsymbol{x}_i \in \mathcal{X}$ is labeled with a candidate label set $Y_i \subset \mathcal{Y}$ that contains the ground-truth label y_i. Our goal is to learn a multi-class classifier $f(\cdot)$ on partially label dataset $\mathcal{D} = \{(\boldsymbol{x}_i, Y_i) | 1 \leqslant i \leqslant n\}$. Let $p_{ik} = f_k(\boldsymbol{x}_i)$ denote the predicted probability of classifier $f(\cdot)$ on label k given instance $\boldsymbol{x}_i$.

Note that for non-structural instances, such as images and text, their d-dimensional features are typically extracted using deep neural network (DNN)-based encoders from their raw feature. For instance, the feature encoder for images is commonly constructed based on convolutional neural networks like LeNet, ResNet, or Wide-ResNet.

2.2 Classifier-consistent PLL loss

The classifier-consistent (CC) PLL loss [18] assumes each candidate label set is uniformly sampled and is presented as follows:

$$L_{\mathrm{cc}}(\boldsymbol{x}_i) = -\log\left(\sum_{k \in Y_i} p_{ik}\right). \tag{1}$$

Minimizing the CC loss is equivalent to maximizing the sum of the classification probabilities of all the candidate labels while minimizing the sum of the classification probabilities of non-candidate labels.

2.3 Risk-consistent PLL loss

The above classifier-consistent PLL loss [18] assigns the same weight to each candidate label and makes the ground-truth label easily overwhelmed by other false positive labels. Thus, a risk-consistent (RC) loss based on the importance-weighting strategy was proposed [18]. By leveraging the widely-used categorical cross entropy loss as the basic classification loss, the risk-consistent PLL loss is formulated as follows:

$$L_{\mathrm{rc}}(\boldsymbol{x}_i) = -\sum_{k=1}^{m} c_{ik} \log p_{ik}, \quad c_{ik} = \frac{p(y_i = k \mid \boldsymbol{x}_i)}{\sum_{j \in Y_i} p(y_i = j \mid \boldsymbol{x}_i)}, \tag{2}$$

where $p(y_i = k \mid \boldsymbol{x}_i)$ represents the probability that instance $\boldsymbol{x}_i$ belongs to category k. Actually, c_{ik} implies how confident the probability of falling into category k is; thus, a confidence vector can be formed

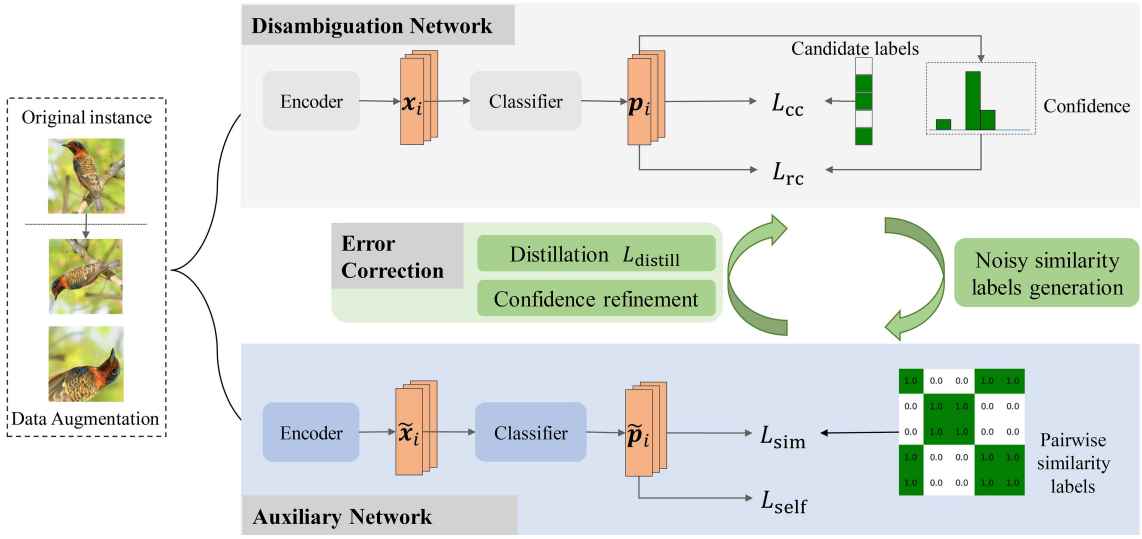

Figure 2 (Color online) Architecture of AsyCo (in the training phase). AsyCo comprises two networks with identical structures, namely the disambiguation network and the auxiliary network. The former is responsible for resolving label ambiguities and learning label confidence, while the latter is trained by pairwise similarity labels constructed according to the learned confidence. Besides, the auxiliary network facilitates error correction for the disambiguation network through information distillation and confidence refinement, thereby mitigating the error accumulation problem.

as $[c_{i0}, c_{i1}, \ldots, c_{im}]$. Since $p\left(y_i = k \mid \boldsymbol{x}_i\right)$ is not accessible from the given data, it is approximated by the classification probability shown as

$$p\left(y_i = k \mid \boldsymbol{x}_i\right) = \begin{cases} f_k\left(\boldsymbol{x}_i\right), & \text{if } k \in Y_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

By calculating $p\left(y_i = k \mid \boldsymbol{x}_i\right)$ as above, the RC PLL loss trains models in a self-training manner.

3 Proposed model

As shown in Figure 2, our model is composed of a disambiguation network, an auxiliary network, and an error correction module. The two networks have identical structures but are trained with different tasks and loss functions, leading to discrepancies in their parameters and capabilities. Specifically, the disambiguation network undergoes training using PLL losses and obtains the confidence of each label in the candidate label set, while the auxiliary network leverages low-noise pairwise similarity labels generated according to the learned label confidence and is trained via supervised learning losses. Finally, the auxiliary network addresses the issue of error accumulation in the disambiguation network through information distillation and confidence refinement. To facilitate understanding, the notations in AsyCo are summarized in Table 1.

3.1 Disambiguation network

Given an instance $\boldsymbol{x}_i$, a classifier is proposed to compute classification logits using a multi-layer perceptron (MLP). The classifier further calculates the classification probability $\boldsymbol{p}_i \in \mathbb{R}^m$ by applying the softmax function. Specifically, the kth element of $\boldsymbol{p}_i$, which represents the probability that the instance $\boldsymbol{x}_i$ is classified into the kth category, is determined as follows:

$$p_{ik} = p\left(y_i = k \mid \boldsymbol{x}_i\right) = f\left(\boldsymbol{x}_i\right) = \frac{\exp\left(\mathrm{MLP}_k\left(\boldsymbol{x}_i\right)/\tau\right)}{\sum_j \exp\left(\mathrm{MLP}_j\left(\boldsymbol{x}_i\right)/\tau\right)}, \tag{4}$$

where $\mathrm{MLP}_k(\boldsymbol{x}_i)$ denotes the classification logit for classifying instance $\boldsymbol{x}_i$ into label k. Additionally, τ serves as a temperature parameter, wherein a higher value of τ results in smoother classification probabilities.

Table 1 Notations for AsyCo.

Notation	Description
$\mathcal{X} \subset \mathbb{R}^d$	d-dimensional feature space
$\mathcal{Y} \subset \{1, 2, \ldots, m\}$	Label space
$\boldsymbol{x}_i \in \mathcal{X}, \tilde{\boldsymbol{x}}_i \in \mathcal{X}$	Features of the ith instance in the disambiguation network and the auxiliary network, respectively
$Y_i \subset \mathcal{Y}$	Candidate label set of the instance $\boldsymbol{x}_i$
$\mathcal{D} = \{(\boldsymbol{x}_i, Y_i) \vert 1 \leqslant i \leqslant n\}$	Partially labeled dataset
$\mathcal{A}(\boldsymbol{x}_i) = \{\boldsymbol{x}_i, \boldsymbol{x}_i', \boldsymbol{x}_i''\}$	Instance set containing the original instance $\boldsymbol{x}_i$ and its augmentations
$f(\cdot)$	Classifier that takes instance feature as input
$\boldsymbol{p}_i, \boldsymbol{p}_i', \boldsymbol{p}_i'', \boldsymbol{c}_i, \boldsymbol{c}_i', \boldsymbol{c}_i''$	Predicted probabilities and label confidence vectors of the instances in $\mathcal{A}(\boldsymbol{x}_i) = \{\boldsymbol{x}_i, \boldsymbol{x}_i', \boldsymbol{x}_i''\}$
$\boldsymbol{w}_i = [w_{i0}, w_{i1}, \ldots, w_{im}]$	The comprehensive label confidence vector of the instances in $\boldsymbol{x}_i$ that integrates $\boldsymbol{c}_i, \boldsymbol{c}_i'$, and $\boldsymbol{c}_i''$
$\gamma(t)$	Non-decreasing factor balancing CC loss and RC loss
k'	Pseudo label of instance $\boldsymbol{x}_i$ generated according to label confidence vector $\boldsymbol{w}_i$
$\tilde{f}(\cdot)$	Classifier in the auxiliary network
$\mathcal{A}(\tilde{\boldsymbol{x}}_i) = \{\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_i', \tilde{\boldsymbol{x}}_i''\}$	Instance set containing the original instance $\tilde{\boldsymbol{x}}_i$ and its augmentations in the auxiliary network
$\tilde{\boldsymbol{p}}_i, \tilde{\boldsymbol{p}}_i', \tilde{\boldsymbol{p}}_i''$	Predicted probability of instances in $\mathcal{A}(\tilde{\boldsymbol{x}}_i)$ calculated by the auxiliary network
$\tilde{\boldsymbol{w}}_i$	The comprehensive label confidence vector of the instance $\tilde{\boldsymbol{x}}_i$ that integrates $\tilde{\boldsymbol{c}}_i, \tilde{\boldsymbol{c}}_i'$, and $\tilde{\boldsymbol{c}}_i''$
$s_{ij} \in \{0, 1\}$	Generated similarity label between the instance $\tilde{\boldsymbol{x}}_i$ and the instance $\tilde{\boldsymbol{x}}_j$
$\mathcal{D}_{sim} = \{\langle \tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j \rangle, s_{ij}\}$	Generated similarity dataset, where $\langle \tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j \rangle \in \mathcal{X} \times \mathcal{X}$
$\mu(t)$	Non-decreasing refinement factor of the label confidence calculated by the auxiliary network
$\hat{\boldsymbol{w}}_i$	The refined label confidence vector of instance $\boldsymbol{x}_i$
τ, T, λ	Hyper-parameters

Data augmentation, which generates richer and harder instances by making slight modifications to the original instances, benefits classification performance. Following DPLL [16], we apply two augmentation methods, i.e., Autoaugment [27] and Cutout [28], to generate two augmentated views for each instance, which are defined as $\boldsymbol{x}_i' = \mathrm{Aug}_1(\boldsymbol{x}_i)$ and $\boldsymbol{x}_i'' = \mathrm{Aug}_2(\boldsymbol{x}_i)$. The original instance and the augmented instances of $\boldsymbol{x}_i$ form a instance set $\mathcal{A}(\boldsymbol{x}_i)$, i.e., $\mathcal{A}(\boldsymbol{x}_i) = \{\boldsymbol{x}_i, \boldsymbol{x}_i', \boldsymbol{x}_i''\}$. Besides, $\boldsymbol{x}_i', \boldsymbol{x}_i''$ and $\boldsymbol{p}_i', \boldsymbol{p}_i''$ denote the features and classification probabilities of the augmented instances, respectively. According to (6), the label confidence vectors of the origin instance $\boldsymbol{x}_i$ and its augmentations $\boldsymbol{c}_i', \boldsymbol{c}_i''$ can be calculated successively, which are defined as $\boldsymbol{c}_i, \boldsymbol{c}_i', \boldsymbol{c}_i''$, respectively.

We extend the original CC and RC losses to accommodate the augmented instances, and train the disambiguation network with the enhanced losses. Specifically, with data augmentation, the CC loss of instance $\boldsymbol{x}_i$ can be computed as follows:

$$L_{\mathrm{cc}}(\boldsymbol{x}_i) = -\frac{1}{|\mathcal{A}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{x}_i^* \in \mathcal{A}(\boldsymbol{x}_i)} \log\left(\sum_{k \in Y_i} p_{ik}^*\right), \tag{5}$$

where $|\mathcal{A}(\boldsymbol{x}_i)|$ represents the cardinality of $\mathcal{A}(\boldsymbol{x}_i)$.

As for RC loss, we define the confidence vector of instance $\boldsymbol{x}_i^* \in \mathcal{A}(\boldsymbol{x}_i)$ as $\boldsymbol{c}_i^*$ and integrate the confidence vectors of all the instances in $\mathcal{A}(\boldsymbol{x}_i)$ to calculate a comprehensive label confidence vector $\boldsymbol{w}_i$, which is more accurate and robust. In detail, the confidence corresponding to class k for instance $\boldsymbol{x}_i$ can be calculated as follows:

$$w_{ik} = \frac{\left(\prod_{\boldsymbol{x}_i^* \in \mathcal{A}(\boldsymbol{x}_i)} c_{ik}^*\right)^{\frac{1}{|\mathcal{A}(\boldsymbol{x}_i)|}}}{\sum_{j \in Y_i}\left(\prod_{\boldsymbol{x}_i^* \in \mathcal{A}(\boldsymbol{x}_i)} c_{ij}^*\right)^{\frac{1}{|\mathcal{A}(\boldsymbol{x}_i)|}}}. \tag{6}$$

Then, the data augmentation enhanced RC loss of instance $\boldsymbol{x}_i$, which is a self-training loss, is as follows:

$$L_{\mathrm{rc}}(\boldsymbol{x}_i) = -\frac{1}{|\mathcal{A}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{x}_i^* \in \mathcal{A}(\boldsymbol{x}_i)} \sum_{k \in Y_i} w_{ik} \log p_{ik}^*. \tag{7}$$

Minimizing the data augmentation enhanced RC loss simultaneously drives the classification probabilities of both the original and augmented instances closer to the confidence vector $[\boldsymbol{w}_i = w_{i1}, w_{i2}, \ldots, w_{im}]$. This process encourages the network's output to be invariant to minor changes made in the feature space, thereby implicitly extracting self-supervised information.

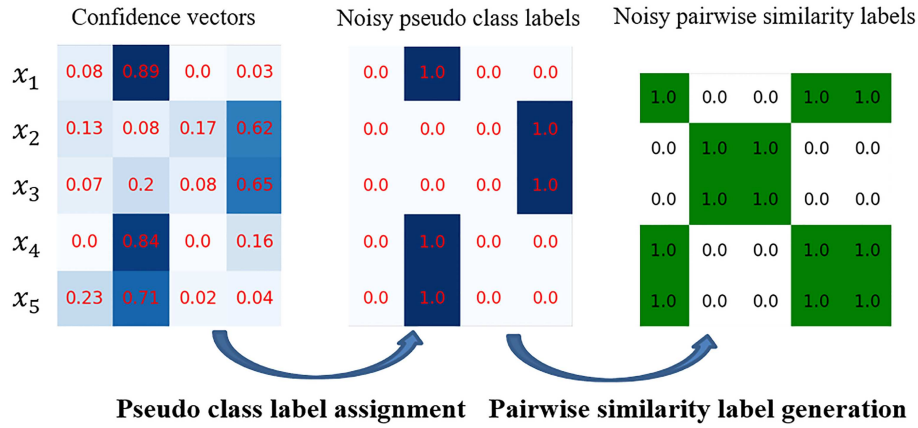

Figure 3 (Color online) Example of generating noisy pairwise similarity labels according to confidence vectors.

Finally, the disambiguation network is trained by the following constructed loss:

$$L_{\text{disam}}\left(\boldsymbol{x}_i\right) = L_{\text{cc}}\left(\boldsymbol{x}_i\right) + \gamma(t) L_{\text{rc}}\left(\boldsymbol{x}_i\right). \tag{8}$$

We introduce a coefficient λ for the RC loss to address the potential instability of learned confidence vectors in the early stages of training. This coefficient serves as a non-decreasing balancing function that gradually increases over the training epoch number t, i.e.,

$$\gamma(t) = \min\left\{\frac{t}{T}\lambda, \lambda\right\}, \tag{9}$$

where λ and T are hyper-parameters.

3.2 Auxiliary network

The auxiliary network has the same structure as the disambiguation network, but their parameters are trained differently. To symbolically differentiate the components and variables of these two networks, the symbols corresponding to the auxiliary network are augmented with tilde symbols. For example, in the auxiliary network, the classifier, the feature, the predicted classification probabilities, and the final label confidence are defined as $\tilde{f}(\cdot)$, $\tilde{\boldsymbol{x}}_i$, $\tilde{\boldsymbol{p}}_i$ and $\tilde{\boldsymbol{w}}_i$, respectively.

As the disambiguation network undergoes training, the precision of the confidence vector continually improves. For a certain portion of instances, their ground-truth labels are identified by the maximum confidence value. This motivates us to assign an exact pseudo class label to each instance according to its label confidence, thereby sufficiently utilizing this clarified information to enhance model training. As shown in Figure 3, assuming instance $\boldsymbol{x}_i$ gets the largest confidence on the k'th label, i.e., $k' = \operatorname{argmax}_k \{w_{ik} \mid y_{ik} \in Y_i\}$, we generate a one-hot label vector for it, of which the k'th element is 1, other elements are 0. Consequently, we obtain a new dataset where each instance is annotated with an exact class label. Because some instances may be mistakenly disambiguated and annotated, there are some noisy data in the generated dataset.

Referring to the work of Wu et al. [26], the noise rates of the pairwise similarity labels are lower than that of the intermediate noisy class labels in most practical scenarios. Therefore, we transform the pseudo class labels into pairwise similarity labels, and utilize the resulting similarity dataset to train the auxiliary network. Specifically, for each pair of instances, if they share the same pseudo class label, we assign a similarity label of 1 to them; otherwise, we assign a similarity label of 0. The generated similarity dataset is defined as $\mathcal{D}_{\text{sim}} = \{\langle \tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j \rangle, s_{ij}\}$, where $\langle \tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j \rangle \in \mathcal{X} \times \mathcal{X}, s_{ij} \in \{0, 1\}$.

For pairs of instances with a similarity label of 1, it is expected that their predicted classification probabilities exhibit high similarity. To capture this inter-instance relationship, we propose a binary cross-entropy loss function as

$$L_{\text{sim}}\left(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j, s_{ij}\right) = \frac{1}{|\mathcal{A}(\tilde{\boldsymbol{x}}_i)|} \sum_{\tilde{\boldsymbol{x}}_i^* \in \mathcal{A}(\tilde{\boldsymbol{x}}_i)} -s_{ij} \log\left(\tilde{\boldsymbol{p}}_i^{\mathrm{T}} \tilde{\boldsymbol{p}}_j\right) - \left(1 - s_{ij}\right) \log\left(1 - \tilde{\boldsymbol{p}}_i^{\mathrm{T}} \tilde{\boldsymbol{p}}_j\right). \tag{10}$$

It should be noted that both $\tilde{\boldsymbol{p}}_i$ and $\tilde{\boldsymbol{p}}_j$ are classification probabilities normalized by the softmax function, thus, $0 \leqslant \tilde{\boldsymbol{p}}_i^{\mathrm{T}} \tilde{\boldsymbol{p}}_j \leqslant 1$.

Furthermore, in order to keep the consistency between the original instances and their augmentations, we construct the following cross entropy-based loss to learn self-supervised information:

$$L_{\mathrm{ssl}}\left(\tilde{\boldsymbol{x}}_i\right) = -\frac{1}{2} \sum_{\tilde{\boldsymbol{x}}_i^* \in \{\tilde{\boldsymbol{x}}_i', \tilde{\boldsymbol{x}}_i''\}} \sum_{y_{ik} \in Y_i} \left(\text{stop-grad}(\tilde{p}_{ik}) \log \tilde{p}_{ik}^*\right), \tag{11}$$

where stop-grad$(\tilde{p}_{ik})$ denotes that we stop the gradients of $\tilde{p}_{ik}$ in L_{ssl} during back-propagation.

Finally, the overall loss to train the auxiliary network is as follows:

$$L_{\mathrm{aux}}(\tilde{\boldsymbol{x}}_i) = L_{\mathrm{ssl}}(\tilde{\boldsymbol{x}}_i) + \gamma(t) \left(\frac{1}{n} \sum_{j=0}^{n} L_{\mathrm{sim}}(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j, s_{ij}) \right), \tag{12}$$

where $\gamma(t)$ is defined in (9). As the training process progresses, the noise rate of the generated similarity labels gradually decreases; thus the weight of L_{sim} improves gradually.

3.3 Error correction

We leverage the auxiliary network to alleviate the issue of error accumulation in disambiguation networks. To achieve this, two error correction strategies, i.e., information distillation and confidence refinement, are established, which affect the training of the disambiguation network in direct and indirect ways, respectively.

For information distillation, regarding the predicted probability of the auxiliary network as ground-truth distribution, we introduce the following KL divergence-based loss to ensure that the predicted probability of the disambiguation network closely aligns with that of the auxiliary network:

$$L_{\mathrm{distill}}\left(\boldsymbol{x}_i\right) = \mathrm{KL}(\text{stop-grad}(\tilde{\boldsymbol{p}}_i) \| \boldsymbol{p}_i). \tag{13}$$

By ignoring the gradients from $\tilde{\boldsymbol{p}}_i$ in L_{distill}, we avoid the impact of the disambiguation network's prediction on the auxiliary network.

Additionally, similarly to the confidence calculation in the disambiguation network as described in (6), the comprehensive confidence vector $\tilde{\boldsymbol{w}}_i$ for instance $\tilde{\boldsymbol{x}}_i$ is obtained using the prediction results of the auxiliary network. Then, $\tilde{\boldsymbol{w}}_i$ is utilized to refine the label confidence of the disambiguation network. During the t-th epoch, the refined confidence is computed as follows:

$$\hat{\boldsymbol{w}}_i(t) = (1 - \mu(t))\boldsymbol{w}_i(t) + \mu(t)\tilde{\boldsymbol{w}}_i(t), \tag{14}$$

where $\mu(t)$ is a non-decreasing function of training epoch t. Here, we set $\mu(t) = \min(\rho \times \max(t - t_0, 0), \mu_{\max})$. t_0 and $\mu_{\max}$ are hyper-parameters, where $\mu(t) = 0$ before the t_0-th training epoch and $0 \leqslant \mu_{\max} \leqslant 1$ is the upper bound of $\mu(t)$. The increase speed of μ depends on ρ. During co-training, the original confidence $\boldsymbol{w}_i$ in (7) is replaced by the refined confidence $\hat{\boldsymbol{w}}_i$.

Effective error correction enhances label disambiguation and enables the generation of purer pseudo class labels, further boosting the accuracy of the auxiliary network. The interplay between these two networks forms a virtuous cycle.

3.4 Training and inference

The overall training loss is as follows:

$$L_{\mathrm{total}}\left(\boldsymbol{x}_i\right) = L_{\mathrm{disam}}\left(\boldsymbol{x}_i\right) + L_{\mathrm{aux}}\left(\tilde{\boldsymbol{x}}_i\right) + \gamma(t) L_{\mathrm{distill}}\left(\boldsymbol{x}_i\right), \tag{15}$$

where the distillation loss based on confidence is also set to the weight $\gamma(t)$.

The detailed training procedure is as follows. Initially, the disambiguation network is warmed up for several epochs to ensure the accurate identification of ground-truth labels for certain training instances by the confidence vectors. Subsequently, the pre-trained model parameters are employed to initialize the parameters of the auxiliary network. Moreover, to enhance the efficiency of model training, noisy similarity labels are generated using instances within the same mini batch.

In the inference phase, the predicted probability ensemble of the two learned classifiers naturally enhances performance but incurs additional prediction overhead. Consequently, we opt to utilize only one network for inference. The choice between the disambiguation network and the auxiliary network is insignificant as they ultimately converge to a similar level of performance. In the inference phase, we utilize the disambiguation network for prediction.

3.5 Complexity analysis

In the training phase, compared to self-training PLL models without co-training, such as DPLL [16], AsyCo requires approximately twice the amount of space due to the co-training of two networks. As for the time complexity, the computational overhead of AsyCo mainly comes from backbone networks, classifiers, and loss functions. Because backbone networks and classifiers are modular and replaceable in AsyCo, without loss of generality, let $O(B)$ and $O(C)$ denote their computation complexity on an instance, respectively. Let the total number of training instances be N, in the disambiguation network and auxiliary network, the classification probability of each instance is calculated via the backbone network and classifier once, so the time cost is $O(2NB + 2NC)$. The time complexity of RC and CC loss in disambiguation network is $O(3N + 3N)$. Since N^2 similarity labels are constructed in the auxiliary network, the time complexity of supervised loss is $O(N^2)$. Besides, the time complexity of self-supervised loss in the auxiliary network is $O(3N)$. Totally, the complexity of the AsyCo is $O(2NB + 2CN + 6N + N^2 + 3N)$. Due to $O(B) \gg O(C)$ and $(B) \gg O(N)$ in most situations, the final complexity of AsyCo depends on $O(2NB)$, which is almost twice as much as non-co-training models. In practice, the parallel computing capabilities of GPUs effectively reduce the impact of co-training on training time by leveraging sufficient data parallelism.

In the inference phase, AsyCo exhibits comparable time and space complexity to DPLL, which is attributed to the utilization of one single network for prediction.

4 Experiments

4.1 Experimental setup

4.1.1 *Datasets*

We conduct experiments on five datasets, which include three widely used benchmark datasets, including SVHN [29], CIFAR-10 and CIFAR-100 [30], a text classification dataset CNAE-9[1)], and a real-world partial label dataset BirdSong, which is for bird song classification task. Following [16], we construct partially labeled datasets for the four benchmark datasets via two generating processes, i.e., a uniform process and an instance-dependent process. In the uniform generating process, incorrect labels have the same probability q to be a candidate label, where q varies in $\{0.1, 0.3, 0.5, 0.7\}$ on SVHN, CIFAR-10, and CNAE-9, and $\{0.01, 0.05, 0.1, 0.2\}$ on CIFAR-100. We conduct the instance-dependent candidate generating process on the image datasets. We pretrain 18-layer ResNet (ResNet-18) firstly, and the probability of incorrect label j turning into a false positive label is calculated as

$$\frac{g'_j\left(\boldsymbol{x}_i\right)}{\max_{k \in Y_i} g'_k\left(\boldsymbol{x}_i\right)},$$

where $g'_j(\boldsymbol{x}_i)$ is the classification probability into label j calculated by the pretrained ResNet-18 given input $\boldsymbol{x}_i$.

4.1.2 *Compared methods*

To evaluate the performance of AsyCo, we choose the following deep PLL methods as competitors: (1) CC [18], a classifier-consistent method based on the assumption that candidate label sets are generated uniformly. (2) RC [18], a risk-consistent method based on the importance of re-weighting strategy. (3) PRODEN [19], a progressive identification method accomplishing classifier learning and label identification simultaneously. (4) PiCO [15], a PLL model utilizing a contrastive learning module along with a novel class prototype-based label disambiguation algorithm. (5) DPLL [16], a model leveraging

1) https://archive.ics.uci.edu/dataset/233/cnae+9.

consistency regularization for deep PLL. (6) NCPD [25], a co-training-based PLL model employing a progressive disambiguation strategy combined with a network cooperation mechanism for PLL. (7) Fully supervised learning, a model trained with the exact ground-truth labels and cross-entropy loss enhanced by data augmentation.

4.1.3 *Implementation details*

AsyCo is implemented using PyTorch, with an 18-layer ResNet utilized as the backbone network, i.e., serving as the feature encoder on image datasets. Since both CNAE-9 and BirdSong are not very large, we construct linear layers as their feature encoders. For these two non-image datasets, we augment the instances by adding random tokens and Gaussian noise, respectively. The optimization of the model is carried out using the SGD optimizer, with a momentum value set to 0.9 and a weight decay set to 1E−4. The initial learning rate is set to 0.1, and at the 100th and 150th epochs, the learning rate is divided by 10. The total number of training epochs is set to 200, with warm-up epochs accounting for 50 when $q = 0.2$ on CIFAR-100 and 20 in other scenarios. The batch size is set to 64. The value of τ is searched within the range of $[1, 5, 10, 20, 30]$, and ultimately selected as $\tau = 20$. When calculating λ, the values of T and $\lambda_{\max} = 1$ are set to 100 and 1, respectively. Furthermore, for the hyper-parameters related to confidence refinement, ρ is set to 0.02, t_0 is determined by adding the number of warm-up epochs to 50, and $\mu_{\max}$ is set to 0.9. The source code can be found at https://github.com/libeibeics/AsyCo.

For a fair comparison, we employ the same backbone network, learning rate, optimizer, and batch size for all methods, including fully supervised learning. Additionally, for those methods that did not originally employ data augmentation techniques (e.g., RC, CC, and PRODEN), we enhance the models by incorporating the same data augmentation methods used in AsyCo. Particularly, to ensure PiCO can achieve its optimal performance, we retain data augmentation as described in its original paper. Moreover, we adopt the values of hyper-parameters from their original papers to guarantee that the compared methods are able to achieve their own best performance. As for training epochs, PiCO undergoes 800 epochs, whereas the remaining models are trained for 200 epochs. To obtain reliable and robust results, we conduct three repeated experiments with different random seeds and report the mean and standard deviation of these results. All models are trained on GeForce RTX 3090s equipped with 24 GB of memory except for NCPD when setting $q = 0.7$ on SVHN. Because there occurs memory limitation error (MLE) in NCPD on the SVHN data set when $q = 0.7$, we run the experiment on V100 with 32 GB of memory to get the results.

4.2 Performance comparison

The performance comparison results are shown in Table 2. From the results, we have the following observation and analysis.

First, DPLL, which is the state-of-the-art deep PLL model, performs differently across datasets. Specifically, in the experiments conducted on CIFAR-10 and CIFAR-100, it is observed that the DPLL model significantly outperforms other compared models, which highlights the effectiveness of manifold consistency regularization in enhancing classification accuracy. But on SVHN, DPLL is slightly inferior to other comparison methods such as PRODEN, RC, and CC. The discrepancy in performance on different datasets of DPLL may result from the nature of the task at hand. SVHN involves a relatively simpler classification task of recognizing 0–9 digits, whereas the image classification task on CIFAR datasets is more complex. Due to the inherent simplicity of the classification task on SVHN, the task can be well-solved using simple models, e.g., RC and CC.

Second, the NCPD model, which is based on symmetric co-training, exhibits strong performance when q is small while significantly deteriorates as q increases. For instance, it outperforms all other methods when $q = 0.1$ on CIFAR-10 and even slightly surpasses AsyCo at the same q value on SVHN. However, when $q = 0.7$ on both CIFAR-10 and SVHN, NCPD performs the worst compared to its competitors. This decline in performance can be attributed to the fact that NCPD converts the partially labeled dataset into a noisy dataset through multi-birth duplication. Consequently, the noise rate of the generated dataset is dependent on the average number of candidate labels, meaning that a higher q leads to a greater noise rate. As a result, larger values of q result in more severe degradation of performance. Additionally, NCPD suffers from another limitation, that is, larger values of q lead to an increased number of instances in the generated noisy dataset, resulting in higher time and space overhead for model training.

Table 2 Accuracy (mean±std) (%) comparison with uniform partial labels on different ambiguity levels on image datasets. The best performance in each column is highlighted in bold. The second best performance in each column is underlined. The improvement is calculated based on the best competitor and is highlighted in bold.

Dataset	Model	$q = 0.1$	$q = 0.3$	$q = 0.5$	$q = 0.7$
SVHN	Fully Supervised		97.435 ± 0.016		
	CC	97.348 ± 0.100	97.139 ± 0.048	96.978 ± 0.020	96.377 ± 0.020
	RC	97.292 ± 0.085	97.243 ± 0.128	97.050 ± 0.049	95.898 ± 0.108
	PRODEN	97.081 ± 0.077	96.445 ± 0.290	96.183 ± 0.325	94.573 ± 0.492
	PiCO	95.680 ± 0.080	95.585 ± 0.015	95.630 ± 0.020	95.150 ± 0.024
	DPLL	97.261 ± 0.029	97.062 ± 0.013	96.797 ± 0.033	94.972 ± 0.106
	NCPD	**97.469 ± 0.011**	97.431 ± 0.045	97.325 ± 0.041	18.865 ± 2.157
	AsyCo	97.374 ± 0.015	**97.471 ± 0.086**	**97.553 ± 0.023**	**97.539 ± 0.013**
	Improv.	–	↑ **0.040**	↑ **0.228**	↑ **1.162**
Dataset	Model	$q = 0.1$	$q = 0.3$	$q = 0.5$	$q = 0.7$
CIFAR-10	Fully Supervised		96.458 ± 0.062		
	CC	94.129 ± 0.181	93.226 ± 0.261	92.102 ± 0.155	88.846 ± 0.031
	RC	94.950 ± 0.100	94.610 ± 0.054	94.139 ± 0.059	92.423 ± 0.051
	PRODEN	94.443 ± 0.213	93.845 ± 0.326	93.466 ± 0.243	91.259 ± 0.780
	PiCO	94.357 ± 0.109	94.183 ± 0.179	93.697 ± 0.238	92.157 ± 0.209
	DPLL	95.905 ± 0.052	95.654 ± 0.208	95.365 ± 0.140	93.856 ± 0.366
	NCPD	96.284 ± 0.050	95.280 ± 0.110	95.280 ± 0.110	76.583 ± 0.522
	AsyCo	**96.645 ± 0.004**	**96.279 ± 0.030**	**96.003 ± 0.013**	**95.550 ± 0.007**
	Improv.	↑ **0.361**	↑ **0.625**	↑ **0.638**	↑ **1.694**
Dataset	Model	$q = 0.01$	$q = 0.05$	$q = 0.1$	$q = 0.2$
CIFAR-100	Fully Supervised		80.385 ± 0.013		
	CC	75.560 ± 0.537	75.138 ± 0.154	73.224 ± 1.017	69.035 ± 0.339
	RC	76.252 ± 0.168	75.689 ± 0.129	74.737 ± 0.282	72.708 ± 0.358
	PRODEN	76.147 ± 0.291	75.682 ± 0.097	74.604 ± 0.285	72.512 ± 0.212
	PiCO	73.145 ± 0.035	72.585 ± 0.145	59.365 ± 0.445	25.545 ± 0.715
	DPLL	79.300 ± 0.262	78.855 ± 0.165	78.064 ± 0.050	76.316 ± 0.232
	NCPD	78.190 ± 0.080	76.990 ± 0.041	71.923 ± 0.042	42.701 ± 0.832
	AsyCo	**80.775 ± 0.010**	**80.433 ± 0.087**	**79.668 ± 0.058**	**78.061 ± 0.001**
	Improv.	↑ **1.475**	↑ **1.578**	↑ **1.604**	↑ **1.745**

Third, in general, the proposed AsyCo demonstrates significant superiority over all its competitors on three datasets with various q values. For instance, on CIFAR-10, when p takes on the values of 0.1, 0.3, 0.5, and 0.7, the performance of AsyCo surpasses that of the best competitor by 0.361%, 0.625%, 0.638%, and 1.694%, respectively. This clearly indicates the remarkable performance of AsyCo. Moreover, the effectiveness of asymmetric dual-task co-training is convincingly demonstrated by the superiority of AsyCo over NCPD. Additionally, it is noteworthy that the accuracy of AsyCo remains highly competitive compared to supervised learning. When q is small, for example, on CIFAR-10 with $q = 0.1$, as well as on CIFAR-100 with $q = 0.01$ and $q = 0.05$. AsyCo achieves even better performance than supervised learning in certain scenarios. We futher remove data augmentation from both AsyCo and supervised learning and compare their performance. As shown in Table 3, without data augmentation, AsyCo still outperforms fully supervised learning significantly, which suggests that AsyCo effectively incorporates different capabilities of two networks through co-training, which can mine high-quality supervised signals from partially labeled data, for example, the relationship between instance pairs.

Furthermore, AsyCo exhibits strong robustness with respect to data quality degradation. Specifically, as the value of q increases, the performance of most compared methods noticeably deteriorates. For instance, on CIFAR-10, the accuracy of the competitors decreases by a range of 2% to 20% as q increases from 0.1 to 0.7. In contrast, the accuracy of AsyCo only fluctuates within the narrower range of 96.645% to 95.550%. Consequently, the disparity in accuracy between the top-performing competitor and AsyCo becomes more pronounced with increasing values of q. Specifically, on SVHN, the performance improvements achieved by AsyCo are 0.228% and 1.162% for $q = 0.5$ and $q = 0.7$, respectively. Similarly, on CIFAR-100, the accuracy improvements for AsyCo at q values of 0.01 and 0.2 are 1.475% and 1.745%,

Table 3 Accuracy (%) comparison to supervised learning without data augmentation. The best performance in each column is highlighted in bold.

Dataset	SVHN	CIFAR-10	CIFAR-100
Fully Supervised w/o D.A.	95.647 ± 0.147	**95.210 ± 0.245**	75.621 ± 0.478
AsyCo w/o D.A.	**96.338 ± 0.030** ($q = 0.1$)	94.981 ± 0.050 ($q = 0.1$)	**76.415 ± 0.175** ($q = 0.01$)

Table 4 Accuracy (mean±std) (%) comparison with uniform partial labels on different ambiguity levels on CNAE-9. The best performance in each column is highlighted in bold. The second best performance in each column is underlined. The improvement is calculated based on the best competitor and is highlighted in bold.

Dataset	Model	$q = 0.1$	$q = 0.3$	$q = 0.5$	$q = 0.7$
CNAE-9	Fully Supervised		95.095 ± 1.044		
	CC	93.673 ± 0.218	92.438 ± 0.437	91.204 ± 0.378	82.870 ± 0.655
	RC	92.901 ± 0.218	91.870 ± 0.838	89.043 ± 0.787	82.870 ± 0.655
	PRODEN	93.673 ± 0.873	93.827 ± 0.436	90.278 ± 0.378	79.907 ± 0.786
	PiCO	91.665 ± 0.465	86.340 ± 0.230	57.713 ± 2.838	44.940 ± 2.008
	DPLL	**95.296 ± 0.074**	93.210 ± 0.436	92.901 ± 1.431	82.716 ± 1.091
	NCPD	95.139 ± 0.694	93.288 ± 0.232	92.439 ± 0.952	84.491 ± 0.231
	AsyCo	95.062 ± 0.437	**93.982 ± 0.756**	**93.235 ± 0.948**	**86.728 ± 0.787**
	Improv.	–	↑ **0.155**	↑ **0.334**	↑ **2.237**

Table 5 Accuracy (mean±std) (%) comparison on the real-world dataset BirdSong. The best performance in each column is highlighted in bold. The second best performance in each column is underlined. The improvement is calculated based on the best competitor and is highlighted in bold.

Model	BirdSong
CC	71.420 ± 0.900
RC	70.263 ± 0.274
PRODEN	70.623 ± 0.845
PiCO	71.700 ± 0.800
DPLL	72.093 ± 0.285
NCPD	66.960 ± 1.100
AsyCo	**72.770 ± 0.070**
Improv.	↑ **0.677**

Table 6 Accuracy (mean±std) (%) comparison on CIFAR-10 and SVHN with instance-dependent partial labels. The best performance in each column is highlighted in bold. The second best performance in each column is underlined. The improvement is calculated based on the best competitor and is highlighted in bold.

Model	SVHN	CIFAR-10
CC	96.072 ± 0.041	93.701 ± 0.006
RC	96.899 ± 0.087	93.270 ± 0.013
PRODEN	95.626 ± 0.084	92.409 ± 0.041
PiCO	95.615 ± 0.045	92.715 ± 0.055
DPLL	95.796 ± 0.015	93.657 ± 0.104
NCPD	96.633 ± 0.056	94.011 ± 0.011
AsyCo	**97.528 ± 0.008**	**95.301 ± 0.046**
Improv.	↑ **0.895**	↑ **1.290**

respectively.

As shown in Tables 4 and 5, on the two non-image datasets, i.e., CNAE-9 and BirdSong, AsyCo still outperforms all the competitors in most situations. In addition, on CNAE-9, AsyCo also shows good robustness as q increases, and the accuracy advantage over the competitors becomes more and more obvious, which is the same as that on the image dataset. This shows that AsyCo can be well generalized to datasets in different domains.

Last but not least, in addition to the outstanding performance on the uniformly generated partially labeled datasets, as shown in Table 6, AsyCo outperforms other models when applied to instance-dependent partially labeled datasets. It exhibits a notable performance increase of 1.290% on CIFAR-10 and achieves an accuracy improvement of 0.895% on SVHN, validating the capabilities and generalizability of AsyCo.

Table 7 Ablation study (%) of co-training, where the disambiguation network is trained individually. The performance degradation ratio compared to the original AsyCo is highlighted in bold.

Dataset	$q = 0.3$	$q = 0.5$	$q = 0.7$
SVHN	97.323 ± 0.021 (↓ **0.148**)	97.154 ± 0.090 (↓ **0.399**)	96.238 ± 0.069 (↓ **1.301**)
Dataset	$q = 0.3$	$q = 0.5$	$q = 0.7$
CIFAR-10	96.191 ± 0.077 (↓ **0.088**)	95.746 ± 0.066 (↓ **0.257**)	94.455 ± 0.102 (↓ **1.095**)
Dataset	$q = 0.05$	$q = 0.1$	$q = 0.2$
CIFAR-100	79.969 ± 0.136 (↓ **0.464**)	79.263 ± 0.056 (↓ **0.405**)	77.921 ± 0.020 (↓ **0.140**)

Table 8 Impact (%) of asymmetric co-training, where model variant SyCo is trained based on symmetric co-training strategy, that is, the two networks are both trained by PLL tasks. Both the performance of AsyCo and the performance degradation ratio compared to AsyCo are highlighted in bold.

	SVHN, $q = 0.7$	CIFAR-10, $q = 0.7$	CIFAR-100, $q = 0.2$
AsyCo	**97.539 ± 0.013**	**95.550 ± 0.007**	**78.061 ± 0.001**
SyCo (Symmetric co-training model variant)	96.584 ± 0.066 ↓ **0.955**	94.733 ± 0.210 ↓ **0.607**	78.028 ± 0.004 ↓ **0.033**

Table 9 Ablation study (%) of error correction strategies. Both the performance of AsyCo and the performance degradation ratio compared to AsyCo are highlighted in bold.

	SVHN, $q = 0.7$	CIFAR-10, $q = 0.7$
AsyCo	**97.539 ± 0.013**	**95.550 ± 0.007**
w/o distillation	97.350 ± 0.053 (↓ **0.189**)	95.400 ± 0.030 (↓ **0.150**)
w/o confidence refinement	97.481 ± 0.057 (↓ **0.058**)	95.456 ± 0.062 (↓ **0.094**)

4.3 Ablation study

4.3.1 *Impact of the auxiliary network*

In order to examine the impact of the auxiliary network, we conduct an experiment by excluding the auxiliary network from AsyCo and training the disambiguation network separately. This setting is equivalent to directly removing the error correction module from AsyCo, thereby eliminating the impact of the auxiliary network on the disambiguation network. The experimental results, shown in Table 7, demonstrate a decrease in performance across all datasets and p values, indicating that co-training can enhance the prediction accuracy of PLL. Particularly, AsyCo exhibits a more pronounced advantage as the value of q increases on CIFAR-10 and SVHN, providing further evidence that co-training enhances the robustness of the model.

4.3.2 *Impact of asymmetric co-training architecture*

Different from existing co-training PLL models, AsyCo is built upon asymmetric co-training architecture. In this subsection, we construct a symmetric co-training model variant called SyCo to explore the impact of the asymmetric co-training architecture. In SyCo, the two networks are initialized differently and both trained with the same PLL loss as (8). Additionally, SyCo employs a symmetric KL divergence-based distillation loss to enable the two networks to interact with each other for error correction. The performance comparison between SyCo and AsyCo is presented in Table 8, which reveals a significant decrease in accuracy, particularly on CIFAR-10 and SVHN. This finding demonstrates that the asymmetric dual-task co-training employed in AsyCo is more effective than traditional symmetric co-training techniques. The underlying reason can be the fact that training the two networks with distinct tasks compels them to explicitly learn from different perspectives. Therefore, the two networks have a higher likelihood of acquiring complementary information, thereby avoiding the error accumulation via communicating with each other.

4.3.3 *Impact of error correction strategy*

We perform an ablation study of the two error correction strategies, namely distillation and confidence refinement. The experimental results are presented in Table 9. It is evident that removing either strategy

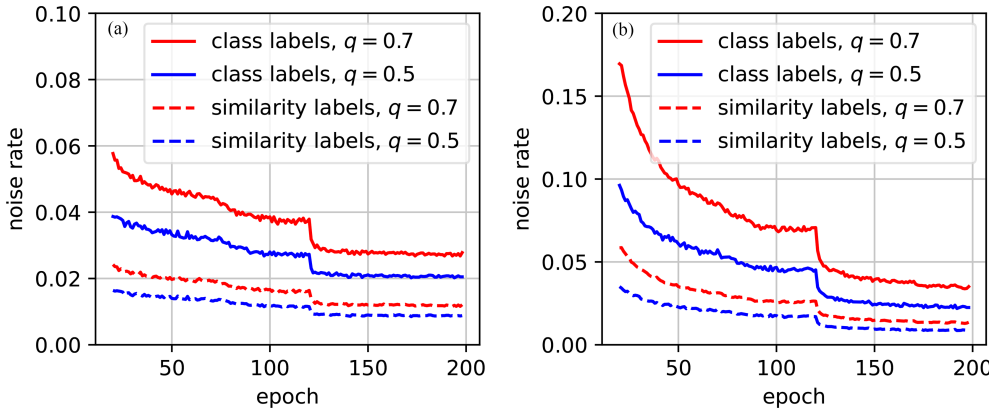

Figure 4 (Color online) Comparison in terms of noise rate between pairwise similarity labels and pseudo class labels during training AsyCo. (a) Noise rates on SVHN; (b) noise rates on CIFAR-10.

Table 10 Prediction accuracy (%) when the auxiliary network is trained based on the pseudo class labels. Both the performance of AsyCo and the performance degradation ratio compared to AsyCo are highlighted in bold.

	SVHN, $q = 0.7$	CIFAR-10, $q = 0.7$	CIFAR-100, $q = 0.2$
AsyCo	**97.539 ± 0.013**	**95.550 ± 0.007**	**78.061 ± 0.001**
AsyCo trained with the pseudo class labels	97.380 ± 0.010 ↓ **0.159**	95.164 ± 0.015 ↓ **0.386**	77.489 ± 0.053 ↓ **0.572**

Table 11 Ablation study (%) of data augmentation. The best performance in each column is highlighted in bold.

	SVHN, $q = 0.7$	CIFAR-10, $q = 0.7$	CIFAR-100, $q = 0.2$
w/o D.A.	96.290 ± 0.019	93.093 ± 0.001	74.627 ± 0.195
w/ 1 D.A.	**97.705 ± 0.001**	95.475 ± 0.033	77.844 ± 0.350
w/ 2 D.A. (the original AsyCo)	97.539 ± 0.013	**95.550 ± 0.007**	**78.061 ± 0.001**
w/ 3 D.A.	97.166 ± 0.262	95.415 ± 0.017	77.985 ± 0.065

leads to a slight decrease in accuracy, which indicates that the combination of them plays a more substantial role. Referring to the significant performance decrease caused by removing the auxiliary network, i.e., removing the whole error correction module shown in Table 7, it can be inferred that either of the proposed error correction strategies contributes to the final classification accuracy. Moreover, removing distillation results in a larger decline in performance compared to removing confidence refinement. This may result from the fact that distillation facilitates a more direct and timely influence of disambiguation models on auxiliary models.

4.3.4 *Impact of label transformation*

The impact of converting pseudo class labels into pairwise similarity labels is analyzed from two aspects. First, a comparison is conducted between the noise rates of pairwise similarity labels and pseudo class labels while optimizing AsyCo, as depicted in Figure 4. It is obvious that the noise rate of similarity labels is significantly lower than that of the noisy class labels. Second, a model variant of AsyCo is constructed that disregards label transformation and instead trains the auxiliary network directly using pseudo class labels. The degradation in performance, as demonstrated in Table 10, clearly illustrates that transforming noisy class labels into noisy pairwise similarity labels reduces the influence of noise labels on prediction accuracy.

4.3.5 *Ablation study of data augmentation*

In order to investigate the impact of data augmentation, we construct model variants based on AsyCo by removing data augmentation and varying the number of data augmentations per instance, as shown in Table 11. The experimental results show that both removing data augmentation and over-increasing the number of times each sample is augmented, e.g., 3 augmentations per sample, bring about a decrease in classification accuracy. Generally, the model performs best on the CIFAR datasets with two data

Table 12 Impact (%) of backbone networks, where ResNet-18 is replaced with Wide-ResNet-34-10. Both the best performance in each column and the performance improvement ratio compared to the original AsyCo are highlighted in bold.

	SVHN, $q = 0.7$	CIFAR-10, $q = 0.7$
DPLL	96.627 ± 0.115 (↑ **1.655**)	94.910 ± 0.060 (↑ **2.753**)
AsyCo	**97.941 ± 0.104 (↑ 0.402)**	**96.643 ± 0.023 (↑ 1.093)**

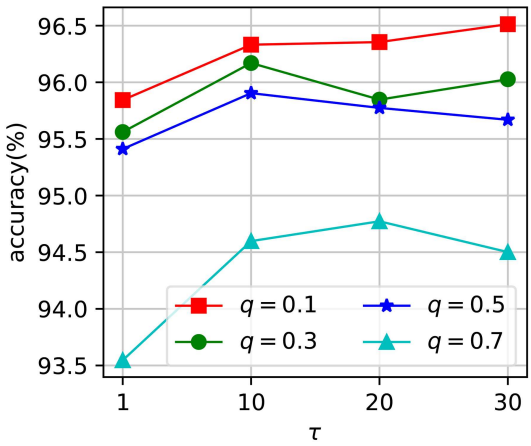

Figure 5 (Color online) Impact of the temperature parameter τ on prediction accuracy on CIFAR-10.

augmentations for each instance. This experiment illustrates the contribution of data augmentation to classification accuracy and the importance of choosing a proper number of data augmentations.

4.4 Further analysis

4.4.1 *Impact of backbone network*

We replace the backbone network ResNet-18 with Wide-ResNet-34-10 in order to analyze the impact of the backbone network on performance. The results presented in Table 12 demonstrate that a stronger backbone network can lead to additional performance improvement in AsyCo. Specifically, when ResNet-18 is replaced with Wide-ResNet, we observe accuracy improvements of approximately 0.402 % and 1.093% on SVHN and CIFAR-10, respectively. These findings indicate the substantial performance potentials of AsyCo. As a comparison, we also investigate the performance of replacing the backbone network for the competitors. And due to space limitations, we list the accuracy of the strongest competitor, i.e., DPLL, on the CIFAR dataset after adopting Wide-ResNet as its backbone network. It can be seen that a stronger backbone network leads DPLL to a significant performance improvement, though the performance of DPLL is still significantly behind on the SVHN and CIFAR-10 datasets compared to AsyCo.

4.4.2 *Impact of temperature parameter τ*

The temperature coefficient τ in (4) determines the smoothness of the classification probability. Here, we study the effect of τ on the disambiguation network. As shown in Figure 5, the accuracy of the disambiguation network is visualized when different values of τ (1, 10, 20, and 30) are applied, where the disambiguation network is trained individually. It is observed that an improvement in performance is evident when $\tau > 1$, regardless of the q values. The reason is that properly smoothed confidence avoids aggressive optimization and reflects detailed information of instances, e.g., the correlation between the instance and false positive labels. However, excessively smooth confidence resulting from a very large value of τ leads to a degradation in performance. This is due to the inability to clearly differentiate the weights of each candidate label.

5 Related work

5.1 Traditional partial-label learning

Traditional PLL methods can be divided into two categories, i.e., average-based methods and identification-based methods. The average-based methods treat each label in a candidate label set equally [5, 31–33]. However, the ground truth label of each instance is easily overwhelmed, especially when the number of candidate labels is large. To alleviate the problem, identification-based methods try to disambiguate ground-truth labels from candidate label sets. Some of them utilize a two-phase strategy [32], i.e., first refining label confidence, then learning the classifier, while others progressively refine confidence during learning the classifier [9]. Besides, manifold consistency regularization, which assumes that similar instances are supposed to have similar label distributions, has been widely employed in PLL to estimate the label confidence and learn the classifier simultaneously [9, 10, 32, 34]. Recently, an algorithm toward multi-instance PLL problem has been explored [35], which assumes each training sample is associated with not only multiple instances but also a candidate label set that contains one ground-truth label and some false positive labels. However, these traditional methods are usually linear or kernel-based models, which are hard to deal with large-scale datasets.

5.2 Deep partial-label learning

With the powerful modeling capability of deep learning, deep PLL methods can handle high-dimensional features and outperform traditional methods. Assuming a uniform set-level partial label generation process, Feng et al. [18] proposed a classifier-consistent loss, which is model-agnostic and can be directly combined with deep classifiers and variant optimizers. However, it treats each candidate label equally. RC [18], PRODEN [19], and LWS [17] leverage self-training and estimate label confidence and train the model with it iteratively. PiCO [15] and DPLL [16] further explore contrastive learning and manifold consistency in self-training PLL models, respectively. Nevertheless, self-training PLL models suffer from error accumulation problems resulting from mistakenly disambiguated instances. To address this issue, NCPD [25] converts the partial labels into noisy labels via multi-birth duplication and adopts a typical co-training NLL method called co-teaching [24]. Unfortunately, the label transformation in NCPD results in a high noise rate, limiting classification accuracy and resulting in high time and space complexity. Besides, the two networks in NCPD are trained with the same input data and loss functions and easily reach a consensus, thereby cannot correct errors for each other effectively, which naturally motivates us to improve them in our research.

6 Conclusion

In this paper, we propose an asymmetric dual-task co-training PLL model AsyCo, which forces them to learn from different views explicitly by training a disambiguation network and an auxiliary network via optimizing different tasks. To alleviate the error accumulation problem of self-training PLL models, we establish an information flow loop between the two networks in AsyCo as their collaboration mechanism, i.e., the disambiguation network provides the auxiliary network with the identified pseudo class labels, while the auxiliary network conducts error correction for the disambiguation network through distillation and confidence refinement. Results of experiments on benchmark datasets fully demonstrate the superior performance of AsyCo compared to existing PLL models and the effectiveness of asymmetric co-training in error elimination. Though AsyCo achieves excellent performance, it also has limitations. Like other co-training-based models, it requires almost twice the computational space to complete the training, which brings higher training overhead. In the future, we will further conduct research on different co-training architectures and network cooperation mechanisms to tap the potential of dual-task co-training models for PLL.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 62106028, 62072450) and Chongqing Overseas Chinese Entrepreneurship and Innovation Support Program. The authors wish to thank the associate editor and anonymous reviewers for their helpful comments and suggestions.

References

1 Zhou Z-H. A brief introduction to weakly supervised learning. Natl Sci Rev, 2018, 5: 44–53
2 Yao J R, Han J W, Zhang D W. A weakly supervised object detection approach using point annotation (in Chinese). Sci Sin Inform, 2022, 52: 461–482

3 Feng J P, Wang X G, Liu W Y. Deep graph cut network for weakly-supervised semantic segmentation. Sci China Inf Sci, 2021, 64: 130105
4 Xu M, Guo L-Z. Learning from group supervision: the impact of supervision deficiency on multi-label learning. Sci China Inf Sci, 2021, 64: 130101
5 Cour T, Sapp B, Taskar B. Learning from partial labels. J Mach Learn Res, 2011, 12: 1501–1536
6 Yang W J, Li C Q, Jiang L X. Learning from crowds with robust support vector machines. Sci China Inf Sci, 2023, 66: 132103
7 Chen C H, Patel V M, Chellappa R. Learning from ambiguously labeled face images. IEEE Trans Pattern Anal Mach Intell, 2018, 40: 1653–1667
8 Zeng Z, Xiao S, Jia K, et al. Learning by associating ambiguously labeled images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2013. 708–715
9 Yu F, Zhang M L. Maximum margin partial label learning. In: Proceedings of Asian Conference on Machine Learning, 2016. 45: 96–111
10 Zhang M L, Zhou B B, Liu X Y. Partial label learning via feature-aware disambiguation. In: Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining, 2016. 1335–1344
11 Lyu G, Feng S, Wang T, et al. GM-PLL: graph matching based partial label learning. IEEE Trans Knowl Data Eng, 2021, 33: 521–535
12 Wang D-B, Li L, Zhang M L. Adaptive graph guided disambiguation for partial label learning. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019. 83–91
13 Xu N, Lv J, Geng X. Partial label learning via label enhancement. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2019. 5557–5564
14 Liu L P, Dietterich T G. A conditional multinomial mixture model for superset label learning. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, 2012. 1: 548–556
15 Wang H, Xiao R, Li Y, et al. Contrastive label disambiguation for partial label learning. In: Proceedings of the International Conference on Learning Representations, 2022
16 Wu D D, Wang D B, Zhang M L. Revisiting consistency regularization for deep partial label learning. In: Proceedings of the 39th International Conference on Machine Learning, 2022. 162: 212–225
17 Wen H, Cui J, Hang H, et al. Leveraged weighted loss for partial label learning. In: Proceedings of the 38th International Conference on Machine Learning, 2021. 139: 91–100
18 Feng L, Lv J, Han B, et al. Provably consistent partial-label learning. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 33: 948–960
19 Lv J, Xu M, Feng L, et al. Progressive identification of true labels for partial-label learning. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 6500–6510
20 Ma F, Meng D, Xie Q, et al. Self-paced co-training. In: Proceedings of the 34th International Conference on Machine Learning, 2017. 70: 2275–2284
21 Blum A, Mitchell T. Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th Annual Conference on Computational Learning Theory, 1998. 92–100
22 Malach E, Shalev-Shwartz S. Decoupling "when to update" from "how to update". In: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017. 961–971
23 Wei H, Feng L, Chen X, et al. Combating noisy labels by agreement: a joint training method with co-regularization. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, 2020. 723–732
24 Han B, Yao Q, Yu X, et al. Co-teaching: robust training of deep neural networks with extremely noisy labels. In: Proceedings of Advances in Neural Information Processing Systems, 2018. 31
25 Yao Y, Gong C, Deng J, et al. Network cooperation with progressive disambiguation for partial label learning. In: Proceedings of Machine Learning and Knowledge Discovery in Databases: European Conference, 2020. 471–488
26 Wu S, Xia X, Liu T, et al. Class2Simi: a noise reduction perspective on learning with noisy labels. In: Proceedings of the 38th International Conference on Machine Learning, 2021. 285–295
27 Cubuk E D, Zoph B, Mané D, et al. AutoAugment: learning augmentation strategies from data. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 113–123
28 DeVries T, Taylor G W. Improved regularization of convolutional neural networks with Cutout. 2017. ArXiv:1708.04552
29 Netzer Y, Wang T, Coates A, et al. Reading digits in natural images with unsupervised feature learning. In: Proceedings of Workshop on Deep Learning and Unsupervised Feature Learning, 2011
30 Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. 2009. https://api.semanticscholar.org/CorpusID:18268744
31 Hüllermeier E, Beringer J. Learning from ambiguously labeled examples. In: Advances in Intelligent Data Analysis. Berlin: Springer, 2006. 419–439
32 Zhang M L, Yu F. Solving the partial label learning problem: an instance-based approach. In: Proceedings of the 24th International Conference on Artificial Intelligence, 2015. 4048–4054
33 Zhang M L, Yu F, Tang C Z. Disambiguation-free partial label learning. IEEE Trans Knowl Data Eng, 2017, 29: 2155–2167
34 Feng L, An B. Partial label learning with self-guided retraining. In: Proceedings of AAAI Conference on Artificial Intelligence, 2019. 33: 3542–3549
35 Tang W, Zhang W J, Zhang M-L. Multi-instance partial-label learning: towards exploiting dual inexact supervision. Sci China Inf Sci, 2024, 67: 132103