PiCO+: Contrastive Label Disambiguation for Robust Partial Label Learning

Haobo Wang, Ruixuan Xiao, Yixuan Li, *Member, IEEE*, Lei Feng, Gang Niu, Gang Chen, *Member, IEEE*, and Junbo Zhao

***Abstract*—Partial label learning (PLL) is an important problem that allows each training example to be labeled with a coarse candidate set with the ground-truth label included. However, in a more practical but challenging scenario, the annotator may miss the ground-truth and provide a wrong candidate set, which is known as the noisy PLL problem. To remedy this problem, we propose the PiCO+ framework that simultaneously disambiguates the candidate sets and mitigates label noise. Core to PiCO+, we develop a novel label disambiguation algorithm PiCO that consists of a contrastive learning module along with a novel class prototype-based disambiguation method. Theoretically, we show that these two components are mutually beneficial, and can be rigorously justified from an expectation-maximization (EM) algorithm perspective. To handle label noise, we extend PiCO to PiCO+, which further performs distance-based clean sample selection, and learns robust classifiers by a semi-supervised contrastive learning algorithm. Beyond this, we further investigate the robustness of PiCO+ in the context of out-of-distribution noise and incorporate a novel energy-based rejection method for improved robustness. Extensive experiments demonstrate that our proposed methods significantly outperform the current state-of-the-art approaches in standard and noisy PLL tasks and even achieve comparable results to fully supervised learning.**

***Index Terms*—Contrastive learning, noisy label learning, partial label learning, prototype-based disambiguation.**

Manuscript received 26 October 2022; revised 14 October 2023; accepted 28 November 2023. Date of publication 13 December 2023; date of current version 3 April 2024. This work was supported by NSFC under Grant 62206247. The work of Yixuan Li was supported in part by AFOSR Young Investigator Program under Grant FA9550-23-1-0184, in part by National Science Foundation (NSF) under Grants IIS-2237037 and IIS-2331669, in part by the Office of Naval Research under Grant N00014-23-1-2643, in part by Philanthropic Fund from SFF, and in part by faculty research awards/gifts from Google and Meta. The work of Lei Feng was supported by the National Natural Science Foundation of China under Grant 62106028. The work of Junbo Zhao was supported by the Fundamental Research Funds for the Central Universities under Grant 226-2022-00028. Recommended for acceptance by A. Kumar. *(Corresponding author: Junbo Zhao.)*

Haobo Wang is with the School of Software Technology, Zhejiang University, Zhejiang 310027, China (e-mail: wanghaobo@zju.edu.cn).

Ruixuan Xiao, Gang Chen, and Junbo Zhao are with the College of Computer Science and Technology, Zhejiang University, Zhejiang 310027, China (e-mail: xiaoruixuan@zju.edu.cn; cg@zju.edu.cn; j.zhao@zju.edu.cn).

Yixuan Li is with the Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI 53701 USA (e-mail: sharonli@cs.wisc.edu).

Lei Feng is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: lfeng@cqu.edu.cn).

Gang Niu is with the RIKEN Center for Advanced Intelligence Project, Tokyo 103-0027, Japan (e-mail: gang.niu.ml@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3342650, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3342650

I. Introduction

THE training of modern deep neural networks typically requires massive labeled data, which imposes formidable obstacles in data collection. Of a particular challenge, data annotation in the real-world can naturally be subject to inherent label ambiguity and noise. For example, as shown in Fig. 1, identifying an Alaskan Malamute from a Siberian Husky can be difficult for a human annotator. The issue of labeling ambiguity is prevalent yet often overlooked in many applications, such as web mining [1] and automatic image annotation [2]. This gives rise to the importance of *partial label learning* (PLL) [3], [4], where each training example is equipped with a set of candidate labels instead of the exact ground-truth label. This stands in contrast to its supervised counterpart where one label must be chosen as the "gold". Arguably, the PLL problem is deemed more common and practical in various situations due to its relatively lower cost to annotations.

The standard PLL setup ideally assumes that the ground-truth label is guaranteed to be included in the candidate set. However, lacking domain knowledge, it is likely the annotators take the wrong set of labels as the candidates and dismiss the true one. Lv et al. [5] formalize this problem as *noisy partial label learning* (noisy PLL). But, they focus on analyzing the theoretical robustness of existing PLL methods instead of providing new solutions. Experimentally, we find that current best-performing PLL algorithms, including PiCO, display degenerated performance in the noisy PLL setup (Section IV-C), e.g., the accuracy of PRODEN drops -10.62% on CIFAR-10 with 20% wrong candidate sets. The main reason is that the label disambiguation procedure of current PLL methods is restricted to the candidate labels, which causes severe overfitting on wrong labels. Notably, in a more general setup, the data sources may further contain out-of-distribution (OOD) noise, which significantly challenges the robustness of existing PLL algorithms. Fig. 1 illustrates the noisy PLL problems with in-distribution noise and out-of-distribution noise.

There are two challenging issues to handling the noisy PLL problem. The first is label disambiguation, i.e., identifying the ground-truth label from the candidate label set, which comes from the nature of partial labeling. The second is to prevent the model from overfitting on the wrong candidate sets. Notably, existing PLL methods [6], [7], [8] mostly focus on the label disambiguation issue and operate under the assumption that data points closer in the feature space are more likely to share the

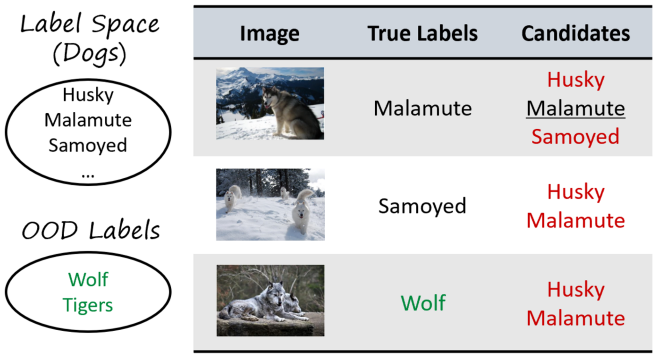


Fig. 1. *Top:* (Standard PLL) A Malamute image with three candidate labels. *Middle:* (PLL with in-distribution noise) A Samoyed image with the candidate set missing the Samoyed label. *Bottom:* (PLL with out-of-distribution noise) A Wolf image with the wrong dog species as candidate labels.

same ground-truth label. However, this strategy largely relies on an off-the-shelf good feature representation. When training deep networks from scratch, it leads to a non-trivial dilemma—the inherent label uncertainty and noise can undesirably manifest in the representation learning process—the quality of which may, in turn, prevent effective label disambiguation. To date, few efforts have been made to resolve this.

This paper bridges the gap by reconciling the intrinsic tension between the three highly dependent problems—representation learning, label disambiguation, and denoising—in one coherent and synergistic framework. Core to our PiCO+ framework is a **P**ar**t**ial label learning with **CO**ntrastive label disambiguation (dubbed **PiCO**) algorithm that produces closely aligned representations for examples from the same classes and facilitates label disambiguation. Specifically, PiCO is tailored for the standard PLL problem, which encapsulates two key components. First, we leverage contrastive learning (CL) [9] to partial label learning, which is unexplored in previous PLL literature. To mitigate the key challenge of constructing positive pairs, we employ the classifier's output and generate pseudo positive pairs for contrastive comparison (Section III-A1). Second, based on the learned embeddings, we propose a novel prototype-based label disambiguation strategy (Section III-A2). Key to our method, we gradually update the pseudo target for classification, based on the closest class prototype. By alternating the two steps above, PiCO converges to a solution with a highly distinguishable representation for accurate classification. On standard PLL benchmarks, PiCO establishes *state-of-the-art* performance on three benchmark datasets, outperforming the baselines by a significant margin (Section IV) and obtains results that are competitive with *fully supervised learning*.

To handle the noisy PLL problem, we then extend PiCO to PiCO+ which additionally incorporates two mechanisms. First, we present a novel distance-based clean sample detection technique that chooses near-prototype examples as clean. Second, to handle the remaining noisy examples, we develop a semi-supervised contrastive learning framework that generalizes the two PiCO components by (i)-*contrastive learning:* construct the positive set by noisy prediction and nearest neighbor embeddings; (ii)-*label disambiguation:* guess the prototype-based soft target for noisy samples. More interestingly, we show that PiCO+ can be robust to out-of-distribution noise as well, and can be further improved via an energy-based rejection mechanism. Through extensive experiments, PiCO+ exhibits significant improvement to state-of-the-art PLL approaches on noisy PLL benchmarks.

Theoretically, we demonstrate that our contrastive representation learning and prototype-based label disambiguation are mutually beneficial, and can be rigorously interpreted from an Expectation-Maximization (EM) algorithm perspective (Section V). First, the refined pseudo labeling improves contrastive learning by selecting pseudo positive examples accurately. This can be analogous to the E-step, where we utilize the classifier's output to assign each data example to one label-specific cluster. Second, better contrastive performance in turn improves the quality of representations and thus the effectiveness of label disambiguation. This can be reasoned from an M-step perspective, where the contrastive loss partially maximizes the likelihood by clustering similar data examples. Finally, the training data will be mapped to a mixture of von Mises-Fisher distributions on the unit hypersphere, which facilitates label disambiguation by using the component-specific label. With proper sample selection, our theoretical results also interpret the semi-supervised PLL procedure that proves the robustness of PiCO+ under label noise.

We have presented preliminary results of this work in [10]. In this paper, we further present a robust extension. Our *main contributions* are summarized as follows:

1) *(Methodology):* To the best of our knowledge, our paper pioneers the exploration of contrastive learning for partial label learning via PiCO and PiCO+.
2) *(Practicality):* Additionally, we propose PiCO+, an extension of PiCO, that targets to mitigate overfitting on noisy candidate sets. We also provide a primary investigation of the OOD noise and show the robustness of PiCO+ to such noise. We believe our work makes a serious attempt at improving the practicality of PLL in open-world environments.
3) *(Experiments):* Empirically, our proposed PiCO+ framework establishes the *state-of-the-art* performance on various PLL tasks, showing its robustness to label ambiguity, label noise, and out-of-distribution data.
4) *(Theory):* We theoretically interpret our framework from the expectation-maximization perspective. Our derivation is also generalizable to other contrastive learning methods and shows the *alignment* property in contrastive learning [11] mathematically equals the M-step in center-based clustering algorithms.

II. Background

In this section, we describe our problem setups.

Standard Partial Label Learning: Let $\mathcal{X}$ be the input space, and $\mathcal{Y} = \{1, 2, \ldots, C\}$ be the output label space. We consider a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, Y_i)\}_{i=1}^{n}$, where each tuple comprises of an image $\boldsymbol{x}_i \in \mathcal{X}$ and a *candidate label set* $Y_i \subset \mathcal{Y}$. Identical to the supervised learning setup, the goal of PLL is to obtain a functional mapping that predicts the one true label associated

with the input. Yet differently, the PLL setup bears significantly more uncertainty in the label space. A basic assumption of PLL is that the ground-truth label y_i is concealed in its candidate set, i.e., $y_i \in Y_i$, and is invisible to the learner. For this reason, the learning process can suffer from inherent ambiguity, compared with the supervised learning task with explicit ground-truth.

Noisy Partial Label Learning: In this paper, our main focus is a more practical setup called *noisy partial label learning* [5], where the true label potentially lies outside the candidate set, i.e., $y_i \notin Y_i$. We further consider a more challenging task where the collected datasets contain out-of-distribution noise [12]. In this case, the samples can come from different data distributions and the ground-truth labels are not even in the whole label space, i.e. $y_i \notin \mathcal{Y}$.

Learning Goals: The uniqueness of PLL is to identify the ground-truth label from the candidate label set. For the noisy PLL problem, it is also required to filter out those out-of-distribution samples and find the correct labels of remaining in-distribution (ID) samples. To achieve these goals, we assign each image $\boldsymbol{x}_i$ a normalized vector $\boldsymbol{s}_i \in [0,1]^C$ as the *pseudo target* during training, whose entries denote the probability of labels being the ground-truth. The total probability mass of 1 is allocated among candidate labels in Y_i. Note that $\boldsymbol{s}_i$ will be updated during the training procedure. Ideally, $\boldsymbol{s}_i$ should put more probability mass on the (unknown) ground-truth label y_i over the course of training. We train a classifier $f : \mathcal{X} \to [0,1]^C$ using cross-entropy loss, with $\boldsymbol{s}_i$ being the target prediction. The per-sample loss is given by:

$$\mathcal{L}_{\text{cls}}(f; \boldsymbol{x}_i, Y_i) = \sum_{j=1}^{C} -s_{i,j} \log(f^j(\boldsymbol{x}_i))$$
$$\text{s.t.} \quad \sum_{j \in Y_i} s_{i,j} = 1 \text{ and } s_{i,j} = 0, \forall j \notin Y_i, \tag{1}$$

where j denotes the indices of labels. $s_{i,j}$ denotes the jth pseudo target of $\boldsymbol{x}_i$. Here f is the softmax output of the networks and we denote f^j as its jth entry. In the remainder of this paper, we omit the sample index i when the context is clear. We proceed by describing our proposed framework.

III. The PiCO+ Framework

In this section, we describe our novel PiCO+ framework in detail. In Section III-A, we introduce our core component—**Par**t**i**al label learning with **CO**ntrastive label disambiguation (PiCO) algorithm, which encapsulates two key components for improved representation quality and high label disambiguation ability. Next, in Section III-B, we present the PiCO+ algorithm to handle label noise by wrapping PiCO into a semi-supervised contrastive partial label learning framework that performs distance-based clean sample selection for robust training. Lastly, in Section III-C, we further equip PiCO+ with an energy-based rejection mechanism to improve its robustness to out-of-distribution noise.

Algorithm 1: Pseudo-Code of PiCO (One Epoch).

Input: Training dataset $\mathcal{D}$, classifier f, query network g, key network g', momentum queue, uniform pseudo-labels $\boldsymbol{s}_i$ associated with $\boldsymbol{x}_i$ in $\mathcal{D}$, class prototypes $\boldsymbol{\mu}_j$ $(1 \le j \le C)$.
for $iter = 1, 2, \ldots,$ **do**
  sample a mini-batch B from $\mathcal{D}$
  `// query and key embeddings generation`
  $B_q = \{\boldsymbol{q}_i = g(\text{Aug}_q(\boldsymbol{x}_i)) | \boldsymbol{x}_i \in B\}$
  $B_k = \{\boldsymbol{k}_i = g'(\text{Aug}_k(\boldsymbol{x}_i)) | \boldsymbol{x}_i \in B\}$
  $A = B_q \cup B_k \cup$ queue
  for $\boldsymbol{x}_i \in B$ **do**
    `// classifier prediction`
    $\tilde{y}_i = \arg\max_{j \in Y_i} f^j(\text{Aug}_q(\boldsymbol{x}_i))$
    `// momentum prototype updating`
    $\boldsymbol{\mu}_c = \text{Normalize}(\gamma \boldsymbol{\mu}_c + (1-\gamma)\boldsymbol{q}_i)$, if $\tilde{y}_i = c$
    `// positive set generation`
    $P(\boldsymbol{x}_i) = \{\boldsymbol{k}' | \boldsymbol{k}' \in A(\boldsymbol{x}_i), \tilde{y}' = \tilde{y}_i\}$
  end
  `// prototype-based label disambiguation`
  for $\boldsymbol{q}_i \in B_q$ **do**
    $\boldsymbol{z}_i = \text{OneHot}(\arg\max_{j \in Y_i} \boldsymbol{q}_i^\top \boldsymbol{\mu}_j)$
    $\boldsymbol{s}_i = \phi \boldsymbol{s}_i + (1-\phi)\boldsymbol{z}_i$
  end
  `// network updating`
  minimize loss $\mathcal{L}_{\text{pico}} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{cont}}$
  `// update the key network and momentum queue`
  momentum update g' by using g
  enqueue B_k and classifier predictions and dequeue
end

A. PiCO Algorithm for Label Disambiguation

1) Contrastive Representation Learning for PLL: The uncertainty in the label space posits a unique obstacle to learning effective representations. In PiCO, we couple the classification loss in (1) with a contrastive term that facilitates a clustering effect in the embedding space. While contrastive learning has been extensively studied in recent literature, it remains untapped in the domain of PLL. The main challenge lies in the construction of a positive sample set. In conventional supervised contrastive learning frameworks [9], the positive sample pairs can be easily drawn according to the ground-truth labels. However, this is not straightforward in the setting of PLL.

Training Objective: To begin with, we describe the standard contrastive loss term. We adopt the most popular setups by closely following MoCo [13] and SupCon [9]. Given each sample $(\boldsymbol{x}, Y)$, we generate two views—a query view and a key view—by way of randomized data augmentation $\text{Aug}(\boldsymbol{x})$. The two images are then fed into a query network $g(\cdot)$ and a key network $g'(\cdot)$, yielding a pair of L_2-normalized embeddings $\boldsymbol{q} = g(\text{Aug}_q(\boldsymbol{x}))$ and $\boldsymbol{k} = g'(\text{Aug}_k(\boldsymbol{x}))$. In implementations, the query network shares the same convolutional blocks as the classifier, followed by a prediction head (see Fig. 2). Following MoCo, the key network uses a momentum update with the query network. We additionally maintain a queue storing the most current key embeddings $\boldsymbol{k}$, and we update the queue chronologically. To this end, we have the following contrastive

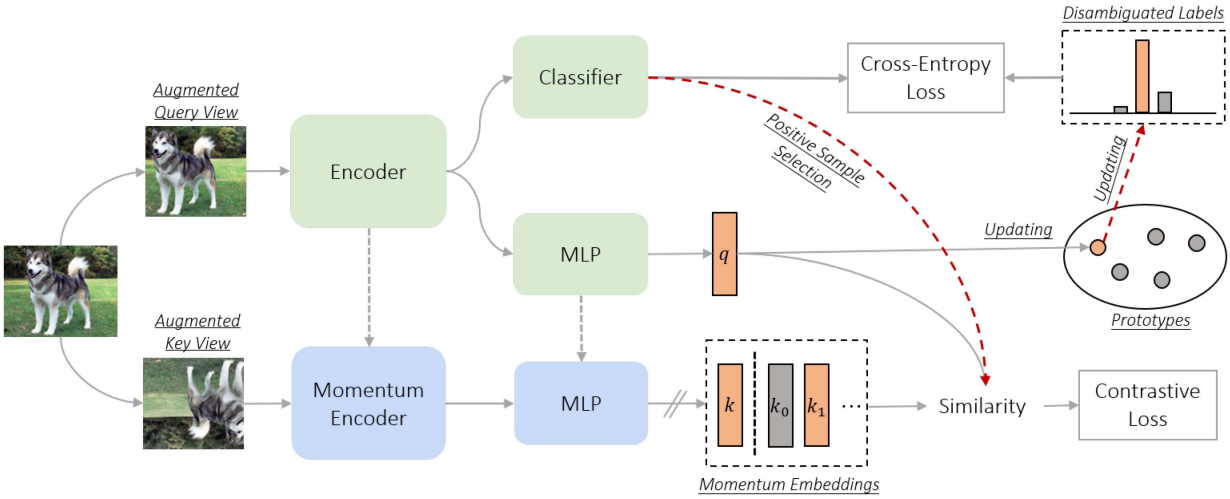


Fig. 2. Illustration of PiCO. The classifier's output is used to determine the positive peers for contrastive learning. The contrastive prototypes are then used to gradually update the pseudo target. The momentum embeddings are maintained by a queue structure. '//' means stop gradient.

embedding pool:

$$A = B_q \cup B_k \cup \text{queue}, \tag{2}$$

where B_q and B_k are vectorial embeddings corresponding to the query and key views of the current mini-batch. Given an example $\boldsymbol{x}$, the per-sample contrastive loss is defined by contrasting its query embedding with the remainder of the pool A,

$$\mathcal{L}_{\text{cont}}(g; \boldsymbol{x}, \tau, A) = -\frac{1}{|P(\boldsymbol{x})|} \sum_{\boldsymbol{k}_+ \in P(\boldsymbol{x})} \log \frac{\exp(\boldsymbol{q}^\top \boldsymbol{k}_+/\tau)}{\sum_{\boldsymbol{k}' \in A(\boldsymbol{x})} \exp(\boldsymbol{q}^\top \boldsymbol{k}'/\tau)}, \tag{3}$$

where $P(\boldsymbol{x})$ is the positive set and $A(\boldsymbol{x}) = A \backslash \{\boldsymbol{q}\}$. $\tau \geq 0$ is the temperature.

Positive Set Selection: As mentioned earlier, the crucial challenge is how to construct the positive set $P(\boldsymbol{x})$. We propose utilizing the predicted label $\tilde{y} = \arg\max_{j \in Y} f^j(\text{Aug}_q(\boldsymbol{x}))$ from the classifier. Note that we restrict the predicted label to be in the candidate set Y. The positive examples are then selected as follows,

$$P(\boldsymbol{x}) = \{\boldsymbol{k}' | \boldsymbol{k}' \in A(\boldsymbol{x}), \tilde{y}' = \tilde{y}\}. \tag{4}$$

where $\tilde{y}'$ is the predicted label for the corresponding training example of $\boldsymbol{k}'$. For computational efficiency, we also maintain a label queue to store past predictions. In other words, we define the positive set of $\boldsymbol{x}$ to be those examples carrying the same *approximated* label prediction $\tilde{y}$. Despite its simplicity, we show that our selection strategy can be theoretically justified (Section V) and also lead to superior empirical results (Section IV). Note that more sophisticated selection strategies can be explored, for which we discuss in Appendix B.5, available online. Putting it all together, we jointly train the classifier as well as the contrastive network. The overall loss function is:

$$\mathcal{L}_{\text{pico}} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{cont}}. \tag{5}$$

Still, our goal of learning high-quality representation by contrastive learning relies on accurate classifier prediction for positive set selection, which remains unsolved in the presence of label ambiguity. To this end, we further propose a novel label disambiguation mechanism based on contrastive embeddings and show that these two components are mutually beneficial.

2) Prototype-Based Label Disambiguation: As we mentioned (and later theoretically prove in Section V), the contrastive loss poses a clustering effect in the embedding space. As a collaborative algorithm, we introduce our novel prototype-based label disambiguation strategy. Importantly, we keep a *prototype* embedding vector $\boldsymbol{\mu}_c$ corresponding to each class $c \in \{1, 2, \ldots, C\}$, which can be deemed as a set of representative embedding vectors. Categorically, a naive version of the pseudo target assignment is to find the nearest prototype of the current embedding vector. Notably this primitive resembles a clustering step. We additionally soften this hard label assignment version by using a moving-average style formula. To this end, we may posit intuitively that the employment of the prototype builds a connection with the clustering effect in the embedding space brought by the contrastive term (Section III-A1). We provide a more rigorous justification in Section V.

Pseudo Target Updating: We propose a softened and moving-average style strategy to update the pseudo targets. Specifically, we first initialize the pseudo targets with a uniform distribution, $s_j = \frac{1}{|Y|}\mathbb{I}(j \in Y)$. We then iteratively update it by the following moving-average mechanism,

$$\boldsymbol{s} = \phi \boldsymbol{s} + (1 - \phi)\boldsymbol{z}, \quad z_c = \begin{cases} 1 & \text{if } c = \arg\max_{j \in Y} \boldsymbol{q}^\top \boldsymbol{\mu}_j, \\ 0 & \text{else} \end{cases} \tag{6}$$

where $\phi \in (0, 1)$ is a positive constant, and $\boldsymbol{\mu}_j$ is a *prototype* corresponding to the jth class. The intuition is that fitting uniform pseudo targets results in a good initialization for the classifier since the contrastive embeddings are less distinguishable at the beginning. The moving-average style strategy then smoothly

updates the pseudo targets toward the correct ones, and meanwhile ensures stable dynamics of training. This is supported by our quantitative results on the confidence curves as shown in Appendix B.1.4, available online. With more rigorous validation provided later in Section V, we provide an explanation for the prototype as follows: (i)-for a given input $\boldsymbol{x}$, the closest prototype is indicative of its ground-truth class label. At each step, $\boldsymbol{s}$ has the tendency to slightly move toward the one-hot distribution defined by $\boldsymbol{z}$ based on (6); (ii)-if an example consistently points to one prototype, the pseudo target $\boldsymbol{s}$ can converge (almost) to a one-hot vector with the least ambiguity.

Prototype Updating: The most canonical way to update the prototype embeddings is to compute it in every iteration of training. However, this would extract a heavy computational toll and in turn cause unbearable training latency. As a result, we update the class-conditional prototype vector similarly in a moving-average style:

$$\begin{aligned} \boldsymbol{\mu}_c &= \text{Normalize}(\gamma\boldsymbol{\mu}_c + (1-\gamma)\boldsymbol{q}), \\ \text{if } c &= \arg\max_{j\in Y} f^j(\text{Aug}_q(\boldsymbol{x}))), \end{aligned} \tag{7}$$

where the momentum prototype $\boldsymbol{\mu}_c$ of class c is defined by the moving-average of the normalized query embeddings $\boldsymbol{q}$ whose predicted class conforms to c. γ is a tunable hyperparameter.

3) Synergy Between Contrastive Learning and Label Disambiguation: While seemingly separated from each other, the two key components of PiCO work in a collaborative fashion. First, as the contrastive term favorably manifests a clustering effect in the embedding space, the label disambiguation module further leverages via setting more precise prototypes. Second, a set of well-polished label disambiguation results may, in turn, reciprocate the positive set construction which serves as a crucial part in the contrastive learning stage. The entire training process converges when the two components perform satisfactorily. We further rigorously draw a resemblance of PiCO with a classical EM-style clustering algorithm in Section V. Our experiments, particularly the ablation study displayed in Section IV-B2, further justify the mutual dependency of the synergy between the two components. The pseudo-code of our complete algorithm is shown in Algorithm 1.

B. PiCO+ for Handling Noisy Candidate Sets

In this section, our aim is to handle the more practical noisy PLL setup [5] where the annotators may produce wrong candidate sets. Empirically, we observe that the current PLL methods, including PiCO, exhibit a significant performance drop in such a setup (Section IV-C). One main reason is that these methods mostly rely on a within-set pseudo-label updating step, but the noisy candidate sets can mislead them toward overfitting on a wrong label. To this end, we propose PiCO+, an extension of PiCO, which learns robust classifiers from noisy partial labels. First, we introduce a distance-based sample selection mechanism that detects *clean examples* whose candidate sets contain the ground-truth labels. Then, we develop a semi-supervised contrastive PLL framework to handle data with noisy candidates. In what follows, we elaborate on our novel PiCO+ framework in detail. A visualized illustration of PiCO+ is shown in Fig. 3.

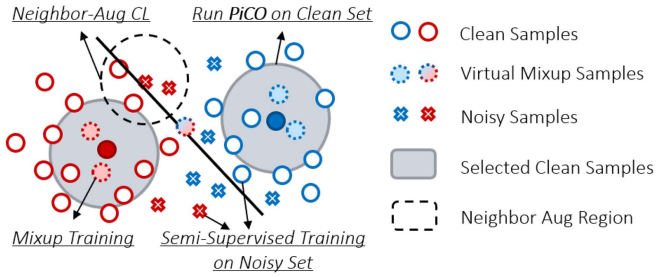


Fig. 3. Illustration of the semi-supervised contrastive learning procedure. A set of clean samples are selected for running PiCO. While the remaining noisy samples are optimized by semi-supervised objectives like neighbor-augmented contrastive learning and label-guessing training. We also incorporate mixup for improved robustness.

1) Distance-Based Clean Example Detection: To remedy overfitting on noisy candidates, we would like to select those reliable candidates, which contain the true labels, to run the PiCO method. In the noisy label learning regime, a widely-adopted strategy is to employ the small-loss selection criterion [14], which is based on the observation that noisy examples typically demonstrate a large loss. However, in the noisy PLL problem, the loss values are less informative since examples with more candidate labels can also incur a larger loss, even if the candidate sets are reliable.

To address this problem, we propose a distance-based selection mechanism as follows,

$$\mathcal{D}_{\text{clean}} = \{(\boldsymbol{x}_i, Y_i) | \boldsymbol{q}_i^\top \boldsymbol{\mu}_{\tilde{y}_i} > \kappa_\delta\}, \tag{8}$$

where $\tilde{y}_i = \arg\max_{j\in Y_i} f^j(\text{Aug}_q(\boldsymbol{x}_i)))$ is the classifier prediction. κ_δ is the $(100-\delta)$-percentile of the consine similarity between the query embedding and the $\tilde{y}_i$th prototype. For example, when $\delta = 60$, it means 60% of examples are above the threshold. Our motivation is that the clustering effect of contrastive learning makes the clean examples dominate the prototype calculation and thus they are distributed close to at least one prototype inside the candidate set. On the other hand, the noisy candidates mostly deviate from all candidate prototypes as their true labels are not contained in their candidate sets.

2) Semi-Supervised Contrastive Learning: While leveraging the clean examples to run a PiCO model is straightforward, the low data utility restricts it from better performance. Consequently, we regard noisy examples as unlabeled ones and develop a semi-supervised learning framework to learn from the two sets. On clean datasets, we assume the true labels are included in the candidate sets and run our PiCO method. It should be particularly noted that we also restrict the positive set construction and prototype updating procedures to merely employ clean examples. But, the pseudo-target updating is performed over all data points.

On the noisy dataset, which is denoted by $\mathcal{D}_{\text{noisy}} = \mathcal{D} \backslash \mathcal{D}_{\text{clean}}$, we follow our design patterns in PiCO to synergetically train the contrastive branch as well as the classifier. It is achieved by the following components:

Neighbor-Augmented Contrastive Learning: Recall that the crucial step for designing contrastive loss is to construct the positive set, which is challenging for noisy samples as their

Algorithm 2: Pseudo-Code of PiCO+.

Input: Training dataset $\mathcal{D}$, selection ratio δ, number of nearest neighbors k, beta distribution shape parameter ς, loss weighting factors α, β.
warm up by running PiCO on $\mathcal{D}$
for $epoch = 1, 2, \ldots,$ **do**
  split a clean set by $\mathcal{D}_{\text{clean}} = \{(\boldsymbol{x}_i, Y_i) | \boldsymbol{q}_i^\top \boldsymbol{\mu}_{\tilde{y}_i} > \kappa_\delta\}$
  set $\mathcal{D}_{\text{noisy}} = \mathcal{D} \backslash \mathcal{D}_{\text{clean}}$
  for $iter = 1, 2, \ldots,$ **do**
    sample a mini-batch B from $\mathcal{D}$
    $B_{\text{clean}} = B \cap \mathcal{D}_{\text{clean}}$, $B_{\text{noisy}} = B \cap \mathcal{D}_{\text{noisy}}$
    run PiCO on B_{clean} to get loss $\mathcal{L}_{\text{clean}}$
    for $\boldsymbol{x}_i \in B$ **do**
      `// label-driven positive set`
      $P_{\text{noisy}}(\boldsymbol{x}_i) = \{\boldsymbol{k}' | \boldsymbol{k}' \in A(\boldsymbol{x}_i), \hat{y}'_i = \hat{y}_i\}$
    end
    for $\boldsymbol{x}_i \in B_{noisy}$ **do**
      `// neighbors-based positive set`
      $P_{\text{kNN}}(\boldsymbol{x}_i) = \{\boldsymbol{k}' | \boldsymbol{k}' \in A(\boldsymbol{x}_i) \cap \mathcal{N}_k(\boldsymbol{x}_i)\}$
      `// label guessing`
      $s'_{ij} = \frac{\exp(\boldsymbol{q}_i^\top \boldsymbol{\mu}_j / \tau)}{\sum_{t=1}^{C} \exp(\boldsymbol{q}_i^\top \boldsymbol{\mu}_t / \tau)}, \quad \forall 1 \le j \le C$
    end
    for $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in B$ **do**
      $\sigma \sim \text{Beta}(\varsigma, \varsigma)$
      `// mixup samples and pseudo-targets`
      $\boldsymbol{x}_{ij}^m = \sigma \text{Aug}_q(\boldsymbol{x}_i) + (1-\sigma)\text{Aug}_q(\boldsymbol{x}_j)$
      $\boldsymbol{s}_{ij}^m = \sigma \hat{\boldsymbol{s}}_i + (1-\sigma)\hat{\boldsymbol{s}}_j$
    end
    if *With OOD Samples* **then**
      `// energy-based OOD rejection`
      calculate energy scores $\mathcal{E}(\boldsymbol{x})$ by Eq. (14)
      fit a GMM model on energy scores
      reject OOD samples by $p(\mathcal{G}|\mathcal{E}(\boldsymbol{x})) < 0.5$
    end
    calculate $\mathcal{L}_{\text{n-cont}}$, $\mathcal{L}_{\text{kNN}}$, $\mathcal{L}_{\text{n-cls}}$, $\mathcal{L}_{\text{mix}}$
    minimize loss
      $\mathcal{L}_{\text{pico+}} = \mathcal{L}_{\text{mix}} + \alpha \mathcal{L}_{\text{clean}} + \beta(\mathcal{L}_{\text{n-cont}} + \mathcal{L}_{\text{kNN}} + \mathcal{L}_{\text{n-cls}})$
  end
end

candidate sets are unreliable. To this end, we first propose a label-driven construction approach. By regarding noisy samples as unlabeled data, it is intuitive to treat all labels as candidates. This gives rise to the following noisy positive set,

$$P_{\text{noisy}}(\boldsymbol{x}) = \{\boldsymbol{k}' | \boldsymbol{k}' \in A(\boldsymbol{x}), \hat{y}' = \hat{y}\},$$
$$\text{where } \hat{y} = \begin{cases} \arg\max_{1 \le j \le C} f^j(\text{Aug}_q(\boldsymbol{x}))) & \text{if } \boldsymbol{x} \in \mathcal{D}_{\text{noisy}}, \\ \arg\max_{j \in Y} f^j(\text{Aug}_q(\boldsymbol{x}))) & \text{else.} \end{cases} \tag{9}$$

That is, we choose the within-set classifier prediction for clean examples and choose the full-label prediction for the remaining. We apply this noisy positive set to all data in $\mathcal{D}$ and calculate a noisy contrastive loss $\mathcal{L}_{\text{n-cont}}$ by (3). This objective serves as our ultimate goal of semi-supervised training that recovers the PiCO algorithm on clean PLL data and noisy unlabeled data.

Nevertheless, as we are less knowledgeable of noisy examples, our estimation on $P_{\text{noisy}}(\boldsymbol{x})$ can be inaccurate and assigns wrong cluster centers. To this end, we incorporate a data-driven technique to regularize the noisy contrastive loss. In specific, we collect the nearest neighbors of noisy samples to be positive peers,

$$P_{\text{kNN}}(\boldsymbol{x}) = \{\boldsymbol{k}' | \boldsymbol{k}' \in A(\boldsymbol{x}) \cap \mathcal{N}_k(\boldsymbol{x})\}, \tag{10}$$

where $\mathcal{N}_k(\boldsymbol{x})$ is the embedding set of $\boldsymbol{x}$'s k-nearest neighbors in the embedding space. Then, we calculate the kNN-based contrastive loss $\mathcal{L}_{\text{kNN}}$ on noisy examples. Note that the original contrastive learning objective naturally encourages the examples to be locally smooth by aligning augmented copies. Our neighbor-augmented loss further enhances this effect to ensure the examples in a local region share the same label. By then, the noisy examples are aligned with their local neighbors, which promotes their clustering effect toward the right labels.

Prototype-Based Label Guessing: Similarly, we would also like to identify the true labels on noisy examples to enhance the classifier training. Although the noisy examples are treated as unlabeled, it is not appropriate to directly set their labels to uniform labels $\frac{1}{C}$ as in PiCO, since the data separation procedure dynamically changes during training. To this end, we leverage the class prototypes to guess their pseudo-targets $\boldsymbol{s}'$ by,

$$s'_j = \frac{\exp(\boldsymbol{q}^\top \boldsymbol{\mu}_j / \tau)}{\sum_{t=1}^{C} \exp(\boldsymbol{q}^\top \boldsymbol{\mu}_t / \tau)}, \quad \forall 1 \le j \le C. \tag{11}$$

We calculate the cross-entropy loss on $\mathcal{D}_{\text{noisy}}$ as defined in (1), which is term as $\mathcal{L}_{\text{n-cls}}$.

Mixup Training: Recently, the mixup regularization technique has been widely adopted to improve the robustness of weakly-supervised learning algorithms [15], [16]. Therefore, we also incorporate it into PiCO+ for boosted performance. Formally, given a pair of images $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we create a virtual training example by linearly interpolating both,

$$\boldsymbol{x}^m = \sigma \text{Aug}_q(\boldsymbol{x}_i) + (1-\sigma)\text{Aug}_q(\boldsymbol{x}_j),$$
$$\boldsymbol{s}^m = \sigma \hat{\boldsymbol{s}}_i + (1-\sigma)\hat{\boldsymbol{s}}_j, \tag{12}$$

where $\sigma \sim \text{Beta}(\varsigma, \varsigma)$ and ς is a hyperparameter. Here, we take the pseudo-target of PiCO on clean examples, and the guessed label on noisy examples, i.e., $\hat{\boldsymbol{s}} = \boldsymbol{s}$ if $\boldsymbol{x} \in \mathcal{D}_{\text{clean}}$ else $\hat{\boldsymbol{s}} = \boldsymbol{s}'$. We define the mixup loss $\mathcal{L}_{\text{mix}}$ as the cross-entropy on $\boldsymbol{x}^m$ and $\boldsymbol{s}^m$.

Finally, we aggregate the above losses together,

$$\mathcal{L}_{\text{pico+}} = \mathcal{L}_{\text{mix}} + \alpha \mathcal{L}_{\text{clean}} + \beta(\mathcal{L}_{\text{n-cont}} + \mathcal{L}_{\text{kNN}} + \mathcal{L}_{\text{n-cls}}), \tag{13}$$

where $\mathcal{L}_{\text{clean}}$ is the PiCO loss on clean examples. Note that overtrusting the guessed labels and positive sets on noisy samples may cause confirmation bias [17] and make the model overfit on wrong labels. Empirically, we set a larger α and a smaller β (e.g., $\alpha = 2, \beta = 0.1$), and thus, the clean samples dominate the learning procedure to ensure favorable noisy example detection ability.

C. Mitigating Out-of-Distribution Noise

Lastly, we investigate the out-of-distribution robustness of PiCO+. In real-world applications, it is possible that some training samples come from different domains but are assigned candidate labels from the predefined label space $\mathcal{Y}$. With proper sample selection, we believe PiCO+ can still be robust to OOD noise. However, the semi-supervised learning procedure is inevitably

subject to OOD samples since it encourages all samples to be pushed toward the prototypes. In that case, the distance-based measures can be inappropriate for detecting OOD samples.

To address this problem, we equip PiCO+ with an energy-based rejection mechanism since OOD samples are known to hold larger energy [18]. Formally, we first calculate the energy score on the classifier by,

$$\mathcal{E}(\boldsymbol{x}) = -\tau' \log \sum_{i=1}^{C} \exp(\tilde{f}^i(\mathrm{Aug}_q(\boldsymbol{x}))/\tau') \tag{14}$$

where $\tilde{f}$ is the network output prior to the softmax layer and $\tau' = 1$ is the energy temperature. One may use a validation set to determine a threshold that separates OOD data, but this can be annoying in practice. Motivated by the noisy label learning literature [16], we fit a two-component Gaussian Mixture Model (GMM) to $\mathcal{E}(\boldsymbol{x})$. For each sample, it is considered to be OOD if its GMM posterior probability $p(\mathcal{G}|\mathcal{E}(\boldsymbol{x})) < 0.5$, where $\mathcal{G}$ is the Gaussian component with a smaller mean, i.e. smaller energy. Then, we reject these OOD data and use the remaining samples for running PiCO+.

IV. Experiments

In this section, we present our main results and part of the ablation results to show the effectiveness of PiCO and PiCO+ methods. In Section IV-B, we first examine the effectiveness of PiCO and its extension PiCO+ in standard PLL datasets. Then, we report the results on noisy PLL setups in Section IV-C. We refer readers to Appendix B, available online for more experimental results and analysis. Code and data are available at: https://github.com/hbzju/PiCO.

A. Setup

Datasets: First, we evaluate PiCO on two commonly used benchmarks — CIFAR-10 and CIFAR-100 [19]. Adopting the identical experimental settings in previous work [20], [21], we generate conventional partially labeled datasets by flipping negative labels $\bar{y} \neq y$ to false positive labels with a probability $q = P(\bar{y} \in Y | \bar{y} \neq y)$. In other words, all $C - 1$ negative labels have a uniform probability to be false positive and we aggregate the flipped ones with the ground-truth to form the candidate set. We consider $q \in \{0.1, 0.3, 0.5\}$ for CIFAR-10 and $q \in \{0.01, 0.05, 0.1\}$ for CIFAR-100. In Section IV-D1, we further evaluate our method on fine-grained classification tasks, where label disambiguation can be more challenging.

For the noisy PLL task, we introduce a noise-controlling parameter $\eta = 1 - P(y \in Y|y)$ that controls the probability of the ground-truth label not being selected as a candidate. As it is possible that some instances are not assigned any candidate and we simply re-generate the candidate set until it has at least one candidate label. We select $\eta \in \{0.1, 0.2\}$ for both datasets; results with stronger noisy ratios are shown in Section IV-C3.

Baselines: We choose the five best-performed partial label learning algorithms to date: 1) LWS [21] weights the risk function by means of a trade-off between losses on candidate labels and the remaining; 2) PRODEN [20] iteratively updates the latent label distribution in a self-training style; 3) CC [22] is a classifier-consistent method that assumes set-level uniform data generation process; 4) MSE and EXP [23] are two simple baselines that adopt mean square error and exponential loss as the risks. For the noisy PLL task, we additionally incorporate one baseline method GCE [5], [24] that generalizes the cross-entropy loss via negative Box-Cox transformation. We report the comparisons with the remaining baselines from [5] in Appendix B.8, available online. The hyperparameters are tuned according to the original methods. For all experiments, we report the mean and standard deviation based on 5 independent runs (with different random seeds).

Implementation Details: Following the standard experimental setup in PLL [21], [22], we split a clean validation set (10% of training data) from the training set to select the hyperparameters. After that, we transform the validation set back to its original PLL form and incorporate it into the training set to accomplish the final model training. We use an 18-layer ResNet as the backbone for feature extraction. Most of experimental setups for the contrastive network follow previous works [9], [13]. The projection head of the contrastive network is a 2-layer MLP that outputs 128-dimensional embeddings. We use two data augmentation modules SimAugment [9] and RandAugment [25] for query and key data augmentation respectively. Empirically, we find that even weak augmentation for key embeddings also leads to good results. The size of the queue that stores key embeddings is fixed to be 8192. The momentum coefficients are set as 0.999 for contrastive network updating and $\gamma = 0.99$ for prototype calculation. For pseudo target updating, we linearly ramp down ϕ from 0.95 to 0.8. The temperature parameter is set as $\tau = 0.07$. The loss weighting factor is set as $\lambda = 0.5$. The model is trained by a standard SGD optimizer with a momentum of 0.9 and the batch size is 256. We train the model for 800 epochs with cosine learning rate scheduling. We also empirically find that classifier warm-up leads to better performance when there are many candidates. Hence, we disable contrastive learning in the first 100 epoch for CIFAR-100 with $q = 0.1$ and 1 epoch for the remaining experiments.

For PiCO+, we basically follow the original PiCO method. The clean sample selection ratio parameter δ is set as 0.8/0.6 for noisy ratio 0.1/0.2, respectively. For neighbor augmentation, we set $k = 16$ for CIFAR-10 and a smaller $k = 5$ for CIFAR-100. In the beginning, the embeddings can be less meaningful and thus, we enable kNN augmentation after the first 100 epochs. We fix the shape parameter of the Beta distribution to $\varsigma = 4$ for mixup training. We set the loss weighting factor $\alpha = 2$ and $\beta = 0.1$. Similar to the standard PLL setup, we warm up the model by fitting uniform targets for 5 and 50 epochs on CIFAR-10 and CIFAR-100 datasets respectively.

B. Main Empirical Results on Standard PLL

1) Main Results: PiCO achieves SOTA results on standard PLL task: As shown in Table I, PiCO significantly outperforms all the rivals by a significant margin on all datasets. Specifically, on CIFAR-10 dataset, we improve upon the best baseline by

TABLE I
ACCURACY COMPARISONS ON STANDARD PLL DATASETS

Dataset	Method	$q = 0.1$	$q = 0.3$	$q = 0.5$
CIFAR-10	PiCO+ (ours)	**95.99** $\pm$ 0.03%	**95.73** $\pm$ 0.10%	**95.33** $\pm$ 0.06%
	PiCO (ours)	94.39 $\pm$ 0.18%	94.18 $\pm$ 0.12%	93.58 $\pm$ 0.06%
	LWS	90.30 $\pm$ 0.60%	88.99 $\pm$ 1.43%	86.16 $\pm$ 0.85%
	PRODEN	90.24 $\pm$ 0.32%	89.38 $\pm$ 0.31%	87.78 $\pm$ 0.07%
	CC	82.30 $\pm$ 0.21%	79.08 $\pm$ 0.07%	74.05 $\pm$ 0.35%
	MSE	79.97 $\pm$ 0.45%	75.64 $\pm$ 0.28%	67.09 $\pm$ 0.66%
	EXP	79.23 $\pm$ 0.10%	75.79 $\pm$ 0.21%	70.34 $\pm$ 1.32%
Dataset	Method	$q = 0.01$	$q = 0.05$	$q = 0.1$
CIFAR-100	PiCO+ (ours)	**76.29** $\pm$ 0.42%	**76.17** $\pm$ 0.18%	**75.55** $\pm$ 0.21%
	PiCO (ours)	73.09 $\pm$ 0.34%	72.74 $\pm$ 0.30%	69.91 $\pm$ 0.24%
	LWS	65.78 $\pm$ 0.02%	59.56 $\pm$ 0.33%	53.53 $\pm$ 0.08%
	PRODEN	62.60 $\pm$ 0.02%	60.73 $\pm$ 0.03%	56.80 $\pm$ 0.29%
	CC	49.76 $\pm$ 0.45%	47.62 $\pm$ 0.08%	35.72 $\pm$ 0.47%
	MSE	49.17 $\pm$ 0.05%	46.02 $\pm$ 1.82%	43.81 $\pm$ 0.49%
	EXP	44.45 $\pm$ 1.50%	41.05 $\pm$ 1.40%	29.27 $\pm$ 2.81%

Bold indicates superior results. On standard PLL datasets, PiCO+ can be regarded as a PiCO algorithm equipped with mixup training.

TABLE II
ABLATION STUDY OF PICO ON STANDARD PARTIAL LABEL LEARNING DATASETS CIFAR-10 ($q = 0.5$) AND CIFAR-100 ($q = 0.05$)

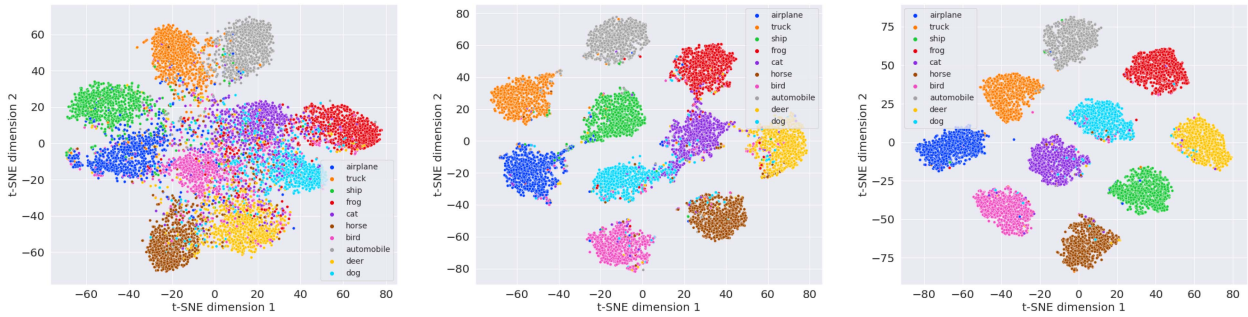
Ablation	$\mathcal{L}_{\text{cont}}$	Label Disambiguation	CIFAR-10	CIFAR-100
PiCO	✓	Ours	**93.58**	**72.74**
PiCO w/o Disambiguation	✓	Uniform Pseudo Target	84.50	64.11
PiCO w/o $\mathcal{L}_{\text{cont}}$	✗	Uniform Pseudo Target	76.46	56.87
PiCO with $\phi = 0$	✓	Soft Prototype Probs	91.60	71.07
PiCO with $\phi = 0$	✓	One-hot Prototype	91.41	70.10
PiCO	✓	MA Soft Prototype Probs	81.67	63.75
Fully-Supervised w/o mixup	✓	N/A	94.91	73.56

4.09%, **4.80%**, and **5.80%** where q is set to 0.1,0.3,0.5 respectively. Moreover, PiCO consistently achieves superior results as the size of the candidate set increases, while the baselines demonstrate a significant performance drop. Besides, it is worth pointing out that previous works [20], [21] are typically evaluated on datasets with a small label space ($C = 10$). We challenge this by showing additional results on CIFAR-100. When $q = 0.1$, all the baselines fail to obtain satisfactory performance, whereas PiCO remains competitive. Moreover, we observe that PiCO achieves results that are comparable to the *fully supervised contrastive learning model* (in Table II), showing that disambiguation is sufficiently accomplished. The comparison highlights the superiority of our label disambiguation strategy. Lastly, we evaluated the PiCO+ method in the standard PLL setup, which equals a PiCO model with mixup training. It can be shown that PiCO+ further improves PiCO by a notable margin, validating the importance of the mixup technique.

PiCO learns more distinguishable representations: We visualize the image representation produced by the feature encoder using t-SNE [26] in Fig. 4. Different colors represent different ground-truth class labels. We use the CIFAR-10 dataset with $q = 0.5$. We contrast the t-SNE embeddings of three approaches: (a) a model trained with uniform pseudo targets, i.e., $s_j = 1/|Y|$ ($j \in Y$), (b) the best baseline PRODEN, and (c) our method PiCO. We can observe that the representation of the uniform model is indistinguishable since its supervision signals suffer from high uncertainty. The features of PRODEN are improved, yet with some class overlapping (e.g., blue and purple). In contrast, PiCO produces well-separated clusters and more distinguishable representations, which validates its effectiveness in learning high-quality representation.

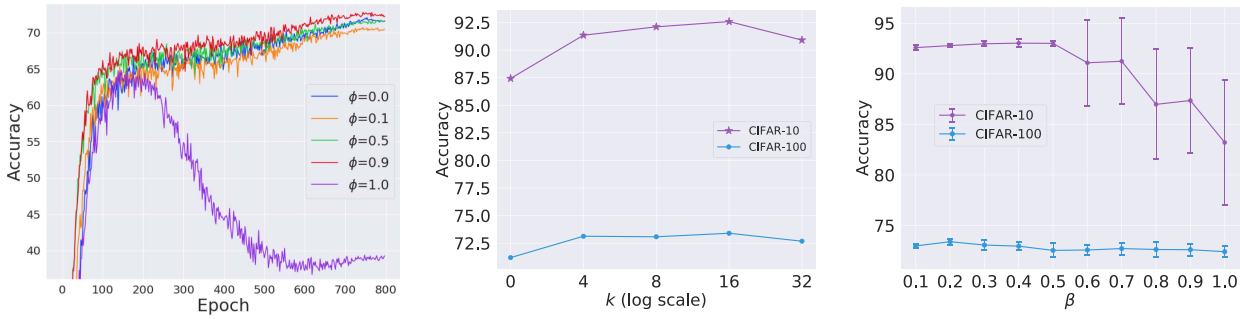
2) Ablation Studies of PiCO: Effect of $\mathcal{L}_{cont}$ and label disambiguation: We ablate the contributions of two key components of PiCO: contrastive learning and prototype-based label disambiguation. In particular, we compare PiCO with two variants: 1) *PiCO w/o disambiguation* which keeps the pseudo target as uniform $1/|Y|$; and 2) *PiCO w/o $\mathcal{L}_{cont}$* which further removes the contrastive learning and only trains a classifier with uniform pseudo targets. From Table II, we can observe that variant 1 substantially outperforms variant 2 (e.g., +8.04% on CIFAR-10), which signifies the importance of contrastive learning for producing better representations. Moreover, with label disambiguation, PiCO obtains results close to *fully supervised setting*, which verifies the ability of PiCO in identifying the ground-truth.

Different Disambiguation Strategy: Based on the contrastive prototypes, various strategies can be used to disambiguate the labels, which corresponds to the E-step in our theoretical analysis. We choose the following variants: 1) *One-hot Prototype* always



(a) Uniform features (b) PRODEN features (c) PiCO features (ours)

Fig. 4. T-SNE visualization of the PiCO representation on CIFAR-10 ($q = 0.5$). Different colors represent the corresponding classes.



(a) PiCO Ablation on ϕ (b) PiCO+ Ablation on k (c) PiCO+ Ablation on β

Fig. 5. (a) Performance of PiCO with varying moving-average factor ϕ on CIFAR-100 ($q = 0.05$). (b) Performance of PiCO+ with varying neighbor number k on CIFAR-10 ($q = 0.5, \eta = 0.2$) and CIFAR-100 ($q = 0.05, \eta = 0.2$). (c) Performance of PiCO+ with varying semi-supervised loss weighting factor β on CIFAR-10 ($q = 0.5, \eta = 0.2$).

assigns a one-hot pseudo target $\boldsymbol{s} = \boldsymbol{z}$ by using the nearest prototype ($\phi = 0$); 2) *Soft Prototype Probs* follows [27] and uses a soft class probability $s_i = \frac{\exp(\boldsymbol{q}^\top \boldsymbol{\mu}_i/\tau)}{\sum_{j\in Y}\exp(\boldsymbol{q}^\top \boldsymbol{\mu}_j/\tau)}$ as the pseudo target ($\phi = 0$); 3) *MA Soft Prototype Probs* gradually updates pseudo target from uniform by using the soft probabilities in a moving-average style. From Table II, we can see that directly using either soft or hard prototype-based label assignment leads to competitive results. This corroborates our theoretical analysis in Section V, since center-based class probability estimation is common in clustering algorithms. However, *MA Soft Prototype Probs* displays degenerated performance, suggesting soft label assignment is less reliable in identifying the ground-truth. Finally, PiCO outperforms the best variant by $\approx 2\%$ in accuracy on both datasets, showing the superiority of our label disambiguation strategy.

Effect of Moving-Average Factor ϕ: We then explore the effect of pseudo target updating factor ϕ on PiCO performance. Fig. 5(a) shows the learning curves of PiCO on CIFAR-100 ($q = 0.05$). We can see that the best result is achieved at $\phi = 0.9$ and the performance drops when ϕ takes a smaller value, particularly on the early stage. When $\phi = 0$, PiCO obtains a competitive result but is much lower than $\phi = 0.9$. This confirms that trusting the uniform pseudo targets at the early stage is crucial in obtaining superior performance. At the other extreme value $\phi = 1$, uniform pseudo targets are used, and PiCO demonstrates a degenerated performance and severe overfitting phenomena. In general, PiCO performs well when $\phi \approx 0.9$.

C. Main Empirical Results on Noisy PLL

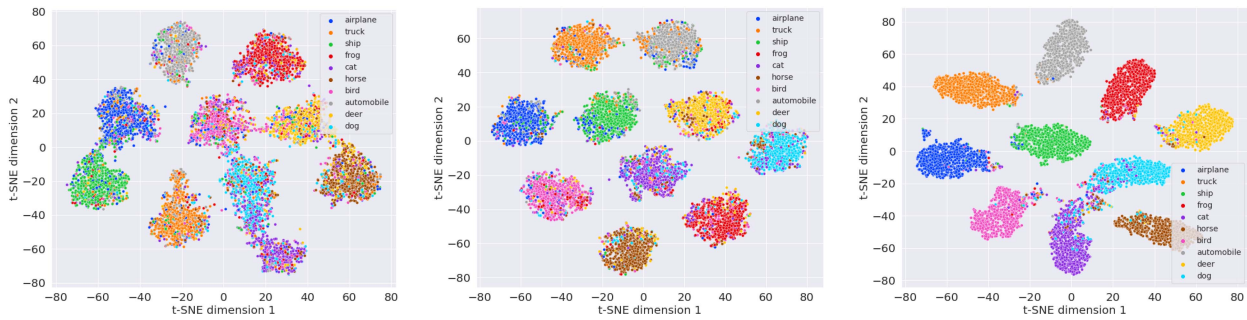
1) Main Results: PiCO+ achieves SOTA results on noisy PLL task: In Table III, we compare PiCO+ with competitive PLL methods on CIFAR datasets, where PiCO+ significantly outperforms baselines. In specific, on CIFAR-10 dataset with $q = 0.5$, PiCO+ improves upon the best competitor by $\mathbf{6.66}\%$ and $\mathbf{12.52}\%$ when η is set to 0.1,0.2 respectively. Notably, under the noisy PLL setup, even when only 10% examples have wrong candidate sets, the baseline algorithms (including PiCO) exhibit severe performance degradation. This is further aggravated on CIFAR-100 with a larger label space, while PiCO+ consistently retains its great robustness.

PiCO+ learns compact and distinguishable features: In Fig. 6, we visualize the feature representations of PiCO+ on the noisy PLL CIFAR-10 dataset with $q = 0.5, \eta = 0.2$. It can be shown that PiCO generates compact representations even with noisy candidate sets, which further supports the clustering effect of contrastive learning. However, both PiCO and PRODEN exhibit severe overfitting on wrong labels. Instead, the features of our PiCO+ are both compact and distinguishable.

TABLE III
ACCURACY COMPARISONS ON NOISY PLL DATASETS

Dataset	Method	$q = 0.3$		$q = 0.5$	
		$\eta = 0.1$	$\eta = 0.2$	$\eta = 0.1$	$\eta = 0.2$
CIFAR-10	PiCO+ (ours)	**95.11** $\pm$ 0.13%	**93.98** $\pm$ 0.39%	**94.45** $\pm$ 0.27%	**92.59** $\pm$ 0.22%
	PiCO (ours)	89.47 $\pm$ 0.37%	84.13 $\pm$ 0.53%	87.79 $\pm$ 0.09%	80.07 $\pm$ 0.60%
	LWS	84.51 $\pm$ 0.13%	77.98 $\pm$ 0.09%	71.02 $\pm$ 7.27%	61.96 $\pm$ 3.22%
	PRODEN	84.56 $\pm$ 0.16%	79.35 $\pm$ 0.12%	81.97 $\pm$ 0.59%	77.15 $\pm$ 0.08%
	CC	72.16 $\pm$ 0.93%	68.42 $\pm$ 0.37%	65.61 $\pm$ 0.43%	51.82 $\pm$ 4.13%
	MSE	53.77 $\pm$ 1.44%	49.73 $\pm$ 2.94%	46.56 $\pm$ 0.18%	39.80 $\pm$ 2.82%
	EXP	75.81 $\pm$ 0.09%	69.97 $\pm$ 0.39%	64.26 $\pm$ 1.02%	54.93 $\pm$ 1.11%
	GCE	74.32 $\pm$ 1.04%	69.90 $\pm$ 0.80%	70.38 $\pm$ 7.63%	50.57 $\pm$ 0.95%
Dataset	Method	$q = 0.05$		$q = 0.1$	
		$\eta = 0.1$	$\eta = 0.2$	$\eta = 0.1$	$\eta = 0.2$
CIFAR-100	PiCO+ (ours)	**74.68** $\pm$ 0.15%	**72.98** $\pm$ 0.22%	**67.58** $\pm$ 1.05%	**62.24** $\pm$ 0.97%
	PiCO (ours)	66.29 $\pm$ 0.10%	59.81 $\pm$ 0.24%	66.15 $\pm$ 0.03%	45.32 $\pm$ 0.89%
	LWS	52.20 $\pm$ 1.47%	42.31 $\pm$ 1.05%	20.54 $\pm$ 4.77%	17.76 $\pm$ 4.47%
	PRODEN	53.40 $\pm$ 0.61%	46.11 $\pm$ 0.38%	47.34 $\pm$ 1.39%	38.03 $\pm$ 1.79%
	CC	42.06 $\pm$ 0.67%	37.90 $\pm$ 3.27%	32.11 $\pm$ 3.95%	22.28 $\pm$ 6.18%
	MSE	31.06 $\pm$ 2.46%	27.36 $\pm$ 0.40%	25.86 $\pm$ 1.87%	22.98 $\pm$ 1.74%
	EXP	23.98 $\pm$ 4.25%	22.37 $\pm$ 5.45%	23.78 $\pm$ 4.59%	22.27 $\pm$ 3.38%
	GCE	35.85 $\pm$ 1.37%	31.65 $\pm$ 0.71%	27.79 $\pm$ 2.80%	24.21 $\pm$ 1.67%

Bold indicates superior results.



(a) PRODEN features (b) PiCO features (c) PiCO+ features (ours)

Fig. 6. T-SNE visualization of the PiCO+ representation on noisy PLL version of CIFAR-10 ($q = 0.5, \eta = 0.2$). Different colors represent the corresponding classes.

2) Ablation Studies of PiCO+: Effect of sample selection: We first study the effectiveness of our distance-based selection mechanism by comparing PiCO+ with two variants: 1) *PiCO+ with SL-Sel* selects clean examples by sorting cross-entropy losses; 2) *PiCO+ with Only Clean* employs the clean samples to run a simple PiCO method. As reported in Table IV, PiCO+ with only clean data exhibits better performance than the vanilla PiCO method (e.g., +7.56%) on CIFAR-10, indicates the selected examples enjoy high purity. Moreover, PiCO+ with SL-Sel underperforms PiCO+, which verifies that the loss values are indeed less informative in the presence of candidate labels and that our strategy is a better alternative.

Effect of semi-supervised contrastive training: Next, we explore the effect of each component in SSL training. We compare PiCO+ with three variants: 1) *PiCO+ w/o* $\mathcal{L}_{n\text{-}cls}$ removes the label guessing technique; 2) *PiCO+ w/o* $\mathcal{L}_{n\text{-}cont}$ removes the label-driven contrastive loss; 3) *PiCO+ w/o kNN* disables the neighbor-augmented contrastive loss; 4) *PiCO+ w/o Mixup* disables the mixup training. From Table IV, we can see that all the components of our SSL framework contribute to the performance improvements. In particular, $\mathcal{L}_{\text{n-cont}}$ has a stronger positive effect on CIFAR-10, and the label guessing component brings extra performance improvements for both datasets. The mixup and neighbor augmentation are the most crucial to the final performance. We note that the mixup training technique also improves the fully-supervised model (e.g., +1.05% on CIFAR-10). But the performance improvement is more substantial on the noisy PLL tasks (e.g., +2.92% on CIFAR-10), indicating its

TABLE IV
ABLATION STUDY OF PICO+ ON NOISY PARTIAL LABEL LEARNING DATASETS CIFAR-10 ($q = 0.5, \eta = 0.2$) AND CIFAR-100 ($q = 0.05, \eta = 0.2$)

Ablation	$\mathcal{L}_{\text{n-cls}}$	$\mathcal{L}_{\text{n-cont}}$	kNN	Mixup	CIFAR-10	CIFAR-100
PiCO+	✓	✓	✓	All Data	**92.59**	**72.98**
PiCO+ with SL-Sel	✓	✓	✓	All Data	91.22	72.54
PiCO+ with Only Clean	✗	✗	✗	Only Clean	87.63	70.72
PiCO+ w/o $\mathcal{L}_{\text{n-cls}}$	✗	✓	✓	All Data	91.14	71.98
PiCO+ w/o $\mathcal{L}_{\text{n-cont}}$	✓	✗	✓	All Data	90.87	72.89
PiCO+ w/o kNN	✓	✓	✗	All Data	87.43	71.19
PiCO+ w/o Mixup	✓	✓	✓	No Mixup	89.67	68.51
Fully-Supervised w/ mixup†	-	-	-	All Data	95.96♮	76.36

† These supervised results are evaluated with mixup like PiCO+ and thus are different from Table 2.
♮ It is slightly smaller than PiCO+ in Table 1 ($q = 0.1$) because of randomized running, but they have no statistically significant difference.

TABLE V
ACCURACY COMPARISONS ON NOISY PLL DATASETS WITH MORE NOISY SAMPLES

Method	CIFAR-10 ($q = 0.5$)		CIFAR-100 ($q = 0.05$)		
	$\eta = 0.3$	$\eta = 0.4$	$\eta = 0.3$	$\eta = 0.4$	$\eta = 0.5$
PiCO+ (ours)	**90.12** $\pm$ 0.51%	**76.09** $\pm$ 3.62%	**70.46** $\pm$ 0.51%	**66.41** $\pm$ 0.58%	**60.50** $\pm$ 0.99%
PiCO (ours)	64.79 $\pm$ 2.08%	34.59 $\pm$ 7.26%	52.18 $\pm$ 0.52%	44.17 $\pm$ 0.08%	35.51 $\pm$ 1.14%
PRODEN	50.32 $\pm$ 1.07%	29.68 $\pm$ 13.29%	39.19 $\pm$ 0.20%	33.64 $\pm$ 0.82%	26.91 $\pm$ 0.83%

Bold indicates superior results.

robustness on noisy data. Lastly, Fig. 5(b) shows the influence of neighbor number k, where PiCO+ works well in a wide range of k values. Nevertheless, a too large k may collect many noisy positive peers and slightly drops the performance.

Effect of loss weighting factor β: Fig. 5 reports the performance of PiCO+ with varying β values. On the CIFAR-10 dataset, we observe a severe performance degradation with β being larger. The variance becomes increasingly larger as well. Similar trends can also be observed on CIFAR-100, though the results are much stabler. It suggests that the usage of noisy examples should be careful as they may result in confirmation bias.

3) The Robustness of PiCO+ With Severe Noise: Finally, we conduct experiments on noisy PLL datasets that contain much more severe noise to show the robustness of our PiCO+ method. In particular, we choose $\eta \in \{0.3, 0.4\}$ and $\eta \in \{0.3, 0.4, 0.5\}$ for CIFAR-10 and CIFAR-100 respectively. Accordingly, we adjust the selection ratio to $\delta = 0.5, 0.4$ when $\eta = 0.4, 0.5$, without changing other setups. Table V compares PiCO+ with the two most competitive baselines PiCO and PRODEN, where PiCO+ obtains very impressive performance. For example, the gaps between PiCO+ and the best baseline are $\mathbf{41.50}\%$ and $\mathbf{24.99}\%$ on CIFAR-10 with $\eta = 0.4$ and CIFAR-100 with $\eta = 0.5$. We conclude that PiCO+ is indeed much more robust than existing PLL algorithms.

D. Experiments on More Challenging Datasets

1) Fine-Grained Partial Label Learning: Recall the dog example highlighted in Section I, where semantically similar classes are more likely to cause label ambiguity. It begs the question of whether PiCO is effective in challenging fine-grained classification tasks. To verify this, we conduct experiments on three datasets: 1) CUB-200 dataset [28] contains 200 bird species; 2) Flowers-102 [29] collects 102 flowers that commonly occurs in the U.K.; 3) CIFAR-100 with hierarchical labels (CIFAR-100-H), where we generate candidate labels that belong to the same superclass.[1] Notably, both CUB-200/Flowers-102 datasets contain a large amount of similar-looking dog/flower samples, and can be quite challenging to disambiguate the candidate labels. We set $q = 0.05$ for CUB-200/Flowers-102 and $q = 0.5$ for CIFAR-100-H. In Table VI, we compare PiCO with baselines, where PiCO outperforms the best method PRODEN by a large margin (**+9.61%** on CUB-200 and **+11.15%** on CIFAR-100-H). In addition, we test the performance of PiCO+ on both standard and noisy PLL versions of fine-grained datasets. For the noisy version, we set the number of neighbors

TABLE VI
ACCURACY COMPARISONS ON FINE-GRAINED CLASSIFICATION DATASETS WITH STANDARD PLL LABELS

Method	CUB-200 ($q = 0.05$)	CIFAR-100-H ($q = 0.5$)	Flowers-102 ($q = 0.05$)
PiCO+	72.05 $\pm$ 0.80%	**75.38** $\pm$ 0.52%	**91.31** $\pm$ 0.10%
PiCO	**72.17** $\pm$ 0.72%	72.04 $\pm$ 0.31%	91.01 $\pm$ 0.17%
LWS	39.74 $\pm$ 0.47%	57.25 $\pm$ 0.02%	48.23 $\pm$ 0.22%
PRODEN	62.56 $\pm$ 0.10%	60.89 $\pm$ 0.03%	88.76 $\pm$ 0.06%
CC	55.61 $\pm$ 0.02%	42.60 $\pm$ 0.11%	57.59 $\pm$ 0.05%
MSE	22.07 $\pm$ 2.36%	39.52 $\pm$ 0.28%	2.70 $\pm$ 1.79%
EXP	9.44 $\pm$ 2.32%	35.08 $\pm$ 1.71%	3.87 $\pm$ 16.31%

[1]CIFAR-100 dataset consists of 20 superclasses, with 5 classes in each superclass.

TABLE VII
ACCURACY COMPARISONS ON FINE-GRAINED CLASSIFICATION DATASETS WITH NOISY PLL LABELS

Method	CUB-200 ($q = 0.05$)	CIFAR-100-H ($q = 0.5$)	Flowers-102 ($q = 0.05$)
PiCO+	**60.65** ± 0.79%	**68.31** ± 0.47%	**84.20** ± 0.18%
PiCO	53.05 ± 2.03%	59.81 ± 0.25%	80.68 ± 0.33%
LWS	18.65 ± 2.15%	22.18 ± 6.12%	35.09 ± 0.36%
PRODEN	44.74 ± 2.47%	48.03 ± 0.47%	65.13 ± 0.09%
CC	26.98 ± 1.16%	34.57 ± 0.99%	41.05 ± 0.12%
MSE	20.92 ± 1.20%	35.20 ± 1.03%	2.67 ± 2.03%
EXP	2.81 ± 14.46%	20.80 ± 4.62%	2.83 ± 22.50%
GCE	5.13 ± 38.65%	33.21 ± 2.03%	0.75 ± 43.11%

The noisy rate is set as η= 0:2.

by $k = 3$ for CUB-200/Flowers-102 and $\eta = 0.2$ for all. The results are listed in Tables VI and VII, where PiCO+ achieves substantially better performance than all the baselines, e.g., improves the best PRODEN algorithm by **+19.07%** accuracy on Flowers-102. Our results validate the effectiveness and robustness of our PiCO+ framework, even in the presence of strong label ambiguity.

2) Experiments on ImageNet: Lastly, we verify the scalability of PiCO+ on large-scale ImageNet datasets. Due to limited computation resources, we follow previous weakly-supervised learning literature [30], [31] and take a sub-sampled version that contains 100 classes. The resulting dataset is still large-scale that contains 128,545 image samples. During running, we set the image resolution as 224×224. The ambiguity degree is set as $q = 0.1$. For the noisy PLL task, we set the noisy rate as $\eta = 0.2$. It can be shown in Table IX that PiCO+ outperforms the baselines by a substantial margin, e.g., improves the best CC algorithm by **+2.78%** on the clean PLL and improves the best GCE method by **+3.74%** on the noisy PLL setups. The results further validate the scalability of PiCO+ on large-scale datasets.

E. Empirical Results on PLL With OOD Noise

Lastly, we investigate the robustness of PiCO+ in the presence of OOD noise. Following [32], we use a series of auxiliary datasets SVHN [33], Places-365 [34] and Texture [35]. We randomly select a subset of 10,000 samples for large-scale SVHN and Places365 datasets. These datasets are combined with CIFAR-10 and CIFAR-100 to synthesize the PLL datasets that contain both in-distribution (ID) noise and OOD noise. We set $q = 0.5$ for CIFAR-10 and $q = 0.05$ for CIFAR-100 to generate candidate labels on all samples and use $\eta = 0.2$ to randomly flip ID data to noisy candidates. Notably, all OOD samples are noisy since their ground-truth labels are not included in the candidate sets. The experimental results are shown in Table VIII. We can observe that even without energy-base rejection, PiCO+ demonstrates great robustness compared with PiCO and PRODEN algorithms. The main reason is that our sample selection procedure dominates the training procedure. Equipped with the rejection mechanism, PiCO+ is able to achieve further improved results e.g., **+35.73%** accuracy compared with PRODEN on CIFAR-10 with SVHN as OOD dataset. These findings clearly validate the robustness of PiCO+ in open-world scenarios.

V. WHY PICO IMPROVES PARTIAL LABEL LEARNING?

In this section, we provide theoretical justification on why the contrastive prototypes help disambiguate the ground-truth label. We show that the *alignment* property in contrastive learning [11] intrinsically minimizes the intraclass covariance in the embedding space, which coincides with the objective of classical clustering algorithms. It motivates us to interpret PiCO through the lens of the expectation-maximization algorithm. To see this, we consider an ideal setting: in each training step, all data examples are accessible and the augmentation copies are also included in the training set, i.e., $A = \mathcal{D}$. Then, the contrastive loss is calculated as,

$$\begin{aligned}
&\tilde{\mathcal{L}}_{\text{cont}}(g;\tau,\mathcal{D}) \\
&= \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{D}}\left\{-\frac{1}{|P(\boldsymbol{x})|}\sum_{\boldsymbol{k}_+\in P(\boldsymbol{x})}\log\frac{\exp(\boldsymbol{q}^\top\boldsymbol{k}_+/\tau)}{\sum_{\boldsymbol{k}'\in A(\boldsymbol{x})}\exp(\boldsymbol{q}^\top\boldsymbol{k}'/\tau)}\right\} \\
&= \underbrace{\frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{D}}\left\{-\frac{1}{|P(\boldsymbol{x})|}\sum_{\boldsymbol{k}_+\in P(\boldsymbol{x})}(\boldsymbol{q}^\top\boldsymbol{k}_+/\tau)\right\}}_{\text{(a)}} \\
&\quad + \underbrace{\frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{D}}\left\{\log\sum_{\boldsymbol{k}'\in A(\boldsymbol{x})}\exp(\boldsymbol{q}^\top\boldsymbol{k}'/\tau)\right\}}_{\text{(b)}}. \qquad (15)
\end{aligned}$$

We focus on analyzing the first term (a), which is often dubbed as the *alignment* term [11]. The main functionality of this term is to optimize the tightness of the clusters in the embedding space. In this work, we connect it with classical clustering algorithms. We first split the dataset to C subsets $S_j \in \mathcal{D}_C$ $(1 \le j \le C)$, where each subset contains examples possessing the same predicted labels. In effect, our selection strategy in (4) constructs the positive set by selecting examples from the same subset. Therefore, we have,

$$\begin{aligned}
\text{(a)} &= \frac{1}{n}\sum_{\boldsymbol{x}\in\mathcal{D}}\frac{1}{|P(\boldsymbol{x})|}\sum_{\boldsymbol{k}_+\in P(\boldsymbol{x})}(||\boldsymbol{q}-\boldsymbol{k}_+||^2-2)/(2\tau) \\
&\approx \frac{1}{2\tau n}\sum_{S_j\in\mathcal{D}_C}\frac{1}{|S_j|}\sum_{\boldsymbol{x},\boldsymbol{x}'\in S_j}||g(\boldsymbol{x})-g(\boldsymbol{x}')||^2+K \\
&= \frac{1}{\tau n}\sum_{S_j\in\mathcal{D}_C}\sum_{\boldsymbol{x}\in S_j}||g(\boldsymbol{x})-\boldsymbol{\mu}_j||^2+K, \qquad (16)
\end{aligned}$$

where K is a constant and $\boldsymbol{\mu}_j$ is the mean center of S_j. Here we approximate $\frac{1}{|S_j|} \approx \frac{1}{|S_j|-1} = \frac{1}{|P(\boldsymbol{x})|}$ since n is usually large. We omitted the augmentation operation for simplicity. The *uniformity* term (b) can benefit information-preserving, and has been analyzed in [11].

TABLE VIII
ACCURACY COMPARISONS ON PLL DATASETS WITH BOTH IN-DISTRIBUTION NOISE ($\eta = 0.2$) AND OUT-OF-DISTRIBUTION NOISE

Base Dataset	CIFAR-10 ($q = 0.5$)			CIFAR-100 ($q = 0.05$)		
OOD Dataset	SVHN	Texture	Places-365	SVHN	Texture	Places-365
PiCO+ with Rejection (ours)	**75.89**	**76.23**	**66.45**	**62.36**	**62.55**	**56.89**
PiCO+ (ours)	66.39	76.01	66.07	59.51	61.11	56.22
PiCO	54.66	75.99	57.73	57.43	58.98	56.04
PRODEN	40.16	40.09	41.23	39.03	38.71	38.17

Bold indicates superior results.

TABLE IX
ACCURACY COMPARISONS ON IMAGENET DATASET WITH 128,545 IMAGES AND 100 SUB-SAMPLED CLASSES

Method	ImageNet ($q = 0.1$)	
	$\eta = 0$	$\eta = 0.2$
PiCO+	**82.95** $\pm$ 0.39%	**76.83** $\pm$ 0.34%
PiCO	81.16 $\pm$ 0.73%	71.93 $\pm$ 0.32%
PRODEN	79.90 $\pm$ 0.81%	69.49 $\pm$ 0.28%
CC	80.17 $\pm$ 0.14%	64.89 $\pm$ 0.34%
GCE	79.15 $\pm$ 0.21%	73.09 $\pm$ 0.41%

We are now ready to interpret the PiCO algorithm as an expectation-maximization algorithm that maximizes the likelihood of a generative model. At the E-step, the classifier assigns each data example to one specific cluster. At the M-step, the contrastive loss concentrates the embeddings to their cluster mean direction, which is achieved by minimizing (16). Finally, the training data will be mapped to a mixture of von Mises-Fisher distributions on the unit hypersphere.

EM Perspective: Recall that the candidate label set is a noisy version of the ground-truth. To estimate the likelihood $P(Y_i, \boldsymbol{x}_i)$, we need to establish the relationship between the candidate and the ground-truth label. Following [6], we make a mild assumption,

Assumption 1: All labels y_i in the candidate label set have the same probability of generating Y_i, but no label outside of Y_i can generate Y_i, i.e. $P(Y_i|y_i) = \hbar(Y_i)$ if $y_i \in Y_i$ else 0. Here $\hbar(\cdot)$ is some function making it a valid probability distribution.

Then, we show that the PiCO implicitly maximizes the likelihood as follows,

E-Step: First, we introduce some distributions over all examples and the candidates $\pi_i^j \geq 0\,(1 \leq i \leq n, 1 \leq j \leq C)$ such that $\pi_i^j = 0$ if $j \notin Y_i$ and $\sum_{j \in Y_i} \pi_i^j = 1$. Let θ be the parameters of g. Our goal is to maximize the likelihood below,

$$\begin{aligned}
&\arg\max_{\theta} \sum_{i=1}^{n} \log P(Y_i, \boldsymbol{x}_i|\theta) \\
&= \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log \sum_{y_i \in Y_i} P(\boldsymbol{x}_i, y_i|\theta) + \sum_{i=1}^{n} \log(\hbar(Y_i)) \\
&= \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log \sum_{y_i \in Y_i} \pi_i^{y_i} \frac{P(\boldsymbol{x}_i, y_i|\theta)}{\pi_i^{y_i}} \\
&\geq \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \sum_{y_i \in Y_i} \pi_i^{y_i} \log \frac{P(\boldsymbol{x}_i, y_i|\theta)}{\pi_i^{y_i}}. \qquad (17)
\end{aligned}$$

The last step of the derivation uses Jensen's inequality. By using the fact that $\log(\cdot)$ function is concave, the inequality holds with equality when $\frac{P(\boldsymbol{x}_i, y_i|\theta)}{\pi_i^{y_i}}$ is some constant. Therefore,

$$\pi_i^{y_i} = \frac{P(\boldsymbol{x}_i, y_i|\theta)}{\sum_{y_i \in Y_i} P(\boldsymbol{x}_i, y_i|\theta)} = \frac{P(\boldsymbol{x}_i, y_i|\theta)}{P(\boldsymbol{x}_i|\theta)} = P(y_i|\boldsymbol{x}_i, \theta), \qquad (18)$$

which is the posterior class probability. In PiCO, it is estimated by using the classifier's output.

To estimate $P(y_i|\boldsymbol{x}_i, \theta)$, classical unsupervised clustering methods intuitively assign the data examples to the cluster centers, e.g., k-means. As in the supervised learning setting, we can directly use the ground-truth. However, under the setting of PLL, the supervision signals are situated between the supervised and unsupervised setups. Based on empirical findings, the candidate labels are more reliable for posterior estimation at the beginning; yet alongside the training process, the prototypes tend to become more trustful. This empirical observation has motivated us to update the pseudo targets in a moving-average style. Thereby, we have a good initialization in estimating class posterior, and it will be smoothly refined during the training procedure. This is verified in our empirical studies; see Section IV-B2 and Appendix B.1.4, available online. Finally, we take one-hot prediction $\tilde{y}_i = \arg\max_{j \in Y} f^j(\boldsymbol{x}_i)$ since each example inherently belongs to exactly one label and hence, we have $\pi_i^j = \mathbb{I}(\tilde{y}_i = j)$.

M-Step: At this step, we aim at maximizing the likelihood under the assumption that the posterior class probability is known. We show that under mild assumptions, minimizing (16) also maximizes a lower bound of likelihood in (17).

Theorem 1: Assume data from the same class in the contrastive output space follow a d-variate von Mises-Fisher (vMF) distribution whose probabilistic density is given by $f(\boldsymbol{x}|\bar{\boldsymbol{\mu}}_i, \kappa) = c_d(\kappa) e^{\kappa \bar{\boldsymbol{\mu}}_i^\top g(\boldsymbol{x})}$, where $\bar{\boldsymbol{\mu}}_i = \boldsymbol{\mu}_i / ||\boldsymbol{\mu}_i||$ is the mean direction, κ is the concentration parameter, and $c_d(\kappa)$ is the normalization factor. We further assume a uniform class prior $P(y_i = j) = 1/C$. Let $n_j = |S_j|$. Then, optimizing (16) and (17) equal to maximize R_1 and R_2 below, respectively.

$$R_1 = \sum_{S_j \in \mathcal{D}_C} \frac{n_j}{n} ||\boldsymbol{\mu}_j||^2 \leq \sum_{S_j \in \mathcal{D}_C} \frac{n_j}{n} ||\boldsymbol{\mu}_j|| = R_2. \qquad (19)$$

The proof can be found in the Appendix A, available online. Theorem 1 indicates that minimizing (16) also maximizes a lower bound of the likelihood in (17). The lower bound is

tight when $||\boldsymbol{\mu}_j||$ is close to 1, which in effect means a strong intraclass concentration on the hypersphere. Intuitively, when the hypothesis space is rich enough, it is possible to achieve a low intraclass covariance in the euclidean space, resulting in a large norm of the mean vector $||\boldsymbol{\mu}_j||$. Then, normalized embeddings in the hypersphere also have an intraclass concentration in a strong sense, because a large $||\boldsymbol{\mu}_j||$ also results in a large κ [36]. Regarding the visualized representation in Fig. 4, we note that PiCO is indeed able to learn compact clusters. Therefore, we have that minimizing the contrastive loss also partially maximizes the likelihood defined in (17).

The Robustness of PiCO+: Note that the above theoretical results can also help analyze PiCO+. On one hand, the clustering effect of PiCO encourages samples to be aligned to prototypes. Since deep networks tend to find easy patterns in the early stage of training [37], those clean samples can be moved faster to the class centers, which is empirically verified in Appendix B.9, available online. On the other hand, after sample selection, our semi-supervised contrastive learning procedure can also be (approximately) regarded as a partial label learning problem if we regard the noisy data are related to a candidate set of full labels. Hence, PiCO+ also runs an EM algorithm that trains robust classifiers with the help of the clustering effect of contrastive learning.

VI. Related Works

Partial Label Learning: (PLL) allows each training example to be annotated with a candidate label set, in which the ground-truth is guaranteed to be included. The most intuitive solution is average-based methods [3], [4], [38], which treat all candidates equally. However, the key and obvious drawback is that the predictions can be severely misled by false positive labels. To disambiguate the ground-truth from the candidates, identification-based methods [39], which regard the ground-truth as a latent variable, have recently attracted increasing attention; representative approaches include maximum margin-based methods [40], [41], graph-based methods [7], [8], [42], [43], and clustering-based approaches [6]. Recently, self-training methods [20], [21], [22] have achieved state-of-the-art results on various benchmark datasets, which disambiguate the candidate label sets by means of the model outputs themselves. But, few efforts have been made to learn high-quality representations to reciprocate label disambiguation.

Contrastive Learning: (CL) [13], [44] is a framework that learns discriminative representations through the use of instance similarity/dissimilarity. A plethora of works has explored the effectiveness of contrastive learning in unsupervised representation learning [13], [44], [45]. Recently, [9] propose supervised contrastive learning (SCL), an approach that aggregates data from the same class as the positive set and obtains improved performance on various supervised learning tasks. The success of SCL has motivated a series of works to apply contrastive learning to a number of weakly supervised learning tasks, including noisy label learning [27], [46], semi-supervised learning [47], [48], etc. Despite promising empirical results, however, these works, lack theoretical understanding. [11] theoretically show that the contrastive learning favors *alignment* and *uniformity*, and thoroughly analyzed the properties of uniformity. But, to date, the terminology *alignment* remains confusing; we show it inherently maps data points to a mixture of vMF distributions.

Noisy Label Learning (NLL) [49] aims at mitigating overfitting on mislabeled samples. One popular strategy is to design robust risk functions, including but does not limit to robust cross-entropy losses [24], [50], [51], sample re-weighting [52], [53], [54] and noise transition matrix-based loss correction [55], [56], [57]. Another active line of research relies on selecting clean samples from noisy ones [58]. Most of them adopt the small-loss selection criterion [14], [59] which is motivated by the fact that deep neural networks tend to memorize easy patterns first [37]. Based on that, the state-of-the-art NLL algorithms [16], [27], [60] regard the unchosen samples as unlabeled and incorporate semi-supervised learning (SSL) for boosted performance. Inspired by these works, PiCO+ incorporates a new distance-based selection criterion and extends the contrastive learning framework to facilitate SSL training. The most related one to our work is [5] which theoretically analyzes the robustness of average-based loss functions for the noisy PLL task. But, [5] does not provide a new empirically strong solution. Instead, our PiCO+ framework establishes promising results against label noise that makes the PLL problem more practical for open-world applications.

Out-of-Distribution Detection: aims at detecting input samples with different characteristics and labels from the training data [12], [32], [61]. A plethora of works has been designed to separate the OOD samples with the pre-trained network, including calibrated output probability [62], maximum softmax probability [63], energy function [18], Mahalanobis distance [64], etc. The most related method to our work is the energy score function [18] which is computed via LogSumExp and is the soft maximum of logits. In our work, attempt to investigate the challenges of out-of-distribution noise in the field of PLL.

VII. Conclusion

In this work, we propose a novel noisy partial label learning framework PiCO+, which is mainly driven by a PiCO algorithm. The key idea is to identify the ground-truth label from the candidate set by using contrastively learned embedding prototypes. Additionally, our PiCO+ extends PiCO by sample selection and semi-supervised training, making it able to learn robust classifiers from noisy partial labels. Theoretical analysis shows that PiCO can be interpreted from an EM algorithm perspective. Empirically, we conducted extensive experiments and show that PiCO and PiCO+ establish state-of-the-art performance and demonstrate robustness to both in-distribution noise and out-of-distribution noise. Our results are competitive with the fully supervised setting, where the ground-truth label is given explicitly. Applications of multi-class classification with ambiguous labeling can benefit from our method, and we anticipate further research in PLL to extend this framework to tasks beyond image classification. We hope our work will draw more attention from the community toward a broader view of using contrastive prototypes for partial label learning.

REFERENCES

[1] J. Luo and F. Orabona, "Learning from candidate labeling sets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 1504–1512.

[2] C. Chen, V. M. Patel, and R. Chellappa, "Learning from ambiguously labeled face images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 7, pp. 1653–1667, Jul. 2018.

[3] E. Hüllermeier and J. Beringer, "Learning from ambiguously labeled examples," *Intell. Data Anal.*, vol. 10, no. 5, pp. 419–439, 2006.

[4] T. Cour, B. Sapp, and B. Taskar, "Learning from partial labels," *J. Mach. Learn. Res.*, vol. 12, pp. 1501–1536, 2011.

[5] J. Lv et al., "On the robustness of average losses for partial-label learning," 2021, *arXiv:2106.06152*.

[6] L. Liu and T. G. Dietterich, "A conditional multinomial mixture model for superset label learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 557–565.

[7] M. Zhang, B. Zhou, and X. Liu, "Partial label learning via feature-aware disambiguation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1335–1344.

[8] G. Lyu, S. Feng, T. Wang, C. Lang, and Y. Li, "GM-PLL: Graph matching based partial label learning," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 2, pp. 521–535, Feb. 2021.

[9] P. Khosla et al., "Supervised contrastive learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 18661–18673.

[10] H. Wang et al., "PICO: Contrastive label disambiguation for partial label learning," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–18.

[11] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 9929–9939.

[12] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," 2021, *arXiv:2110.11334*.

[13] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9726–9735.

[14] B. Han et al., "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8536–8546.

[15] D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5050–5060.

[16] J. Li, R. Socher, and S. C. H. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–14.

[17] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2020, pp. 1–8.

[18] W. Liu, X. Wang, J. D. Owens, and Y. Li, "Energy-based out-of-distribution detection," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 21464–21475.

[19] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[20] J. Lv, M. Xu, L. Feng, G. Niu, X. Geng, and M. Sugiyama, "Progressive identification of true labels for partial-label learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6500–6510.

[21] H. Wen, J. Cui, H. Hang, J. Liu, Y. Wang, and Z. Lin, "Leveraged weighted loss for partial label learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11091–11100.

[22] L. Feng et al., "Provably consistent partial-label learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 10948–10960.

[23] L. Feng, T. Kaneko, B. Han, G. Niu, B. An, and M. Sugiyama, "Learning with multiple complementary labels," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3072–3081.

[24] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8792–8802.

[25] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "RandAugment: Practical data augmentation with no separate search," 2019, *arXiv: 1909.13719*.

[26] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.

[27] J. Li, C. Xiong, and S. C. H. Hoi, "MoPro: Webly supervised learning with momentum prototypes," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–15.

[28] P. Welinder et al., "Caltech-UCSD birds 200," California Institute of Technology, Tech. Rep. CNS-TR-2010-001, 2010.

[29] M. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. IEEE 6th Indian Conf. Comput. Vis. Graph. Image Process.*, 2008, pp. 722–729.

[30] Z. Gao et al., "Learning from multiple annotator noisy labels via sample-wise label fusion," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 407–422.

[31] K. Cao, M. Brbic, and J. Leskovec, "Open-world semi-supervised learning," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–15.

[32] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20827–20840.

[33] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011. [Online]. Available: http://ufldl.stanford.edu/%20housenumbers/nips2011_housenumbers.pdf

[34] B. Zhou, À. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2018.

[35] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3606–3613.

[36] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra, "Clustering on the unit hypersphere using von Mises-Fisher distributions," *J. Mach. Learn. Res.*, vol. 6, pp. 1345–1382, 2005.

[37] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–15.

[38] M. Zhang and F. Yu, "Solving the partial label learning problem: An instance-based approach," in *Proc. 24th Int. Conf. Artif. Intell.*, 2015, pp. 4048–4054.

[39] R. Jin and Z. Ghahramani, "Learning with multiple labels," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 897–904.

[40] N. Nguyen and R. Caruana, "Classification with partial labels," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2008, pp. 551–559.

[41] H. Wang et al., "Online partial label learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 455–470.

[42] D. Wang, L. Li, and M. Zhang, "Adaptive graph guided disambiguation for partial label learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 83–91.

[43] N. Xu, J. Lv, and X. Geng, "Partial label learning via label enhancement," in *Proc. Conf. Assoc. Adv. Artif. Intell.*, 2019, pp. 5557–5564.

[44] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding,", 2018, *arXiv: 1807.03748*.

[45] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.

[46] Z. Wu, T. Wei, J. Jiang, C. Mao, M. Tang, and Y. Li, "NGC: A unified framework for learning with open-world noisy data," 2021, *arXiv:2108.11035*.

[47] J. Li, C. Xiong, and S. C. H. Hoi, "CoMatch: Semi-supervised learning with contrastive graph regularization," 2020, *arXiv: 2011.11183*.

[48] Y. Zhang, X. Zhang, R. C. Qiu, J. Li, H. Xu, and Q. Tian, "Semi-supervised contrastive learning with similarity co-calibration," 2021, *arXiv:2105.07387*.

[49] B. Han et al., "A survey of label-noise representation learning: Past, present and future," 2020, *arXiv: 2011.04406*.

[50] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 322–330.

[51] L. Feng, S. Shu, Z. Lin, F. Lv, L. Li, and B. An, "Can cross entropy loss be robust to label noise?," in *Proc. Int. Joint Conf. Artif. Intell.*, 2020, pp. 2206–2212.

[52] L. Jiang, Z. Zhou, T. Leung, L. Li, and L. Fei-Fei, "MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2309–2318.

[53] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4331–4340.

[54] J. Shu et al., "Meta-weight-net: Learning an explicit mapping for sample weighting," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1917–1928.

[55] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 1–9.

[56] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2233–2241.

[57] Y. Yao et al., "Dual T: Reducing estimation error for transition matrix in label-noise learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 7260–7271.
[58] X. Xia et al., "Sample selection with uncertainty of losses for learning with noisy labels," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–23.
[59] H. Wei, L. Feng, X. Chen, and B. An, "Combating noisy labels by agreement: A joint training method with co-regularization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13723–13732.
[60] D. T. Nguyen, C. K. Mummadi, T. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox, "SELF: Learning to filter noisy labels with self-ensembling," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–16.
[61] Q. Wu, Y. Chen, C. Yang, and J. Yan, "Energy-based out-of-distribution detection for graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2023, pp. 1–16.
[62] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–27.
[63] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–12.
[64] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7167–7177.

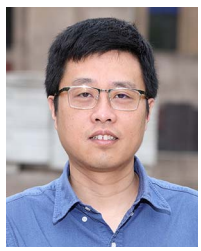
Haobo Wang received the BEng and PhD degrees from Zhejiang University, China, in 2018 and 2023. He is currently an assistant professor in the School of Software and Technology, Zhejiang University. His paper PiCO received the Outstanding Paper Award honorable mention in ICLR 2022. He has published more than 30 papers showcased at top conferences and journals, such as TPAMI, ICLR, NeurIPS, AAAI, IJCAI, and EMNLP. His research interests include machine learning and data mining, especially on weakly-supervised learning and large language models.

Ruixuan Xiao received the BEng degree in computer science from Zhejiang University, in 2022, and is a currently working toward the master degree with the Data Intelligence Laboratory, College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interest focuses on weakly-supervised learning, data mining, and large language models.

Yixuan Li (Member, IEEE) received the PhD degree from Cornell University, in 2017, advised by John E. Hopcroft. She is an assistant professor in the Department of Computer Sciences, the University of Wisconsin-Madison. Subsequently, she was a postdoctoral scholar in the computer science department, Stanford University. Her research focuses on the algorithmic and theoretical foundations of learning in open worlds. She has served as Area Chair for ICLR, NeurIPS, ICML, and Program Chair for Workshop on Uncertainty and Robustness in Deep Learning. She is the recipient of the AFOSR Young Investigator Program (YIP) award, NSF CAREER award, MIT Technology Review Innovator Under 35, Forbes 30 Under 30 in Science, and multiple faculty research awards from Google, Meta, and Amazon. Her works received a NeurIPS Outstanding Paper Award, and an ICLR Outstanding Paper Award Honorable Mention, in 2022.

Lei Feng received the PhD degree in computer science from Nanyang Technological University, Singapore, in 2021. He is currently a visiting professor (full-time) with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He was a professor with the College of Computer Science, Chongqing University, China. He was named to Forbes 30 under 30 Asia 2022 and Forbes 30 Under 30 China 2021. He received the ICLR 2022 outstanding paper award honourable mention. His main research interests include weakly supervised learning, trustworthy deep learning, and data science. He has published more than 60 papers at top conferences and journals, such as ICML, NeurIPS, ICLR, KDD, CVPR, ICCV, AAAI, and IJCAI. He has also served as a senior program committee member for IJCAI 2021 and AAAI 2022, and a program committee member (or reviewer) for other top conferences and journals.

Gang Niu received the PhD degree in computer science from Tokyo Institute of Technology, in 2013. He is currently an indefinite-term senior research scientist, RIKEN Center for Advanced Intelligence Project. Before joining RIKEN, he was a senior software engineer with Baidu and then an assistant professor with the University of Tokyo. He joined RIKEN as a research scientist in 2018, and he was tenured in 2020 and promoted to senior research scientist in 2023. He has published more than 100 journal articles and conference papers, including 34 ICML, 25 NeurIPS (1 oral and 4 spotlights), and 12 ICLR (1 outstanding paper honorable mention, 3 orals, and 1 spotlight) papers. He has co-authored the book "Machine Learning from Weak Supervision: An Empirical Risk Minimization Approach" (the MIT Press). On the other hand, he has served as an area chair/meta-reviewer 23 times and a senior area chair/senior meta-reviewer 1 time. He also serves/has served as an action editor of TMLR, an editorial board member of MLJ, and a guest editor of a special issue at MLJ. Moreover, he has served as a publication chair for ICML 2022, and has co-organized 11 workshops, 1 competition, and 3 tutorials.



Gang Chen (Member, IEEE) received the PhD degree in computer science from Zhejiang University, Hangzhou China. He is a professor with the College of Computer Science, Zhejiang University. He is also the director of Key Laboratory of Intelligent Computing Based Big Data of Zhejiang Province. His research interests include database management technology, intelligent computing based Big Data and massive internet systems. He is a member ACM and a standing member of the China Computer Federation Database Professional Committee.

Junbo Zhao received the PhD degree from New York University, in 2019 under the supervision of Turing Award Laureate Yann LeCun. He is currently a ZJU100-Young assistant professor in the College of Computer Science and Technology at Zhejiang University. He has received numerous awards including the Forbes 30 Under 30 Cover Figure of the technology track 2022, an Alibaba Cloud MVP, a Baidu Young AI Scholar, and the inaugural WAIC Young Scientist. He has published more than 40 papers at top AI conferences, such as NeurIPS, ICML, ICLR and WWW, and gathered more than 15000 citations according to the stats of Google Scholar. His current academic interests are Large Language Models, Table Pre-training, Machine Learning, and AI+X.