Few-Shot Partial-Label Learning

Yunfeng Zhao[1,2] , **Guoxian Yu**[1,2*] , **Lei Liu**[1,2] , **Zhongmin Yan**[1,2] , **Lizhen Cui**[1,2] , **Carlotta Domeniconi**[3]

[1]School of Software Engineering, Shandong University, Jinan, Shandong, China
[2]Joint SDU-NTU Centre for Artificial Intelligence Research, Shandong University, Jinan, China
[3]Department of Computer Science, George Mason University, VA, USA
yunfengzhao@mail.sdu.edu.cn, {gxyu, l.liu, yzm, clz}@sdu.edu.cn, carlotta@cs.gmu.edu

Abstract

Partial-label learning (PLL) generally focuses on inducing a noise-tolerant multi-class classifier by training on overly-annotated samples, each of which is annotated with a set of labels, but only one is the valid label. A basic promise of existing PLL solutions is that there are sufficient partial-label (PL) samples for training. However, it is more common than not to have just few PL samples at hand when dealing with new tasks. Furthermore, existing few-shot learning algorithms assume precise labels of the support set; as such, irrelevant labels may seriously mislead the meta-learner and thus lead to a compromised performance. How to enable PLL under a few-shot learning setting is an important problem, but not yet well studied. In this paper, we introduce an approach called FsPLL (Few-shot PLL). FsPLL first performs adaptive distance metric learning by an embedding network and rectifying prototypes on the tasks previously encountered. Next, it calculates the prototype of each class of a new task in the embedding network. An unseen example can then be classified via its distance to each prototype. Experimental results on widely-used few-shot datasets demonstrate that our FsPLL can achieve a superior performance than the state-of-the-art methods, and it needs fewer samples for quickly adapting to new tasks.

1 Introduction

In partial label learning (PLL) [Cour *et al.*, 2011], each ‘partial-label’ (PL) training sample is annotated with a set of candidate labels, among which only one is the ground-truth label. The aim of PLL is to induce a noise-tolerant multi-class classifier from such PL samples. PLL is currently one of the most prevalent weakly-supervised learning paradigms, which include inaccurate supervision, where the given labels do not always correspond to the ground-truth; incomplete supervision, where only a subset of the training data is labeled; and inexact supervision, where the training data have only coarse-grained labels [Zhou, 2018]. This paper focuses on the first paradigm, where the given labels of the training data do not always represent the ground-truth. This learning problem arises in diverse domains, where a large number of inaccurately annotated samples can be easily collected, and it is very difficult (or impossible) to identify the true labels from the given ones [Zheng *et al.*, 2017; Tu *et al.*, 2020].

Let $\mathcal{X} \in \mathbb{R}^d$ denotes the d-dimensional instance feature space and $\mathcal{Y} = \{0,1\}^l$ denotes the label space with l distinct labels. The aim of PLL is to learn a noise-robust multi-class classification model $f : \mathcal{X} \rightarrow \mathcal{Y}$ with the PL dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | 1 \leq i \leq n\}$, where $\mathbf{x}_i \in \mathcal{X}$ is the feature vector of the i-th instance, $\mathbf{y}_i$ is the multi-hot label vector of candidate labels ($\mathcal{Y}_i \subset \mathcal{Y}$) of the i-th instance, and $z_i \in \mathcal{Y}_i$ is the unknown ground-truth label of this instance. The key challenge to address the PLL problem is to recover the ground-truth label concealed within the candidate label set for every training instance. Existing PLL methods can be roughly categorized into averaging-based disambiguation and identification-based disambiguation. The former class of methods typically equally treats each candidate label during the process of model induction, and performs label prediction by averaging the modeling outputs [Cour *et al.*, 2011; Gong *et al.*, 2017]. The second category of methods models the ground-truth label of the training instance as a latent variable, and estimates it via an iterative refining procedure [Yu and Zhang, 2017; Yu *et al.*, 2018; Chai *et al.*, 2020].

These PLL approaches rely on the assumption that sufficient labeled/unlabeled training data which are relevant to the task are available. They don’t perform well in a **few-shot** scenario, where each class has only few training samples, annotated with inaccurate labels. Although Few-Shot Learning (FSL) has been extensively applied in diverse domains [Snell *et al.*, 2017; Finn *et al.*, 2017; Wang *et al.*, 2020], the existing FSL methods typically assume that the labels of the few-shot support samples are **noise free**. Unfortunately, the violation of this assumption seriously compromise the performance of the few-shot classifier, as shown in our experiments. To the best of our knowledge, how to make FSL effective with few-shot PL samples, is an open and under-studied problem. To bridge this gap, we propose a Few-shot PLL approach (**FsPLL**), which is based on the prototypical network [Snell *et al.*, 2017] and on the local manifold [Belkin *et al.*, 2006]

[*]corresponding author. This work is supported by NSFC (62031003 and 61872300)

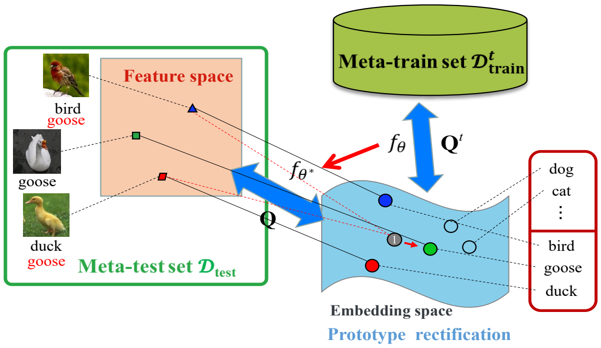


Figure 1: The overall schematic framework of FsPLL. FsPLL learns an embedding network (f_θ) and rectifies label confidence matrix $\{\mathbf{Q}^t\}_{t=1}^T$ to perform adaptive distance metric learning based on PL samples previous encountered ($\mathcal{D}^t_{train}$). Next, it updates label confidence matrix $\mathbf{Q}$ and rectifies prototypes w.r.t. the new task ($\mathcal{D}_{test}$) in the embedding space. An unseen example can then be classified via its distance to prototypes. The 'circle with 1' in the embedding space is the contaminated prototype (no rectification) of 'goose'.

in feature space, which states that instances that have similar feature vectors are more likely to share a same ground-truth label. More specifically, FsPLL first aims at iterative rectifying the ground-truth class prototypes of support PL samples and learning an embedding network, where, based on previous tasks, every sample is closer to its ground-truth prototype, and further apart from its non-ground-truth prototypes. Next, it calculates the prototype of each class of the new task by embedding network and prototype rectification. Then, an unseen example can be classified via its distance to each prototype. The whole framework of FsPLL is illustrated in Fig. 1.

The main contributions of our work are as follows:

(i) We focus on a practical and general PLL setting, where the training samples of the target task are few-shot. We also tackle the problem of noisy labels of few-shot support samples, which can seriously mislead the meta-learner when adapting to the target task. Both issues are not addressed by existing PLL solutions and few-shot/meta learning methods.

(ii) We introduce a prototype rectification strategy with prototypical embedding network to learn the underlying ground-truth prototypes of support and query PL samples, which is less impacted by irrelevant labels and can more credibly adapt to new tasks.

(iii) Extensive experiments on benchmark few-shot datasets show that our FsPLL outperforms the state-of-the-art PLL approaches [Zhang *et al.*, 2016; Wu and Zhang, 2018; Feng and An, 2019; Wang *et al.*, 2019] and baseline FSL methods [Snell *et al.*, 2017; Finn *et al.*, 2017]. The overlook of irrelevant labels of few-shot PL samples indeed seriously compromises the performance of FSL methods, and our FsPLL can greatly remedy this problem.

2 Related Work

2.1 Partial Label Learning

PLL is different from learning from noisy labels [Natarajan *et al.*, 2013], where training samples are incorrectly annotated with the wrong label; it is also different from semi-supervised learning [Belkin *et al.*, 2006], where some training samples are completely unlabeled but can be leveraged for training; and also different from weak-label learning [Sun *et al.*, 2010; Dong *et al.*, 2018], where the labels of training samples are incomplete. The current efforts for PLL can be roughly grouped into two categories: the averaging-based and the identification-based disambiguation.

The averaging-based disambiguation technique generally induces the classifier model by treating all candidate labels equally. Following this protocol, some instances-based methods [Hüllermeier and Beringer, 2006; Gong *et al.*, 2017] classify the ground-truth y of an unseen instance $\mathbf{x}$ by averaging the candidate labels of its neighbors, i.e., $y = \arg\max_{y\in\mathcal{Y}} \sum_{\mathbf{x}_i\in\mathcal{N}(\mathbf{x})} \mathbb{I}(y\in\mathcal{S}_i)$, where $\mathcal{S}_i$ denotes the candidate label set of the i-th instance and $\mathcal{N}(\mathbf{x})$ denotes the set of neighbors of instance $\mathbf{x}$, while other parametric methods [Cour *et al.*, 2011; Zhang *et al.*, 2016] aim at inducing a parametric model θ by maximizing the gap between the average modeling output of the candidate labels and that of the non-candidate ones, i.e., $\max(\sum_{i=1}^m(\frac{1}{|\mathcal{S}_i|}\sum_{y\in\mathcal{S}_i} F(\mathbf{x}_i, y;\theta) - \frac{1}{|\widehat{\mathcal{S}}_i|}\sum_{\widehat{y}\in\widehat{\mathcal{S}}_i} F(\mathrm{x}_i, \widehat{y};\theta)))$ where $\widehat{\mathcal{S}}_i$ denotes the set of non-candidate labels. As to the identification-based disambiguation technique [Feng and An, 2019; Yan and Guo, 2020], the ground-truth labels of the training instances are seen as latent variables and to be optimized by an iterative refining procedure. Following this paradigm, some methods train the model based on the maximum likelihood criterion [Jin and Ghahramani, 2002] or the maximum margin criterion [Nguyen and Caruana, 2008]. Recently, some teams mine the topological information [Zhang *et al.*, 2016; Feng and An, 2018] in the instance feature space to help the optimization of label confidence.

Nevertheless, although these methods can disambiguate labels and induce a noise-tolerance classifier by different techniques, they can hardly work in a more universal scenario, in which the PL samples we collected are few-shot, which break the premise of many-shot training samples per label for inducing a PLL classifier. In fact, existing PLL methods still work in a close label set fashion. But in practice, we may often come into new scenarios, where we can only collect few-shot PL samples and each target label is annotated to several samples. To enable PLL in this general setting, we propose FsPLL to learn noise-robust class prototypes by an embedding network and by rectifying prototypes therein.

2.2 Few-shot Learning

FSL [Li *et al.*, 2006] is an example of meta-learning [Huisman *et al.*, 2020], where a learner is trained on several related tasks during the meta-training phase, so that it can generalize well to unseen (but related) tasks using just few samples with supervision during the meta-testing phase. Existing FSL solutions mainly focus on supervised learning problems, and usually one may term as N-*way* K-*shot* classification, where N stands for the number of classes and K means the number of training samples per class, so each task contains KN samples. Given limited support samples for training, unreliable empirical risk minimization is the core issue of FSL, and existing solutions for FSL can be grouped from the perspective of data, model and algorithm [Wang *et al.*, 2020]. Data augmentation-based FSL methods aim to acquire more supervised training samples by generating more samples from original few-shot samples, weakly-labeled/unlabeled data or similar datasets [Douze *et al.*, 2018], and thus to reduce the uncertainty of empirical risk minimization. Model-based FSL methods typically manage to shrink the ambient hypothesis space into a smaller one by extracting prior knowledge in the meta-training phase [Snell *et al.*, 2017; Ren *et al.*, 2018], so empirical risk minimization becomes more reliable and overfitting issue is reduced. Algorithm-based FSL approaches use prior knowledge to guide the seek of optimal model parameters by providing a good initialized parameter or directly learning an optimizer for new tasks [Finn *et al.*, 2017].

Unfortunately, most FSL methods ideally assume the support samples in meta-testing set is with accurate supervision, namely, these samples are precisely annotated with labels. But these support samples are PL ones with irrelevant labels, which mislead the adaption of FSL methods toward the target task (as shown in Fig. 1) and cause a compromised performance. To address this problem, our FsPLL performs the optimization of embedding network and prototype rectification therein in an iterative manner. In this way, the learnt embedding network and prototypes are less impacted by irrelevant labels of PL samples, and can credibly adapt to new tasks.

3 The Proposed Methodology

Suppose we are given a small support/training set of n PL samples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | 1 \leq i \leq n\}$ and its corresponding label space and feature space are $\mathcal{Y} = \{0,1\}^l$ and $\mathcal{X} \in \mathbb{R}^d$, respectively. The goal of FsPLL is to induce a multi-class classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$, which can precisely predict the ground-truth label of an unseen instance $\mathbf{x}$ under this few-shot classification scenario. Different from existing PLL methods, FsPLL should and can utilize the knowledge previously acquired from meta-training phase to quickly adapt to the new classification task $\mathcal{D}$ in the meta-testing phase. In the meta-training phase, FsPLL learns an embedding network (meta-knowledge) to project PL samples more nearby with their ground-truth prototypes and apart from their non ground-truth prototypes by iteratively rectifying these prototypes in this embedding space. In the meta-testing phase, it rectifies the prototypes of support PL samples using the embedding network and then classifies new samples by their distance to rectified prototypes in the embedding space. In this paper, we take Prototypical Network (PN) [Snell *et al.*, 2017] as the base of our embedding network. The framework overview of FsPLL is given in Fig. 1. The following subsections elaborate on the two phases.

3.1 Meta-training Phase

The meta-training phase mainly aims to extract prior knowledge from multiple relevant tasks for the target task. Suppose we are given $T \gg 1$ few-shot datasets (tasks) denoted as $\mathcal{D}_{train}^t$ $(1 \leq t \leq T)$. For each dataset $\mathcal{D}_{train}^t = \{\mathbf{X}_s^t, \tilde{\mathbf{X}}_q^t, \mathbf{Y}^t\}$, where $\mathbf{X}_s^t = (\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_{n_s}^t) \in \mathbb{R}^{d \times n_s}$ denotes the data matrix of support samples, $\tilde{\mathbf{X}}_q^t = (\tilde{\mathbf{x}}_1^t, \tilde{\mathbf{x}}_2^t, \cdots, \tilde{\mathbf{x}}_{n_q}^t) \in \mathbb{R}^{d \times n_q}$ denotes data matrix of query samples, $\mathbf{Y}^t = (\mathbf{y}_1^t, \mathbf{y}_2^t, \cdots, \mathbf{y}_{n_s}^t) \in \mathbb{R}^{l \times n_s}$ is the corresponding label matrix of support samples, and $n_s + n_q < n$. $\mathbf{Y}_{ci}^t = 1$ means the c-th label is a candidate label of the i-th sample; $\mathbf{Y}_{ci}^t = 0$ otherwise. Let $\mathbf{Q}^t \in \mathbb{R}^{l \times n_s}$ denotes the underlying label confidence matrix of support samples and it is initialized as $\mathbf{Y}^t$, where $\mathbf{Q}_{ci}^t$ indicates the confidence of the c-th label as the ground-truth label of the i-th sample.

From these datasets, we aim at learning an embedding network, i.e., $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^m$, by which we can obtain the representation of every label in the embedding space and can be more robust to irrelevant labels of support samples therein. Suppose $\mathbf{P}^t = (\mathbf{p}_1^t, \mathbf{p}_2^t, \ldots, \mathbf{p}_l^t) \in \mathbb{R}^{m \times l}$ is the prototype/representation matrix of l class labels of the t-th task, where $\mathbf{p}_c^t$ denotes the prototype of the c-th label in the embedding space. PN [Snell *et al.*, 2017] computes the prototype by $\mathbf{p}_c^t = \frac{\sum_{i=1}^{n_s} \mathbf{Y}_{ci}^t \times f_\theta(\mathbf{x}_i^t)}{\sum_{i=1}^{n_s} \mathbf{Y}_{ci}^t}$, while Semi-PN [Ren *et al.*, 2018], a variant of PN, further uses unlabeled examples to improve the prototype learning. They both simply take all PL samples annotated with the c-th label to induce the prototype, ignoring that some PL samples actually not annotated with this label. Therefore, PN and Semi-PN give contaminated prototypes. For example, prototype of goose ('circle with 1') in Fig. 1 is misled by irrelevant labels, which consequently compromises the classification performance, especially when support PL samples with excessive irrelevant labels. To address this issue, FsPLL performs prototype rectification and label confidence update in an iterative way to seek noise-robust embedding network and prototypes in the embedding space, as shown in Fig. 1. FsPLL defines each prototype based on the confidence weighted mean of corresponding support samples in the embedding space as follows:

$$\mathbf{p}_c^t = \frac{\sum_{i=1}^{n_s} \mathbf{Q}_{ci}^t \times f_\theta(\mathbf{x}_i^t)}{\sum_{i=1}^{n_s} \mathbf{Q}_{ci}^t}. \tag{1}$$

Unlike prototypes optimized by PN, FsPLL rectifies the prototypes using iterative updated label confident matrix $\mathbf{Q}^t$, and thus explicitly accounts for the irrelevant labels of samples.

It is expected for a sample to be closer to its ground-truth prototype in the embedding space; this would enable a confident label prediction in this space. Given this, we use a softmax to update the label confidence matrix $\mathbf{Q}^t$ as follows:

$$\mathbf{Q}_{ci}^t = \begin{cases} \frac{\exp(-d(f_\theta(\mathbf{x}_i^t), \mathbf{p}_c^t))}{\sum_{c=1}^{l} \exp(-d(f_\theta(\mathbf{x}_i^t), \mathbf{p}_c^t)) \times \mathbf{Y}_{ci}^t}, & \text{if } \mathbf{Y}_{ci}^t = 1 \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where $d(f_\theta(\mathbf{x}_i^t), \mathbf{p}_c^t)$ quantifies the Euclidean distance between sample $\mathbf{x}_i^t$ and prototype $\mathbf{p}_c^t$ in the embedding space. The labels of a PL sample can be disambiguated by referring to labels of its neighborhood samples [Wang *et al.*, 2019]. We observe that PN and Eq. (1) disregard the neighborhood support samples when computing the prototype. Unlike these PLL methods that disambiguate in the original feature space or linearly projected subspace, FsPLL further updates the label confidence matrix in the embedding space as follows:

$$\mathbf{Q}_{ci}^t = \mathbf{Q}_{ci}^t + \frac{\lambda}{|\mathcal{N}_k(\mathbf{x}_i^t)|} \sum_{\mathbf{x}_j^t \in \mathcal{N}_k(\mathbf{x}_i^t)} \mathbf{Q}_{cj}^t, \text{ if } \mathbf{Y}_{ci}^t = 1, \tag{3}$$

where $\mathcal{N}_k(\mathbf{x}_i^t)$ includes the k-nearest samples of $\mathbf{x}_i^t$, and the neighborhood is determined by Euclidean distance in the embedding space. λ trade-offs the confidence from the sample itself and those from neighborhood samples. In this way, FsPLL utilizes local manifold of samples to rectify prototypes.

Based on the rectified prototypes and embedding network f_θ, we can predict the label of a query sample with a softmax over its distances to all prototypes in the embedding space as:

$$p_\theta(z_j^t = c \mid \tilde{\mathbf{x}}_j^t) = \frac{\exp(-d(f_\theta(\tilde{\mathbf{x}}_j^t), \mathbf{p}_c^t))}{\sum_{i=1}^{l} \exp(-d(f_\theta(\tilde{\mathbf{x}}_j^t), \mathbf{p}_i^t))}, \qquad (4)$$

where z_j^t is the unknown ground-truth label of the j-th query sample. To make the representation of every query sample in the embedding space closer to its ground-truth prototype and apart from its non ground-truth prototypes, FsPLL minimizes the negative log-probability of the most likely label of a query example as follows:

$$\mathbf{J}(\theta, \tilde{\mathbf{x}}_i^t) = -\log(\max_{c=1,\cdots,l} p_\theta(z_j^t = c \mid \tilde{\mathbf{x}}_i^t)). \qquad (5)$$

By minimizing the above equation, FsPLL can obtain the rectified prototypes $\mathbf{P}^t$ and the corresponding embedding network parameterized by f_θ for task $\mathcal{D}_{train}^t$. We want to remark that the l-th labels for different tasks is not always the same.

The meta-training phase involves a lot of different tasks, each of which is composed of support/query samples. To enable a good generalization ability, it attempts to gain the optimal mode parameter θ^* by minimizing the average negative log-probability of the most likely labels of all query samples over T tasks as follows:

$$\theta^* = \arg\min_\theta \sum_{t=1}^{T} \frac{1}{n_q} \sum_{i=1}^{n_q} \mathbf{J}(\theta, \tilde{\mathbf{x}}_i^t). \qquad (6)$$

To this end, FsPLL obtains an embedding network f_{θ^*} that is robust to irrelevant labels of PL samples across T tasks. Via this network, a PL sample in the embedding space is made closer to its ground-truth prototype than to other prototypes, and the generalization and fast adaption ability are pursued among T different tasks.

3.2 Meta-testing Phase

In the meta-testing phase, we are only given a small set of PL samples, which compose the target task with support and query samples. These support samples are overly-annotated with irrelevant labels, while query samples are without label information. We want to highlight that the labels of these PL samples are *disjoint* with the labels used in the meta-training phase. In other words, the PL samples are few-shot ones. Here, FsPLL aims to use the meta-knowledge (embedding network f_{θ^*}) acquired in the meta-training phase to precisely annotate the query samples based on the inaccurately supervised few-shot support examples.

Formally, FsPLL aims to quickly generalize to a new task $\mathcal{D}_{test} = \{\mathbf{X}_s, \tilde{\mathbf{X}}_q, \mathbf{Y}\}$, where $\mathbf{X}_s \in \mathbb{R}^{d \times n_s}$, $\tilde{\mathbf{X}}_q \in \mathbb{R}^{d \times n_q}$ and $\mathbf{Y} \in \mathbb{R}^{l \times n_s}$ denote the data matrices of support examples, of query examples, and of labels of query examples, respectively. Alike the meta-training phase, FsPLL first computes the prototypes $\mathbf{P} \in \mathbb{R}^{m \times l}$ of this new task in the embedding space using the confidence-weighted mean of support samples $\mathbf{X}_s$ and label confidence matrix $\mathbf{Q}$ as in Eq. (1). Then the label confidence matrix $\mathbf{Q}$ of the support samples is updated based on a softmax over their distances to prototypes as in Eq. (2) and local manifold as in Eq. (3). FsPLL repeats the above two steps to rectify the prototypes and update label confidence matrix for adapting to the target task. Note, the embedding network f_{θ^*} is fixed during the above repetitive optimization.

Given a query sample $\mathbf{x}_i$, FsPLL classifies its label z_i using its distance to rectified prototypes $\mathbf{P} \in \mathbb{R}^{m \times l}$ as follows:

$$z_i = \arg\max_q p_{\theta^*}(z_i = q \mid \mathbf{x}_i) \ \ (q = 1, \cdots, l). \qquad (7)$$

4 Experiments

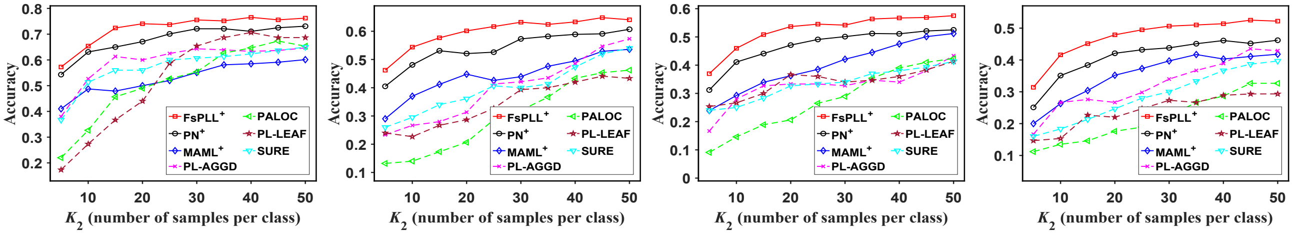
4.1 Experimental Setup

Datasets. We conduct experiments on two benchmark FSL datasets (**Omniglot** [Lake *et al.*, 2011] and **miniImageNet** [Vinyals *et al.*, 2016]). Following the canonical protocol adopted by previous PLL methods [Wang *et al.*, 2019; Zhang *et al.*, 2016], we generate the semi-synthetic PL datasets on Omniglot and miniImageNet by two controlling parameters p and r. p controls the proportion of PL samples, and r controls the number of irrelevant labels of a PL sample, which are randomly selected from the label space of corresponding task. Each $\mathcal{D}_{train}^t$ consisted of $N_1 = 30$ classes were randomly sampled from 4800/80 train classes of Omniglot/miniImageNet without replacement. As to the meta-testing set, we randomly selected another N_2 classes from 1692/20 test classes without replacement. For each selected class, $K_1 = 5$ (K_2) samples were randomly chosen from 20/600 samples without replacement for the meta-training (meta-testing) support samples, and the remaining/15 samples per class were randomly chosen as the query samples.

Compared Methods. We compare FsPLL against four recent PLL methods (PL-LEAF [Zhang *et al.*, 2016], PALOC [Wu and Zhang, 2018], SURE [Feng and An, 2019], PL-AGGD [Wang *et al.*, 2019]), two representative FSL methods (MAML [Finn *et al.*, 2017], PN [Snell *et al.*, 2017]), and FsPLL-nM (a variant of FsPLL) which disregards the local manifold of training samples but updates the label confidence matrix via Eq. (2) for prototype rectification.Each compared method is configured with the suggested parameters according to the corresponding literature. As to our FsPLL, the trade-off parameter λ is fixed as 0.5 (0 for FsPLL-nM), the number of nearest neighbors $k = K_2 - 1$, the number of iterations for prototype rectification in each epoch is fixed to 10. In addition, we use the Adam [Kingma and Ba, 2015] optimizer, the learning rate is fixed as 0.001 and cut into half per 20 epochs. For fair comparison purposes, FsPLL also uses the embedding network proposed by [Vinyals *et al.*, 2016], which is also used by compared FSL methods. For Omniglot, the size of prototypes is $m = 64$; while for miniImageNet, $m = 1600$. For non-FSL PLL methods, they also used the image features extracted by [Vinyals *et al.*, 2016]. They only use the samples in meta-testing set for training and validation. We randomly generate $\mathcal{D}_{train}$ ($T = 100$) as the meta-training tasks in each round, and report average results on $\mathcal{D}_{test}$ in 100 rounds for reducing the randomness.

	$N_2 = 5$		$N_2 = 10$		$N_2 = 20$		$N_2 = 30$	
	$K_2 = 5$	$K_2 = 10$	$K_2 = 5$	$K_2 = 10$	$K_2 = 5$	$K_2 = 10$	$K_2 = 5$	$K_2 = 10$
	$r = 1$							
FsPLL	**.892±.083**	**.895±.051**	**.789±.045**	**.823±.067**	**.712±.034**	**.757±.056**	**.665±.015**	**.701±.046**
FsPLL-nM	.852±.092	.886±.072	.776±.070	.816±.062	.695±.053	.745±.047	.643±.008	.693±.042
PN	.579±.104	.636±.105	.435±.070	.485±.071	.317±.043	.360±.044	.255±.032	.291±.034
MAML	.673±.079	.647±.097	.592±.067	.642±.053	.514±.061	.544±.032	.421±.018	.475±.065
PL-AGGD	.664±.118	.777±.103	.576±.086	.714±.076	.450±.053	.649±.063	.459±.043	.601±.057
PALOC	.616±.116	.726±.111	.528±.075	.651±.078	.447±.058	.568±.058	.392±.047	.513±.049
SURE	.629±.125	.782±.106	.574±.082	.721±.074	.506±.060	.657±.056	.465±.041	.604±.046
PL-LEAF	.629±.117	.768±.107	.568±.087	.712±.070	.495±.060	.630±.060	.452±.043	.592±.054
FsPLL$^+$	**.995±.029**	**.997±.009**	**.990±.020**	**.993±.012**	**.986±.009**	**.990±.010**	**.981±.009**	**.986±.008**
PN$^+$	.965±.046	.985±.030	.956±.037	.981±.021	.939±.027	.968±.020	.924±.025	.958±.018
MAML$^+$	.858±.016	.902±.036	.849±.026	.877±.014	.774±.019	.831±.023	.638±.035	.795±.189
	$r = 2$							
FsPLL	**.673±.098**	**.742±.073**	**.712±.068**	**.756±.063**	**.654±.049**	**.689±.063**	**.598±.063**	**.602±.038**
FsPLL-nM	.616±.171	.706±.136	.566±.100	.665±.087	.494±.065	.584±.056	.442±.053	.527±.048
PN	.476±.121	.559±.114	.378±.073	.442±.074	.270±.044	.321±.047	.213±.032	.258±.033
MAML	.498±.101	.553±.098	.458±.078	.549±.093	.427±.037	.472±.075	.397±.043	.437±.036
PL-AGGD	.496±.131	.668±.129	.490±.086	.664±.082	.451±.055	.578±.061	.416±.053	.545±.054
PALOC	.473±.117	.611±.125	.456±.083	.591±.086	.385±.056	.525±.057	.402±.049	.524±.061
SURE	.488±.133	.665±.132	.496±.091	.670±.082	.450±.064	.593±.057	.413±.046	.561±.051
PL-LEAF	.484±.134	.650±.125	.489±.086	.645±.083	.436±.059	.586±.060	.398±.045	.525±.058
FsPLL$^+$	**.975±.076**	**.997±.009**	**.991±.010**	**.994±.010**	**.986±.009**	**.989±.012**	**.980±.009**	**.985±.007**
PN$^+$	.825±.117	.926±.076	.871±.064	.945±.041	.850±.047	.926±.032	.826±.040	.908±.030
MAML$^+$	.675±.076	.798±.056	.783±.024	.760±.076	.668±.023	.727±.025	.619±.026	.679±.450

Table 1: Classification accuracy (mean±std) of comparison methods on **Omniglot**. $\{$FsPLL, PN, MAML$\}^+$ separately use precise labels of meta-training samples. $N_2(K_2)$: the number of support classes (training samples per class). The best performance in each setting is **boldface**.



(a) $N_2 = 5$ (b) $N_2 = 10$ (c) $N_2 = 15$ (d) $N_2 = 20$

Figure 2: Accuracy of each compared method vs. K_2 (number of support samples per class in the meta-testing set) on **miniImageNet** ($r = 1$).

4.2 Result Analysis

Results on Omniglot. Table 1 reports the accuracy of each compared method on **Omniglot** as p is fixed to 1, r is fixed to 1 or 2, N_2 is fixed to 5, 10, 20 or 30, K_2 is fixed to 5 or 10. Due to the page limit, the results of $r = 3$ are not reported, while similar trends can be observed also. From this Table, we have the following observations:

(i) FsPLL significantly outperforms other compared methods across all the settings, which proves the effectiveness of FsPLL on few-shot PL samples. The performance margin between FsPLL and non-FSL methods are more prominent w.r.t. a small K_2, since these non-FSL methods build on the promise of many-shot PL samples for training. Although PN and MAML additionally use many tasks with support PL samples for the few-shot setting, they often lose to many-shot PLL methods. That is because they are both heavily misled by irrelevant labels of support samples. In contrast, our FsPLL is much less impacted by irrelevant labels of support samples, it reduces the negative impact of irrelevant labels by iteratively rectifying the prototypes and embedding network. By virtue of precise labels of meta-train samples, PN$^+$ and MAML$^+$ outperform many-shot PLL methods, but they still lose to FsPLL$^+$ by a large margin. These observations confirm that the noisy labels of PL samples heavily mislead the adaption of meta-learner toward the target task.

(ii) Prototype rectification can greatly reduce the negative impact of irrelevant labels of PL samples. This is supported by the performance margin between FsPLL (FsPLL$^+$) and PN (PN$^+$). They both perform distance metric learning in the embedding space to learn prototypes and classify samples therein, but FsPLL additionally rectifies the prototypes in the embedding space by explicitly modeling irrelevant labels and mining local manifold.

(iii) Local manifold helps prototype rectification, this is verified by the clear margin between FsPLL and FsPLL-nM, especially when the number of irrelevant labels is large.

(iv) As N_2 steps from 5 to 30 under a fixed K_2, the performance of each compared method gradually decreases. This is due to the increased class labels and task complexity. The random guess accuracy decrease from 1/5 to 1/30. Even though, FsPLL (FsPLL$^+$) always maintains a better performance than PN (PN$^+$) and MAML (MAML$^+$). On the other hand, as the increase of K_2 under a fixed N_2, each compared method has an improved performance, since more support samples can be used for training. We see non-FSL PLL methods frequently outperform FSL methods (PN and MAML) when $K_2 = 10$. This fact again proves the vulnerability of FSL methods on few-shot PL samples.

(v) As the increase of r, all methods have a reduced performance, since support samples have more irrelevant labels, which seriously compromise the performance of many-shot PLL and FSL methods. This fact signifies the importance to account for PL samples. All compared methods have a relatively large standard deviation, that is due to noisy labels were randomly injected, and more noisy labels cause an even larger fluctuation. We applied signed-rank test to check the statistical significance between FsPLL/FsPLL$^+$ and other compared methods, all p-values are small than 0.001.

Results on miniImageNet. We also conduct experiments on **miniImageNet** with the following control setting: $r \in \{1, 2, 3\}$ with $p = 1$, $N_2 \in \{5, 10, 15, 20\}$ and $K_2 \in [5, 50]$. We enlarge the range of K_2 to check how FsPLL works in many-shot setting. Due to page limit, we only report the results of compared methods when $r = 1$, while similar trends can be observed with other settings. As shown in Fig. 2, FsPLL again outperforms state-of-the-art FSL and many-shot PLL methods under different K_2 shots, and the conclusions are similar as those on Omniglot. With the increase of K_2, all methods show an increased performance, and FsPLL still has a higher accuracy than other methods when $K_2 > 20$, which proves the effectiveness of FsPLL in many-shot settings.

4.3 Further Analysis

Impact of PL samples on FSL methods. We conduct additional experiments to further investigate the impact of noisy support set of meta-training and meta-testing and if FsPLL could be applied to the standard few-shot classification cases, where each sample is precisely labeled. For this investigation, we introduce another variant FsPLL^{++}, which uses precise labels of support samples in the meta-training and meta-testing stages. For comparison, we introduce PN^{++} for PN. So FsPLL^{++}/PN^{++} gives the upper bound performance of FsPLL/PN. Fig. 3 shows the performance of FsPLL and PN and their variants under the setting of $N_2 = 10$, $K_2 = 5$ and $r = 2$ on Omniglot. In the figure, FsPLL/PN uses PL samples both in the meta-training and meta-testing stages; while FsPLL$^+$/PN$^+$ uses precise labels of support samples in the meta-training stage, and PL samples in the meta-testing stage. FsPLL significantly outperforms PN whenever there are support PL samples with irrelevant labels. They can have a comparable performance with precise labels of all support samples. FsPLL improves the accuracy of PN by 88%, and FsPLL$^+$ improves this of PN$^+$ by 13%. More importantly, FsPLL$^+$ has a similar accuracy with FsPLL^{++}. These results not only confirm the negative impact of noisy PL samples on FSL methods, but also prove the effectiveness of FsPLL on handling noisy labels of PL samples and FsPLL can also be applied to the standard few-shot classification cases.

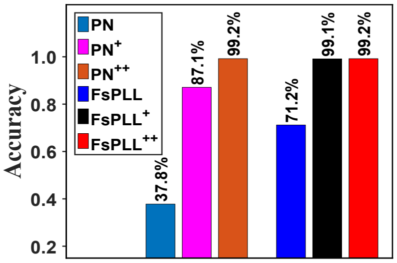
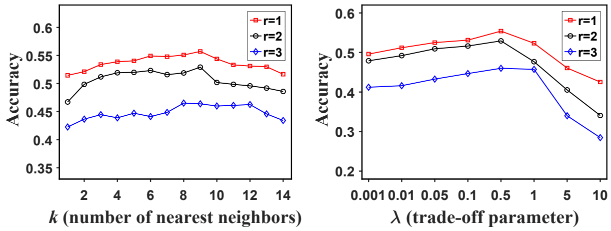


Figure 3: The performance of PN and FsPLL in three different settings on Omniglot. FsPLL^{++} and PN^{++} use precise labels of meta-training and meta-testing samples, give the upper bound accuracy.



(a) Accuracy vs. k

(b) Accuracy vs. λ

Figure 4: Accuracy of FsPLL on miniImageNet under different input values of k and of λ, here $N_2 = 10$ and $K_2 = 10$. (a) Accuracy varies with k ($\lambda = 0.5$); (b) Accuracy varies with λ ($k = K_2 - 1$).

Parameter analysis. We study the parameter sensitivity of FsPLL w.r.t. λ and k (see Eq. (3)), which uses the local manifold to update the label confidence matrix, and consequently rectify the prototype and embedding network f_θ. As shown in Fig. 4(a) and Fig. 4(b), FsPLL first manifests a gradually increased accuracy until $k \approx K_2 - 1$ ($\lambda \approx 0.5$). This trend shows the benefit of local manifold for updating the label confidence matrix and rectifying the prototypes.However, the accuracy starts to decrease as k and λ further increase. That is due to the over-weight (large λ) of local manifold and the inclusion of unreliable neighbors (large k) from other classes.

5 Conclusion

This paper studies the problem of few-shot learning with noisy support samples and proves noisy labels of support samples can greatly compromise the performance. We introduce a Few-shot Partial Label Learning approach (FsPLL) to address this problem. FsPLL learns an embedding network and rectifies prototypes to reduce the impact of noisy labels. Extensive experimental results prove the effectiveness of FsPLL in both few-shot and many-shot settings.

References

[Belkin *et al.*, 2006] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7(11):2399–2434, 2006.

[Chai *et al.*, 2020] Jing Chai, Ivor W Tsang, and Weijie Chen. Large margin partial label machine. *TNNLS*, 31(7):2594–2608, 2020.

[Cour *et al.*, 2011] Timothee Cour, Ben Sapp, and Ben Taskar. Learning from partial labels. *JMLR*, 12:1501–1536, 2011.

[Dong *et al.*, 2018] Hao-Chen Dong, Yu-Feng Li, and Zhi-Hua Zhou. Learning from semi-supervised weak-label data. In *AAAI*, pages 2926–2933, 2018.

[Douze *et al.*, 2018] Matthijs Douze, Arthur Szlam, Bharath Hariharan, and Hervé Jégou. Low-shot learning with large-scale diffusion. In *CVPR*, pages 3349–3358, 2018.

[Feng and An, 2018] Lei Feng and Bo An. Leveraging latent label distributions for partial label learning. In *IJCAI*, pages 2107–2113, 2018.

[Feng and An, 2019] Lei Feng and Bo An. Partial label learning with self-guided retraining. In *AAAI*, pages 3542–3549, 2019.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017.

[Gong *et al.*, 2017] Chen Gong, Tongliang Liu, Yuanyan Tang, Jian Yang, Jie Yang, and Dacheng Tao. A regularization approach for instance-based superset label learning. *TCYB*, 48(3):967–978, 2017.

[Huisman *et al.*, 2020] Mike Huisman, Jan N van Rijn, and Aske Plaat. A survey of deep meta-learning. *arXiv preprint arXiv:2010.03522*, 2020.

[Hüllermeier and Beringer, 2006] Eyke Hüllermeier and Jürgen Beringer. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 10(5):419–439, 2006.

[Jin and Ghahramani, 2002] Rong Jin and Zoubin Ghahramani. Learning with multiple labels. In *NeurIPS*, volume 2, pages 897–904. Citeseer, 2002.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Lake *et al.*, 2011] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Annual Cog. Sci.*, pages 2568–2573, 2011.

[Li *et al.*, 2006] Fei-Fei Li, Fergus Rob, and Perona Pietro. One-shot learning of object categories. *TPAMI*, 28(4):594–611, 2006.

[Natarajan *et al.*, 2013] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NeurIPS*, pages 1196–1204, 2013.

[Nguyen and Caruana, 2008] Nam Nguyen and Rich Caruana. Classification with partial labels. In *KDD*, pages 551–559, 2008.

[Ren *et al.*, 2018] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4077–4087, 2017.

[Sun *et al.*, 2010] Yu-Yin Sun, Yin Zhang, and Zhi-Hua Zhou. Multi-label learning with weak label. In *AAAI*, pages 593–598, 2010.

[Tu *et al.*, 2020] Jinzheng Tu, Guoxian Yu, Carlotta Domeniconi, Jun Wang, Guoqiang Xiao, and Maozu Guo. Multi-label crowd consensus via joint matrix factorization. *KAIS*, 62(4):1341–1369, 2020.

[Vinyals *et al.*, 2016] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NeurIPS*, pages 3637–3645, 2016.

[Wang *et al.*, 2019] Deng-Bao Wang, Li Li, and Min-Ling Zhang. Adaptive graph guided disambiguation for partial label learning. In *KDD*, pages 83–91, 2019.

[Wang *et al.*, 2020] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3):1–34, 2020.

[Wu and Zhang, 2018] Xuan Wu and Min-Ling Zhang. Towards enabling binary decomposition for partial label learning. In *IJCAI*, pages 2868–2874, 2018.

[Yan and Guo, 2020] Yan Yan and Yuhong Guo. Partial label learning with batch label correction. In *AAAI*, pages 6575–6582, 2020.

[Yu and Zhang, 2017] Fei Yu and Min-Ling Zhang. Maximum margin partial label learning. *Machine Learning*, 106(4):573–593, 2017.

[Yu *et al.*, 2018] Guoxian Yu, Xia Chen, Carlotta Domeniconi, Jun Wang, Zhao Li, Zili Zhang, and Xindong Wu. Feature-induced partial multi-label learning. In *ICDM*, pages 1398–1403, 2018.

[Zhang *et al.*, 2016] Min-Ling Zhang, Bin-Bin Zhou, and Xu-Ying Liu. Partial label learning via feature-aware disambiguation. In *KDD*, pages 1335–1344, 2016.

[Zheng *et al.*, 2017] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *VLDB Endowment*, 10(5):541–552, 2017.

[Zhou, 2018] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.