Aligned Objective for Soft-Pseudo-Label Generation in Supervised Learning

Ning Xu [1 2] **Yihao Hu** [1 2] **Congyu Qiao** [1 2] **Xin Geng** [1 2]

Abstract

Soft pseudo-labels, generated by the softmax predictions of the trained networks, offer a probabilistic rather than binary form, and have been shown to improve the performance of deep neural networks in supervised learning. Most previous methods adopt classification loss to train a classifier as the soft-pseudo-label generator and fail to fully exploit their potential due to the misalignment with the target of soft-pseudo-label generation, aimed at capturing the knowledge in the data rather than making definitive classifications. Nevertheless, manually designing an effective objective function for a soft-pseudo-label generator is challenging, primarily because datasets typically lack ground-truth soft labels, complicating the evaluation of the soft pseudo-label accuracy. To deal with this problem, we propose a novel framework that alternately trains the predictive model and the soft-pseudo-label generator guided by a meta-network-parameterized label enhancement objective. The parameters of the objective function are optimized based on the feedback from both the performance of the predictive model and the soft-pseudo-label generator in the learning task. Additionally, the framework offers versatility across different learning tasks by allowing direct modifications to the task loss. Experiments on the benchmark datasets validate the effectiveness of the proposed framework. Source code is available at https://github.com/palm-ml/SEAL

[1]School of Computer Science and Engineering, Southeast University, Nanjing, China [2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China. E-mail: {xning, yhhu, qiaocy, xgeng}@seu.edu.cn. Correspondence to: Xin Geng <xgeng@seu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Soft pseudo-labels, i.e., the soft labels that are not originally included in the training dataset but generated by the softmax predictions of the trained networks, are of paramount importance for state-of-the-art deep learning methods. Numerous previous researches (Zhang et al., 2019; Hinton et al., 2015) have demonstrated the success of soft pseudo-labels in improving the generalization performance of deep neural networks (DNNs) in supervised learning. As pseudo-labels could improve the predictive performance of deep neural networks, they have been successfully applied across various domains, including computer vision (Lukov et al., 2022; Algan & Ulusoy, 2022), natural language processing (Sun et al., 2019; Ngo et al., 2022), and data mining (Li et al., 2015; Xu et al., 2020).

Self-knowledge distillation (Zhang et al., 2019) utilizes a DNNs classifier's predictions as the soft pseudo-labels to train the model itself to improve the test accuracy. Online distillation (Zhang et al., 2018) adopts an ensemble of classifiers' predictions as the soft pseudo-label to learn mutually from each other. Label enhancement (Xu et al., 2023) adopts Graph Convolutional Network (GCN) to generate soft pseudo-labels to deal with the multi-label problem. In addition, soft pseudo-labels are adopted to learn with weak supervision. PENCIL (Yi & Wu, 2019) utilizes back-propagation to probabilistically update and correct soft pseudo-labels via updating the network parameters to deal with noisy labels in the training dataset. To identify the true label, partial label learning (Lv et al., 2020) generates soft pseudo-labels to strengthen the weight of the correct label among the candidate label set for training in each epoch.

Most previous methods adopt classification loss to train a classifier as the soft-pseudo-label generator, which is not aligned with the target of soft-pseudo-label generation. The objective of training a classifier is maximizing the generalization of the classifier itself while training a soft-pseudo-label generator is aimed at capturing the knowledge in the data to generate the appropriate soft pseudo-labels for improving the performance of the student models. Therefore, despite making an effort to tune the generated soft pseudo-labels, previous methods usually do not unleash the full potential of soft pseudo-labels for improving the performance of deep neural networks. Nevertheless, manually designing an effective objective function for a soft-pseudo-label generator is challenging, primarily because datasets typically lack ground-truth soft labels, complicating the evaluation of the soft pseudo-label accuracy.

To deal with the problem, instead of focusing on manually designing an explicit loss for a classifier to output soft pseudo-labels, we propose a novel framework named SEAL, i.e., Soft-psEudo-lAbeL generator with a learnable label enhancement (Xu et al., 2019; 2020) objective, to alternately train the predictive model and the soft-pseudo-label generator guided by a meta-network-parameterized objective function. Specifically, a meta-network dynamically adjusts the parameters of the loss function used for a soft-pseudo-label generator. This optimization is informed by feedback from both the predictive model and the soft-pseudo-label generator, based on their performance in the learning task, creating a feedback loop that refines the training process. Additionally, our framework offers versatility across different learning tasks by allowing direct modifications to the task loss, making it suitable for a range of applications. Our contributions can be summarized as follows:

- We propose a novel framework named SEAL to generate soft pseudo-labels for improving the performance of deep neural networks in supervised learning via inducing a soft-pseudo-label generator optimized by a loss function parameterized by a meta-network. Extensive experiments validate the effectiveness of SEAL.
- We theoretically demonstrate that the soft pseudo-labels could enable the predictive model to achieve a larger sample margin for classes. This expansion, in turn, results in a more robust and tight generalization of the predictive model when trained using soft pseudo-labels.
- SEAL is flexible to learning tasks as the task loss designed for the supervised learning problems can be modified directly. We empirically show that the framework can be applied to partial label learning problems and achieve state-of-the-art learning performance.

2. Related Work

In supervised learning, the generation process of soft pseudo-labels exists in many mainstream algorithms, including knowledge distillation, self-knowledge distillation, label enhancement, and label smoothing.

In knowledge distillation (Hinton et al., 2015), a large teacher model is trained on the dataset with the original hard label and generates soft pseudo-labels to teach a lightweight student model. Self-distillation (Zhang et al., 2019) generates soft pseudo-labels by the model itself, which claims that the soft pseudo-labels can be regarded as the model's deepest section's output used to guide the training of shallow sections. Xu et al. (Xu & Liu, 2019) leverage distorted versions of images to generate soft pseudo-labels for the current batch images. Yun et al. (Yun et al., 2020) introduce class-wise self-distillation that randomly chooses a different sample of the same class to generate a soft pseudo-label for the current sample. Building on this, Zhang et al. (Zhang et al., 2021a) propose to generate soft pseudo-labels based on the statistics of the model prediction for the target category. Taking a different approach, Kim et al. (Kim et al., 2021) suggest generating soft pseudo-labels adaptively by combining the ground-truth and past predictions from the model itself. Some methods also have low training costs for generating soft pseudo-labels. Liang et al. (Liang et al., 2022) propose an efficient self-distillation method that uses the on-the-fly prediction of a network to generate soft pseudo-labels that conform to a distribution. Additionally, Shen et al. (Shen et al., 2022) rearrange the sequential sampling by constraining half of each mini-batch coinciding with the previous iteration. The soft pseudo-labels are generated from the previous iteration, which is computationally efficient.

Label smoothing (Szegedy et al., 2016) prevents the network from becoming over-confident by regulating the model training by substituting one-hot labels with smoothed alternatives, which can also be regarded as soft pseudo-labels. Rather than using a uniform distribution, low-rank adaptive label smoothing (Ghoshal et al., 2021) adopts a more informative noise distribution to generate the soft pseudo-label for each class. To address the challenges in multi-label learning (Zhang & Zhou, 2014), Label enhancement (LE) (Xu et al., 2020) recovers label distribution for training multi-label classifiers, which could also be regarded as soft pseudo-labels. The Graph-Laplacian-based LE method (Xu et al., 2019) utilizes a local similarity matrix to maintain the structural integrity of the feature space, thereby converting discrete labels into soft labels. Zhang et al. (Zhang et al., 2021b) implemented a label propagation technique to disseminate labeling-importance information throughout the network, facilitating the generation of soft pseudo-labels. The manifold-based LE (Hou et al., 2016) leverages the locally linear embedding technique to derive soft pseudo-labels. Low-rank representation LE method (Tang et al., 2020) captures the global relationships among samples and predicts implicit label correlations for generating soft pseudo-labels. Furthermore, Zhu et al. (Zhu et al., 2020) incorporate both the structural relationships between instances and privileged information to generate soft pseudo-labels.

Soft pseudo-labels are commonly employed to serve as a kind of supervision refinement technique in weakly-supervised learning, such as partial label learning (PLL) (Lv et al., 2020; Zhang & Yu, 2015) where the correct label is hidden in candidate label set. Yao et al. (Yao et al., 2020) generate soft pseudo-labels by assembling the model predictions of different epochs and regard them as the auxiliary supervision information for the next epoch. Progressive identification-based methods (Lv et al., 2020; Feng et al., 2020) normalize the softmax output on candidate labels to create soft pseudo-labels. Xu et al. (Xu et al., 2021) treat the soft pseudo-label as a latent variable and utilize variational inference to approximate the poster of the soft labels. Wang et al. (Wang et al., 2022) generate soft pseudo-labels by assessing the similarity between contrastive embeddings and each class prototype. Meanwhile, the consistency-based method (Wu et al., 2022) proposes bi-level optimization for generating soft pseudo-label. In the further analysis of our experiments, we will demonstrate that our framework could deal with weakly-supervised data by taking the PLL experiment as an example.

3. Proposed Method

3.1. Preliminaries

First of all, we briefly introduce some necessary notations. Let $\mathcal{X} = \mathbb{R}^q$ be the q-dimensional instance space and $\mathcal{Y} = \{1, 2, ..., c\}$ be the label space with c class labels. Given the training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)|1 \leq i \leq n\}$ where $\boldsymbol{x}_i$ denotes the q-dimensional instance and $y_i \subseteq \mathcal{Y}$ denotes the ground-truth label associated with $\boldsymbol{x}_i$. In this paper, we aim to find a multi-class classifier $f : \mathcal{X} \mapsto \Delta^{c-1}$ in the function class $\mathcal{F}$ with the training set $\mathcal{D}$, where Δ^{c-1} represents the c-dimensional simplex. For each training example $(\boldsymbol{x}_i, y_i)$, we use the logical label vector $\boldsymbol{l}_i = [l_i^1, l_i^2, \dots, l_i^c]^\top \in \{0,1\}^c$ to represent whether j is the ground-truth label, i.e., $l_i^j = 1$ if $j = y_i$, otherwise $l_i^j = 0$. The soft pseudo-label of $\boldsymbol{x}_i$ is denoted by $\boldsymbol{s}_i = [s_i^1, s_i^2, \dots, s_i^c]^\top \in [0,1]^c$ where $\sum_{j=1}^c s_i^j = 1$. Then $\mathbf{L} = [\boldsymbol{l}_1, \boldsymbol{l}_2, \dots, \boldsymbol{l}_n]$ and $\mathbf{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \dots, \boldsymbol{s}_n]$ represent the initial label matrix and soft pseudo-label matrix, respectively.

3.2. Overview

SEAL alternately trains the predictive model and the soft-pseudo-label generator guided by a meta-network-parameterized objective function. The meta-network dynamically adjusts the parameters of the objective function used for a soft-pseudo-label generator. This optimization is informed by feedback from both the predictive model and the soft-pseudo-label generator, based on their performance in the learning task, creating a feedback loop that refines the training process. Additionally, our framework offers versatility across different learning tasks by allowing direct modifications to the task loss.

SEAL forges a link between the predictive model's parameters and the meta-network's parameters by utilizing the soft-pseudo-label generator as a conduit within the meta-learning framework at each iteration. The process is initiated by cloning the parameters from both the predictive model and soft-pseudo-label generator only for meta-learning purposes. Subsequently, the meta-network optimizes the duplicate parameters of the pseudo-label model, making the updated duplicate parameters depend on the meta-network's parameters. This is followed by optimizing the predictive model's parameters using the soft pseudo-labels produced by the soft-pseudo-label generator by with the updated duplicate parameters, thereby making the predictive model's updated duplicate parameters also depend on the meta-network's parameters. The task loss then evaluates the output of the predictive model using the updated duplicate parameters, and the gradient is backpropagated towards the meta-network. Upon finishing the meta-learning step, the refined meta-network is further employed to optimize the soft-pseudo-label generator to improve the predictive model.

3.3. The SEAL Framework

To train the predictive DNNs f parameterized by $\boldsymbol{\phi}$, we minimize the following empirical risk estimator $\widehat{R}(f)$ by leveraging the original label $\boldsymbol{l}_i$ and soft pseudo-label $\boldsymbol{s}_i$ of each instance $\boldsymbol{x}_i$:

$$\widehat{R}(f) = \frac{1}{n}\sum_{i=1}^{n}\left(\ell(f(\boldsymbol{x}_i;\boldsymbol{\phi}), \boldsymbol{l}_i) + \lambda\ell(f(\boldsymbol{x}_i;\boldsymbol{\phi}), \boldsymbol{s}_i)\right), \quad (1)$$

where ℓ is a cross-entropy function, and the multiplicative factor λ is used to balance the supervision signal provided by the original label $\boldsymbol{l}_i$ and soft pseudo-label $\boldsymbol{s}_i$. The soft pseudo-label $\boldsymbol{s}_i$ is generated by a soft-pseudo-label generator g parameterized by $\boldsymbol{\psi}$, i.e.,

$$\boldsymbol{s}_i = g(\boldsymbol{x}_i;\boldsymbol{\psi}). \quad (2)$$

where the architecture of g is the same as f.

Instead of manually designing an explicit loss for a classifier to output soft pseudo-labels, SEAL proposed a meta-network-parameterized objective function for training the soft-pseudo-label generator g to align with the target of soft-pseudo-label generation. The parameterized objective function is represented by h parameterized by a meta-network $\boldsymbol{\omega}$ (Bechtle et al., 2020). Then the soft-pseudo-label generator g is optimized by minimizing

$$\mathcal{L}(g) = \frac{1}{n}\sum_{i=1}^{n} h(g(\boldsymbol{x}_i;\boldsymbol{\psi}), \boldsymbol{l}_i;\boldsymbol{\omega}). \quad (3)$$

SEAL alternately trains the predictive model f, the soft-pseudo-label generator g, and the parameterized objective h in a meta-learning process.

Firstly, we begin with copying the parameters of the predictive model and soft-pseudo-label generator with $\boldsymbol{\phi}' = \boldsymbol{\phi}$ and $\boldsymbol{\psi}' = \boldsymbol{\psi}$, which are only used for the meta-learning process. Then, we optimize the parameters $\boldsymbol{\psi}'$ of the soft-pseudo-label generator g via using the following parametric objective function $\mathcal{L}^{\mathrm{m}}$:

$$\mathcal{L}^{\mathrm{m}}(g(\mathbf{X};\boldsymbol{\psi}'),\mathbf{L})=\frac{1}{n}\sum_{i=1}^{n}h(g(\boldsymbol{x}_i;\boldsymbol{\psi}'),\boldsymbol{l}_i;\boldsymbol{\omega}),\tag{4}$$

After the parameter $\boldsymbol{\psi}'$ is optimized to the optimal parameter $\boldsymbol{\psi}'^{\star}$, $\boldsymbol{\psi}'^{\star}$ will always depend on the parameters $\boldsymbol{\omega}$ of the meta-network h, which we could explicitly express the dependency as $\boldsymbol{\psi}'^{\star}(\boldsymbol{\omega})$.

Next, let $\mathbf{S}'=[\boldsymbol{s}'_1,\boldsymbol{s}'_2,\ldots,\boldsymbol{s}'_n]$ denote soft pseudo-labels generated by the soft-pseudo-label generator $g(\cdot;\boldsymbol{\psi}'^{\star})$ according to Eq. (2). We optimize the parameters $\boldsymbol{\phi}'$ of the predictive model f via the loss function $\mathcal{L}^{\mathrm{pl}}$ as follows:

$$\mathcal{L}^{\mathrm{pl}}(f(\mathbf{X};\boldsymbol{\phi}'),\mathbf{S}')=\frac{1}{n}\sum_{i=1}^{n}\ell^{\mathrm{pl}}(f(\boldsymbol{x}_i;\boldsymbol{\phi}'),\boldsymbol{s}'_i),\tag{5}$$

where ℓ^{pl} serves as a bridge between the predictive model f and soft-pseudo-label generator g. Note that $\mathcal{L}^{\mathrm{pl}}(f(\mathbf{X};\boldsymbol{\phi}'),\mathbf{S}'))$ also perform back-propagation on $\mathbf{S}'$, which allows the optimal parameters $\boldsymbol{\phi}'^{\star}$ after updating to be dependent on the parameters $\boldsymbol{\psi}'$ of the soft-pseudo-label generator g, i.e., $\boldsymbol{\phi}'^{\star}(\boldsymbol{\psi}'^{\star})$, and further dependent on the parameters $\boldsymbol{\omega}$ of the meta-network h, i.e., $\boldsymbol{\phi}'^{\star}(\boldsymbol{\psi}'^{\star}(\boldsymbol{\omega}))$.

Finally, upon finishing the optimization of the parameters $\boldsymbol{\phi}'$ of the predictive model f', we use the following task loss $\mathcal{L}^{\mathrm{t}}$ to optimize the parameters $\boldsymbol{\omega}$ of the meta-network h:

$$\mathcal{L}^{\mathrm{t}}(f(\mathbf{X};\boldsymbol{\phi}'^{\star}),\mathbf{L})=\frac{1}{n}\sum_{i=1}^{n}\ell^{\mathrm{t}}(f'(\boldsymbol{x}_i;\boldsymbol{\phi}'^{\star}),\boldsymbol{l}_i),\tag{6}$$

where $\ell^{\mathbf{t}}$ is a specified loss function to some specified task.

Overall, the meta-learning objective can be formulated as the optimization problem as follows:

$$\begin{aligned}&\min_{\boldsymbol{\omega}}\mathcal{L}^{\mathrm{t}}(f'(\mathbf{X};\boldsymbol{\phi}'^{\star}),\mathbf{L})\\ \text{s.t.}\quad&\boldsymbol{\phi}'^{\star}=\arg\min_{\boldsymbol{\phi}'}\mathcal{L}^{\mathrm{pl}}(f'(\mathbf{X};\boldsymbol{\phi}'),\mathbf{S}')\\ &\boldsymbol{\psi}'^{\star}=\arg\min_{\boldsymbol{\psi}'}\mathcal{L}^{\mathrm{m}}(g'(\mathbf{X};\boldsymbol{\psi}'),\mathbf{L})\end{aligned}\tag{7}$$

Here, to solve the optimization of Eq. (7), we adopt a batch-style strategy to update $\boldsymbol{\psi}'$, $\boldsymbol{\phi}'$ and $\boldsymbol{\omega}$ through a single optimization loop, respectively, to guarantee the efficiency of the algorithm. We employ stochastic gradient descent (SGD) optimization to optimize Eq. (4) (5) and (6). Specifically, in each iteration k of training, the training set $\mathcal{D}$ is shuffled into I mini-batches, containing m training samples $\{(\boldsymbol{x}_i,\boldsymbol{y}_i)|1\leq i\leq m\}$. Firstly, the updating equation of the soft-pseudo-label generator parameters $\boldsymbol{\psi}'^{[k-1]}$ can be formulated by moving along the direction of the objective loss in Eq. (4) into the new $\boldsymbol{\psi}'^{[k]}$ on a mini-batch training data as follows:

$$\begin{aligned}\boldsymbol{\psi}'^{[k]}&=\boldsymbol{\psi}'^{[k-1]}\\&\quad-\frac{\alpha}{m}\sum_{i=1}^{m}\frac{\partial h(g'(\boldsymbol{x}_i;\boldsymbol{\psi}'^{[k-1]}),\boldsymbol{l}_i;\boldsymbol{\omega}^{[k-1]})}{\partial\boldsymbol{\psi}'^{[k-1]}},\end{aligned}\tag{8}$$

where α is the step size. Secondly, the updating equation for the predictive model parameters $\boldsymbol{\phi}'^{[k-1]}$ can be expressed by adjusting them in the direction of the objective loss in Eq. (5) on a mini-batch of training data, resulting in the new parameter set $\boldsymbol{\phi}'^{[k]}$, as follows:

$$\boldsymbol{\phi}'^{[k]}=\boldsymbol{\phi}'^{[k-1]}-\frac{\beta}{m}\sum_{i=1}^{m}\frac{\partial\ell^{\mathrm{pl}}(f'(\boldsymbol{x}_i;\boldsymbol{\phi}'^{[k-1]}),\boldsymbol{s}'_i)}{\partial\boldsymbol{\phi}'^{[k-1]}},\tag{9}$$

where β is the step size and $\boldsymbol{s}'_i=g(\boldsymbol{x}_i;\boldsymbol{\psi}'^{[k]})$. Thirdly, the iterative update for the predictive model parameters $\boldsymbol{\omega}^{[k-1]}$ can be expressed by aligning them with the direction of the objective loss in Eq. (6) on a mini-batch of training data, yielding the refined parameter set $\boldsymbol{\omega}^{[k]}$, as outlined below:

$$\boldsymbol{\omega}^{[k]}=\boldsymbol{\omega}^{[k-1]}-\frac{\kappa}{m}\sum_{i=1}^{m}\frac{\partial\ell^{\mathrm{t}}(f'(\boldsymbol{x}_i;\boldsymbol{\phi}'^{[k-1]}),\boldsymbol{l}_i)}{\partial\boldsymbol{\omega}^{[k-1]}},\tag{10}$$

where κ is the step size. Finally, we update the soft-pseudo-label generator parameters $\boldsymbol{\psi}^{[k-1]}$ via the updated meta-network h on a mini-batch training data as follows:

$$\begin{aligned}\boldsymbol{\psi}^{[k]}=&\boldsymbol{\psi}^{[k-1]}\\&-\frac{\alpha}{m}\sum_{i=1}^{m}\frac{\partial h(g(\boldsymbol{x}_i;\boldsymbol{\psi}^{[k-1]}),\boldsymbol{l}_i;\boldsymbol{\omega}^{[k]})}{\partial\boldsymbol{\psi}^{[k-1]}},\end{aligned}\tag{11}$$

where we keep the step size the same as that in Eq. (8).

In this way, as the soft-pseudo-label generator parameters $\boldsymbol{\psi}$ and meta-network parameters $\boldsymbol{\omega}$ are updated iteratively, the soft pseudo-labels $\mathbf{S}$ produced by the soft-pseudo-label generator g is also refined to contribute to the training of the predictive model step by step. The algorithmic description of SEAL is presented in Algorithm 1.

3.4. Practical Implementation

Gradual increase of λ. In the risk estimator Eq. (1), we use a balancing factor λ during the whole training procedure, which is suggested not to be fixed. At the beginning of the training phase, the meta-network h is not prepared to train the soft-pseudo-label generator g to produce reliable soft pseudo-labels, which will lead to a performance drop of the classifier. Hence, we apply a linear ramp-up function to increase λ from a small weight to the given λ_e during the first T' epochs:

$$\lambda(t)=\min\{\frac{t}{T'}\lambda_e,\lambda_e\}.\tag{12}$$

Algorithm 1 SEAL Algorithm

Require: The training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{l}_i) | 1 \leq i \leq n\}$, epoch T, iteration I;
for $t = 1, \ldots, T$ **do**
  Shuffle the training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{l}_i) | 1 \leq i \leq n\}$ into I mini-batches;
  for $k = 1, \ldots, I$ **do**
    Copy the parameters of the predictive model and soft-pseudo-label generator with $\boldsymbol{\phi}'^{[k-1]} = \boldsymbol{\phi}^{[k-1]}$ and $\boldsymbol{\psi}'^{[k-1]} = \boldsymbol{\psi}^{[k-1]}$;
    Update $\boldsymbol{\psi}'^{[k-1]}$ to $\boldsymbol{\psi}'^{[k]}$ by Eq. (8);
    Update $\boldsymbol{\phi}'^{[k-1]}$ to $\boldsymbol{\phi}'^{[k]}$ by Eq. (9);
    Update $\boldsymbol{\omega}^{[k-1]}$ to $\boldsymbol{\omega}^{[k]}$ by Eq. (10);
    Update $\boldsymbol{\psi}^{[k-1]}$ to $\boldsymbol{\psi}^{[k]}$ by Eq. (11);
    Obtain the soft pseudo-labels $\boldsymbol{s}_i$ for each example $\boldsymbol{x}_i$ by Eq. (2);
    Update the predictive model parameters $\boldsymbol{\phi}$ by forward computation and back-propagation with the empirical risk estimator in Eq. (1);
  end for
end for
Ensure: The predictive model $f(\cdot; \boldsymbol{\phi})$.

This dynamic strategy gradually makes the second term in Eq. (1) with soft pseudo-labels dominate in the whole risk estimator, as the meta-network h is being trained during the first T' epochs,

Chioces of ℓ^{pl}. We use the bidirectional Kullback-Leibler (KL) divergence loss as ℓ^{pl} to optimize $\boldsymbol{\phi'}$ in Eq. (5):

$$\begin{aligned}\ell^{\text{pl}}(f(\boldsymbol{x}_i; \boldsymbol{\phi}'), \boldsymbol{s}'_i) = \mathrm{KL}(\boldsymbol{s}'_i || f(\boldsymbol{x}_i; \boldsymbol{\phi}')) \\ + \mathrm{KL}(f(\boldsymbol{x}_i; \boldsymbol{\phi}') || \boldsymbol{s}'_i),\end{aligned} \tag{13}$$

Chioces of ℓ^{t}. For supervised learning, considering the top-k prediction, we use the following loss as the task loss ℓ^{t} to optimize $\boldsymbol{\omega}$ in Eq. (6):

$$\begin{aligned}&\ell^{\text{t}}(f(\boldsymbol{x}_i; \boldsymbol{\phi}'^{\star}), \boldsymbol{l}_i) = \\ &- \log\Big(\sum_{k=1}^{c} P(j)\big(\sum_{j=1}^{k} Q_{j,y_i}(f(\boldsymbol{x}_i; \boldsymbol{\phi}'^{\star}))\big)\Big),\end{aligned} \tag{14}$$

where $P(j)$ is the given prior of the top-j, and Q_{j,y_i} is the predicted probability of y_i being the j-th best prediction for $\boldsymbol{x}_i$. We employ the library provided by (Petersen et al., 2022) to estimate Q_{j,y_i}.

SEAL is flexible to learning tasks as the task loss designed for the supervised learning problems can be modified directly. For example, we could utilize the following loss via considering the consistency between the feature and label spaces (Wu et al., 2022) as the task loss ℓ^{t} in SEAL to deal with partial label learning (Zhang et al., 2017) :

$$\ell^{\mathbf{t}}(f(\boldsymbol{x}_i; \boldsymbol{\phi}'^{\star}), \boldsymbol{l}_i) = \sum_{k=1}^{K} \mathrm{KL}(\boldsymbol{u}_i || f(\boldsymbol{x}_i^k; \boldsymbol{\phi}'^{\star})), \tag{15}$$

where $\{\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^K\}$ denotes a K-augmentation set for the instance $\boldsymbol{x}_i$, and $\boldsymbol{u}_i = [u_{i1}, u_{i2}, \ldots, u_{ic}]$ with

$$u_{ij} = \frac{(\prod_{k=1}^{K} f_j(\boldsymbol{x}_i^k; \boldsymbol{\phi}'^{\star}))^{\frac{1}{K}}}{\sum_{o=1}^{c} (\prod_{k=1}^{K} f_o(\boldsymbol{x}_i^k; \boldsymbol{\phi}'^{\star}))^{\frac{1}{K}}}. \tag{16}$$

3.5. Theoretical Analysis

We will theoretically demonstrate that the soft pseudo-labels could enable the predictive model to achieve a larger sample margin for classes, which results in a more robust and tight generalization of the predictive model when trained using soft pseudo-labels.

Let $\mathbf{Z} = [\boldsymbol{z}_1^\top, \boldsymbol{z}_2^\top, \ldots, \boldsymbol{z}_n^\top] \in \mathbb{R}$ denote the feature matrix extracted by the backbone of the classifier, $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1^\top, \boldsymbol{\theta}_2^\top, \ldots, \boldsymbol{\theta}_c^\top]$ represent the parameters of the last classifier layer, implemented with a linear layer. Let $P_j(\boldsymbol{x})$ denote the class-conditional distribution, i.e., $P_j(\boldsymbol{x}) = P(\boldsymbol{x}|y = j)$, and $[j] = \{i | y_i = j\}$ denote the sample indices corresponding to class j. We define the margin for a class j as follows:

$$\gamma_j = \min_{i \in [j]} \boldsymbol{\theta}_j^\top \boldsymbol{z}_i - \max_{j' \neq j} \boldsymbol{\theta}'^\top_j \boldsymbol{z}_i. \tag{17}$$

Let $G(\mathcal{F}, \mathcal{X}, \mathcal{Y})$ denote the generalization error bound induced from the class margin γ (Cao et al., 2019). Then under the assumption about the extracted features in Appendix A, we could obtain the following theorem:

Theorem 3.1. *Let $\boldsymbol{\Theta}$ be the parameters of the last model layer trained by Gradient Descend starting from random initialization. Suppose that at epoch T, $\forall k \in \mathcal{Y}$ with $k \neq p$, $\boldsymbol{\theta}_k$ arrives at the optimal prototype $\boldsymbol{v}_k$. Then, we fix the extracted features $\{\boldsymbol{z}_i | 1 \leq i \leq n\}$ and the prototype except $\boldsymbol{\theta}_p$ to continue the training process for $\boldsymbol{\theta}_p$. Let $G^s(\mathcal{F}, \mathcal{X}, \mathcal{Y}), G^h(\mathcal{F}, \mathcal{X}, \mathcal{Y})$ denote the generalization error bound based on the class margin derived from the empirical risk estimators with soft pseudo-labels and initial hard labels, respectively. At epoch $T' > T$, we could have*

$$G^s(\mathcal{F}, \mathcal{X}, \mathcal{Y}) < G^h(\mathcal{F}, \mathcal{X}, \mathcal{Y}).$$

The proof of Theorem 3.1 is provided in Appendix A. Theorem 3.1 shows that the expected generalization error of the model trained on proper soft pseudo-labels could be bounded by a smaller upper bound than the model trained on initial hard labels.

Table 1. Classification accuracy of each comparing approach (mean ± std). The best performance is shown in boldface. •/○ indicates whether the performance of SEAL is statistically superior/inferior to the comparing algorithm on each dataset (pairwise t-test at 0.05 significance level).

Methods	CIFAR-10	CIFAR-100	TinyImageNet
CE	$91.36 \pm 0.33\%$•	$68.11 \pm 0.45\%$•	$54.28 \pm 0.27\%$•
LSR	$91.62 \pm 0.28\%$	$68.96 \pm 0.60\%$•	$54.74 \pm 0.28\%$•
TF-KD$_{self}$	$91.66 \pm 0.04\%$	$70.44 \pm 0.26\%$	$54.34 \pm 0.34\%$•
TF-KD$_{reg}$	$91.14 \pm 0.35\%$•	$69.47 \pm 0.56\%$•	$54.11 \pm 0.37\%$•
CS-KD	$91.38 \pm 0.14\%$•	$68.63 \pm 0.17\%$•	$55.67 \pm 0.58\%$•
PS-KD	$91.23 \pm 0.22\%$•	$68.39 \pm 0.81\%$•	$54.35 \pm 0.75\%$•
DLB	$91.29 \pm 0.23\%$•	$68.48 \pm 0.40\%$•	$53.61 \pm 0.19\%$•
USKD	$91.47 \pm 0.15\%$•	$69.33 \pm 0.66\%$•	$55.33 \pm 0.32\%$•
SEAL	$\mathbf{91.79 \pm 0.11\%}$	$\mathbf{70.72 \pm 0.17\%}$	$\mathbf{56.76 \pm 0.22\%}$

4. Experiments

4.1. Datasets

We employ three benchmark datasets for multi-class classification including CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and TinyImageNet (Le & Yang, 2015), to evaluate the proposed approach. Both CIFAR-10 and CIFAR-100 consist of a total of $60,000$ images with resolutions of 32×32 pixels, distributed across 10 classes and 100 classes, respectively. The TinyImageNet dataset is a subset of ILSVRC-2012 (Russakovsky et al., 2015) and contains 200 classes. Each class is represented by 500 training samples and 50 testing samples, all resized to 64×64. In our preprocessing routine, training images from all datasets underwent random cropping and resizing to a uniform 32×32 pixel format post-normalization, whereas test images were solely normalized. We allocated 10% of the training data from each dataset for validation purposes.

4.2. Baselines

We compare the performance of SEAL with seven soft-pseudo-label approaches:

- LSR (Szegedy et al., 2016): A label-smoothing regularization approach which replaces the one-hot encoded true label with a smoothed distribution that assigns some probabilities to incorrect labels.
- TF-KD$_{self}$ (Yuan et al., 2020): A self-training knowledge distillation approach which trains the student model in a normal way to obtain a pre-trained model and uses it to generate soft labels to train itself.
- TF-KD$_{reg}$ (Yuan et al., 2020): A teacher-free knowledge distillation approach which uses manually designed soft labels that the probability of a correct class is much higher than that of an incorrect one.
- CS-KD (Yun et al., 2020): A class-wise self-distillation approach which matches or distills the predictive distribution of the model between different samples of the same label.
- PS-KD (Kim et al., 2021): A progressive self-distillation approach which adjusts the training targets progressively by combining the ground-truth and past predictions from the model itself.
- DLB (Shen et al., 2022): An efficient self-distillation approach which distills the on-the-fly generated smooth labels in the previous iteration after rearranging the sampling sequence.
- USKD (Yang et al., 2023): A self-distillation approach generates customized soft labels for both target and non-target classes without a teacher.

In addition, we also compare the performance of SEAL against CE, which directly uses initial binary labels to train the model with cross-entropy loss.

We employ ResNet-20, ResNet-44 and ResNet-18 for CIFAR-10, CIFAR-100 and TinyImageNet as backbone respectively. For the meta-network, we follow (Bechtle et al., 2020; Raymond et al., 2023) and use a small feedforward neural network with two hidden layers and 40 hidden units. Smooth leaky ReLU activations are used in the hidden layers and smooth Softplus activation is used in the output layer. We configure the total number of epochs as 200 and set the batch size to 128. We employ the SGD optimizer with a momentum of 0.9 and a weight decay of 1×10^{-4}, where the initial learning rate is established at 0.1 with a decay factor of 10%. Additionally, we incorporate standard data augmentation techniques, including Random Horizontal Flipping and Random Cropping.

4.3. Experimental Results

Table 1 shows the classification accuracy of each comparative approach for different benchmark datasets. We perform 5 trials with different random seeds and the reported metrics

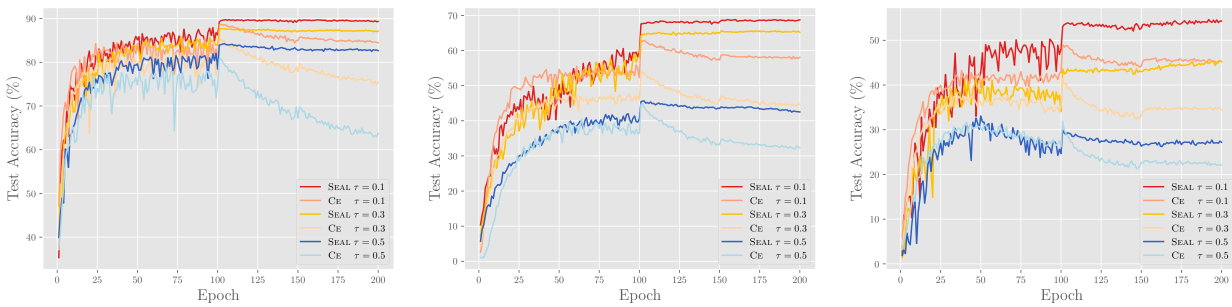

Figure 1. The test accuracy of SEAL and CE on corrupted benchmark datasets with noisy rate $\tau \in \{0.1, 0.3, 0.5\}$

include both the mean and standard deviation. The best performance is shown in boldface. •/◦ indicates whether the performance of SEAL is statistically superior/inferior to the comparing algorithm on each dataset (pairwise t-test at 0.05 significance level). As shown in table 1, it is impressive to observe that:

- SEAL achieves the best performance against all the comparing approaches on all benchmark datasets.
- SEAL achieves superior or at least comparable performance to other approaches on `CIFAR-10` and `CIFAR-100`.
- SEAL achieves superior to other approaches on `TinyImageNet` and exceeds the performance of the second-best algorithm by 1.09%.
- With the growing complexity of datasets, our approach demonstrates progressively superior performance.

Additionally, to prove the robustness to label noise of our method, we manually corrupt the benchmark datasets with symmetric noisy labels under the noisy ratio $\tau \in \{0.1, 0.3, 0.5\}$. Subsequently, for each selected training instance, we replace its correct label with another possible label to create a noisy label. Figure 1 illustrates the curves of test accuracy comparing the baseline CE with SEAL under different noisy rate τ on the benchmark datasets. It can be observed from the figure that SEAL achieves higher test accuracy against CE in all scenarios and the performance of SEAL is robust in the last half of the training stage. These observations demonstrate that SEAL is robust to label noise.

4.4. Further Analysis

4.4.1. Ablation and Convergence Study

To show the helpfulness of the meta-network-parameterized objective function in SEAL, the vanilla variant of SEAL, i.e., SEAL-NM, is adopted to conduct the ablation study. For SEAL-NM, the meta-network-parameterized objective function is replaced by the cross-entropy loss. Table 2 shows that SEAL achieves superior performance to SEAL-NM on all datasets, which demonstrates the usefulness of the meta-network-parameterized objective function in SEAL for improving the performance of the classifier.

Besides, Figure 2 illustrates the soft pseudo-labels generated by soft-pseudo-label generator in SEAL converges with the number of epochs on `CIFAR-100`, which shows that the the soft pseudo-labels in SEAL could converge efficiently.

Table 2. Classification accuracy (mean $\pm$ std) of SEAL and its variant on the benchmark datasets. The best performance is shown in boldface. •/◦ indicates whether the performance of SEAL is statistically superior/inferior to the comparing algorithm on each dataset (pairwise t-test at 0.05 significance level).

Datasets	SEAL	SEAL-NM
CIFAR-10	$\mathbf{91.79 \pm 0.11\%}$	$91.41 \pm 0.33\%$•
CIFAR-100	$\mathbf{70.72 \pm 0.17\%}$	$68.63 \pm 0.42\%$•
TinyImageNet	$\mathbf{56.76 \pm 0.22\%}$	$55.31 \pm 0.64\%$•

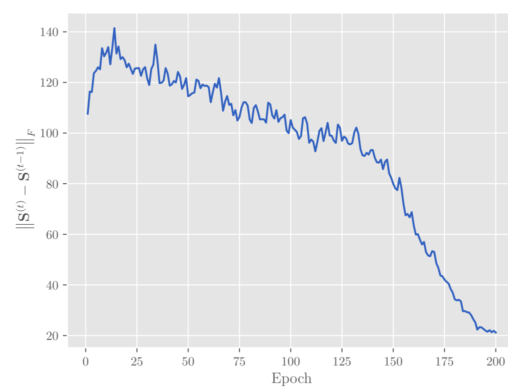

Figure 2. Convergence of the generated soft pseudo-label matrix $\mathbf{S}$ on `CIFAR-100`.

Table 3. Classification accuracy (mean_{std}) of each comparing algorithm in terms of the different proportion of false positive candidate labels. The best performance (the larger the better) is shown in boldface. •/○ indicates whether the performance of SEAL is statistically superior/inferior to the comparing algorithm on each dataset (pairwise t-test at 0.05 significance level).

Methods	CIFAR-10				CIFAR-100		
Flipping Probability	0.3	0.5	0.7	0.9	0.03	0.05	0.1
PRODEN	$91.27_{0.33}$ •	$90.42_{0.14}$ •	$89.26_{0.33}$ •	$84.02_{1.16}$ •	$65.03_{0.69}$ •	$64.28_{0.50}$ •	$64.04_{0.24}$ •
CC	$89.75_{0.50}$ •	$87.57_{0.35}$ •	$84.88_{0.25}$ •	$79.96_{0.54}$ •	$64.75_{0.37}$ •	$63.84_{0.89}$ •	$61.66_{0.24}$ •
VALEN	$89.19_{0.49}$ •	$88.36_{0.30}$ •	$87.29_{0.43}$ •	$44.75_{1.96}$ •	$66.77_{0.99}$ •	$65.97_{0.87}$ •	$65.27_{0.18}$ •
CAVL	$89.23_{3.64}$ •	$88.26_{4.06}$ •	$53.38_{2.81}$ •	$13.19_{2.33}$ •	$57.75_{2.35}$ •	$47.07_{3.19}$ •	$25.83_{1.71}$ •
IDGP	$92.07_{0.32}$ •	$91.04_{0.13}$ •	$88.37_{2.50}$ •	$47.76_{1.51}$ •	$68.19_{0.02}$ •	$67.68_{0.29}$ •	$62.39_{1.74}$ •
PICO	$88.14_{0.25}$ •	$86.00_{0.24}$ •	$76.47_{1.65}$ •	$25.63_{6.25}$ •	$60.50_{0.39}$ •	$58.69_{0.52}$ •	$37.91_{1.56}$ •
PLCR	$90.90_{0.14}$ •	$90.39_{0.21}$ •	$89.31_{0.39}$ •	$81.39_{2.79}$ •	$67.01_{0.33}$ •	$65.78_{0.64}$ •	$62.46_{0.37}$ •
SEAL	$\mathbf{92.93_{0.29}}$	$\mathbf{91.98_{0.28}}$	$\mathbf{89.88_{0.34}}$	$\mathbf{85.36_{0.41}}$	$\mathbf{71.27_{0.40}}$	$\mathbf{68.75_{0.67}}$	$\mathbf{67.21_{1.31}}$

4.4.2. LEARNING WITH PARTIAL LABELS

In this subsection, we utilize SEAL to deal with partial label learning (PLL) (Zhang et al., 2016) by modifying the task loss as Eq. (15). Adopting the same experimental settings in previous PLL work (Lv et al., 2020; Wang et al., 2022; Feng et al., 2020), we manually corrupt `CIFAR-10` and `CIFAR-100` into partially labeled versions by flipping negative labels to false positive labels with the probability $\{0.3, 0.5, 0.7, 0.9\}$ and $\{0.03, 0.05, 0.1\}$, respectively. The performance of SEAL is compared against seven deep PLL methods:

- PRODEN (Lv et al., 2020): A progressive identification approach which approximately minimizes a risk estimator and identifies the true labels in a seamless manner;
- CC (Feng et al., 2020): A classifier-consistent approach which also uses the loss correction strategy to learn the classifier that approaches the optimal one;
- VALEN (Xu et al., 2021): An instance-dependent PLL approach which recovers the latent label distribution via variational inference methods;
- CAVL (Zhang et al., 2022): A progressive identification approach which exploits the class activation value to identify the true label in candidate label sets.
- IDGP (Qiao et al., 2023): A disambiguation approach which builds the model upon a decompositional generation process of candidate labels.
- PICO (Wang et al., 2022): A data-augmentation-based method which identifies the true label via contrastive learning with learned prototypes for image datasets.
- PLCR (Wu et al., 2022): A data-augmentation-based method which identifies the true label via consistency regularization with random augmented instances.

Here, we employ a ResNet-32 as the backbone. The total number of epochs is set to 200. We use SGD optimizer with a momentum of 0.9 and weight decay of 1×10^{-3}. Learning rates are chosen from the orders of magnitude $\{10^{-2}, 10^{-3}, 10^{-4}\}$ guided by the performance on the validation dataset. Common data augmentations are applied, including Random Horizontal Flipping, Random Cropping, Cutout (Devries & Taylor, 2017), and Auto Augment (Cubuk et al., 2019).

Table 3 illustrates the classification accuracy comparing the PLL baselines with SEAL under different flipping probability of negative labels to false positive labels on `CIFAR-10` and `CIFAR-100`. By changing the task loss in our framework SEAL, we successfully adapt to the PLL task and demonstrate significantly superior performances against all the PLL baselines on all settings.

5. Conclusion

In this paper, we propose a novel framework SEAL that trains the predictive model using soft pseudo-labels generated by the specialized soft-pseudo-label generator with a meta-network-parameterized objective. SEAL alternately trains the predictive model and the soft-pseudo-label generator guided by a meta-network-parameterized objective function. The meta-network dynamically adjusts the parameters of the objective function used for a soft-pseudo-label generator. This optimization is informed by feedback from both the predictive model and the soft-pseudo-label generator, based on their performance in the learning task. SEAL is flexible to learning tasks as the task loss designed for the supervised learning problems can be modified directly. Experiments validate the effectiveness of the proposed method.

Acknowledgments

This research was supported by the National Science Foundation of China (62206050, 62125602, and 62076063), China Postdoctoral Science Foundation (2021M700023), Jiangsu Province Science Foundation for

Youths (BK20210220), Young Elite Scientists Sponsorship Program of Jiangsu Association for Science and Technology (TJ-2022-078), and the Big Data Computing Center of Southeast University.

Impact Statement

The potential broader impact of our work might be that the need for accurately annotated data would be significantly reduced. As a result, the rate of unemployment for data annotation specialists might be increased.

References

Algan, G. and Ulusoy, I. Metalabelnet: Learning to generate soft-labels from noisy-labels. *IEEE Transactions on Image Processing*, 31:4352–4362, 2022.

Bechtle, S., Molchanov, A., Chebotar, Y., Grefenstette, E., Righetti, L., Sukhatme, G. S., and Meier, F. Meta learning via learned loss. In *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy*, pp. 4161–4168, 2020.

Cao, K., Wei, C., Gaidon, A., Aréchiga, N., and Ma, T. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems, NeurIPS 2019, Vancouver, BC, Canada*, volume 32, pp. 1565–1576, 2019.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA*, pp. 113–123, 2019.

Devries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

Feng, L., Lv, J., Han, B., Xu, M., Niu, G., Geng, X., An, B., and Sugiyama, M. Provably consistent partial-label learning. In *Advances in Neural Information Processing Systems, NeurIPS 2020, virtual*, volume 33, pp. 10948–10960, 2020.

Ghoshal, A., Chen, X., Gupta, S., Zettlemoyer, L., and Mehdad, Y. Learning better structured representations using low-rank adaptive label smoothing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria*, 2021.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Hou, P., Geng, X., and Zhang, M. Multi-label manifold learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA*, pp. 1680–1686, 2016.

Kim, K., Ji, B., Yoon, D., and Hwang, S. Self-knowledge distillation with progressive refinement of targets. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada*, pp. 6547–6556, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems, NeurIPS 2018, Montréal, Canada*, volume 31, pp. 8168–8177, 2018.

Li, Y., Zhang, M., and Geng, X. Leveraging implicit relative labeling-importance information for effective multi-label learning. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA*, pp. 251–260, 2015.

Liang, J., Li, L., Bing, Z., Zhao, B., Tang, Y., Lin, B., and Fan, H. Efficient one pass self-distillation with zipf's label smoothing. In *17th European Conference on Computer Vision, ECCV 2022, Tel Aviv, Israel*, volume 13671, pp. 104–119, 2022.

Lukov, T., Zhao, N., Lee, G. H., and Lim, S. Teaching with soft label smoothing for mitigating noisy labels in facial expressions. In *17th European Conference on Computer Vision, ECCV 2022, Tel Aviv, Israel*, volume 13672, pp. 648–665, 2022.

Lv, J., Xu, M., Feng, L., Niu, G., Geng, X., and Sugiyama, M. Progressive identification of true labels for partial-label learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Virtual Event*, volume 119, pp. 6500–6510, 2020.

Ngo, N. T., Van, L. N., and Nguyen, T. H. Unsupervised domain adaptation for text classification via meta self-paced learning. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea*, pp. 4741–4752, 2022.

Petersen, F., Kuehne, H., Borgelt, C., and Deussen, O. Differentiable top-k classification learning. In *International Conference on Machine Learning, ICML 2022, Baltimore, Maryland, USA*, volume 162, pp. 17656–17668, 2022.

Qiao, C., Xu, N., and Geng, X. Decompositional generation process for instance-dependent partial label learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda*, 2023.

Raymond, C., Chen, Q., Xue, B., and Zhang, M. Online loss function learning. *CoRR*, abs/2301.13247, 2023.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Shen, Y., Xu, L., Yang, Y., Li, Y., and Guo, Y. Self-distillation from the last mini-batch for consistency regularization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA*, pp. 11933–11942, 2022.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China*, pp. 4322–4331, 2019.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA*, pp. 2818–2826, 2016.

Tang, H., Zhu, J., Zheng, Q., Wang, J., Pang, S., and Li, Z. Label enhancement with sample correlations via low-rank representation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA*, pp. 5932–5939, 2020.

Wang, H., Xiao, R., Li, Y., Feng, L., Niu, G., Chen, G., and Zhao, J. Pico: Contrastive label disambiguation for partial label learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event*, 2022.

Wang, M. and Ma, C. Early stage convergence and global convergence of training mildly parameterized neural networks. In *Advances in Neural Information Processing Systems, NeurIPS 2022, New Orleans, LA, USA*, volume 35, pp. 743–756, 2022.

Wu, D., Wang, D., and Zhang, M. Revisiting consistency regularization for deep partial label learning. In *International Conference on Machine Learning, ICML 2022, Baltimore, Maryland, USA*, volume 162, pp. 24212–24225, 2022.

Xu, N., Liu, Y.-P., and Geng, X. Label enhancement for label distribution learning. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1632–1643, 2019.

Xu, N., Shu, J., Liu, Y., and Geng, X. Variational label enhancement. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Virtual Event*, volume 119, pp. 10597–10606, 2020.

Xu, N., Qiao, C., Geng, X., and Zhang, M. Instance-dependent partial label learning. In *Advances in Neural Information Processing Systems, NeurIPS 2021, virtual*, volume 34, pp. 27119–27130, 2021.

Xu, N., Shu, J., Zheng, R., Geng, X., Meng, D., and Zhang, M. Variational label enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):6537–6551, 2023.

Xu, T. and Liu, C. Data-distortion guided self-distillation for deep neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA*, volume 33, pp. 5565–5572, 2019.

Yang, Z., Zeng, A., Li, Z., Zhang, T., Yuan, C., and Li, Y. From knowledge distillation to self-knowledge distillation: A unified approach with normalized loss and customized soft labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17185–17194, 2023.

Yao, Y., Deng, J., Chen, X., Gong, C., Wu, J., and Yang, J. Deep discriminative CNN with temporal ensembling for ambiguously-labeled image classification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA*, pp. 12669–12676, 2020.

Yi, K. and Wu, J. Probabilistic end-to-end noise correction for learning with noisy labels. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA*, pp. 7017–7025, 2019.

Yuan, L., Tay, F. E. H., Li, G., Wang, T., and Feng, J. Revisiting knowledge distillation via label smoothing regularization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA*, pp. 3902–3910, 2020.

Yun, S., Park, J., Lee, K., and Shin, J. Regularizing class-wise predictions via self-knowledge distillation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA*, pp. 13873–13882, 2020.

Zhang, C., Jiang, P., Hou, Q., Wei, Y., Han, Q., Li, Z., and Cheng, M. Delving deep into label smoothing. *IEEE Transactions on Image Processing*, 30:5984–5996, 2021a.

Zhang, F., Feng, L., Han, B., Liu, T., Niu, G., Qin, T., and Sugiyama, M. Exploiting class activation value for partial-label learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event*, 2022.

Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., and Ma, K. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South)*, pp. 3712–3721, 2019.

Zhang, M. and Yu, F. Solving the partial label learning problem: An instance-based approach. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina*, pp. 4048–4054, 2015.

Zhang, M., Zhou, B., and Liu, X. Partial label learning via feature-aware disambiguation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA*, pp. 1335–1344, 2016.

Zhang, M., Zhang, Q., Fang, J., Li, Y., and Geng, X. Leveraging implicit relative labeling-importance information for effective multi-label learning. *IEEE Transactions on Knowledge and Data Engineering*, 33(5):2057–2070, 2021b.

Zhang, M.-L. and Zhou, Z.-H. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

Zhang, M.-L., Yu, F., and Tang, C.-Z. Disambiguation-free partial label learning. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2155–2167, 2017.

Zhang, Y., Xiang, T., Hospedales, T. M., and Lu, H. Deep mutual learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA*, pp. 4320–4328, 2018.

Zhou, X., Liu, X., Zhai, D., Jiang, J., Gao, X., and Ji, X. Prototype-anchored learning for learning with imperfect annotations. In *International Conference on Machine Learning, ICML 2022, Baltimore, Maryland, USA*, volume 162, pp. 27245–27267, 2022.

Zhu, W., Jia, X., and Li, W. Privileged label enhancement with multi-label learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, Yokohama, Japan*, pp. 2376–2382, 2020.

A. Appendix

We starts with introducing some necessary notations. Let $\mathbf{Z} = [\boldsymbol{z}_1^\top, \boldsymbol{z}_2^\top, \ldots, \boldsymbol{z}_n^\top] \in \mathbb{R}^{2\times n}$ denote the feature matrix extracted by the backbone of the classifier, $\boldsymbol{\Theta}(t) = [\boldsymbol{\theta}_1^\top(t), \boldsymbol{\theta}_2^\top(t), \ldots, \boldsymbol{\theta}_c^\top(t)] \in \mathbb{R}^{a\times c}$ represent the parameters at the t-th epoch of the last classifier layer, implemented with a linear layer. Following (Zhou et al., 2022), we refer to $\boldsymbol{\theta}_j$ as the prototype of the class j. Let $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n]^\top \in \mathbb{R}^{n\times c}$ denote the label matrix for supervision, i.e., $\mathbf{W} = \mathbf{L}$ if we use hard labels, otherwise $\mathbf{W} = \mathbf{S}$. For multi-class learning, we use `softmax` operation and `cross-entropy` loss function. Hence, the empirical risk estimator at the t-th epoch is as follows:

$$\mathcal{L}(t) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} w_i^j \log \frac{e^{\boldsymbol{\theta}_j^\top(t)\boldsymbol{z}_i}}{\sum_{k=1}^{c} e^{\boldsymbol{\theta}_k^\top(t)\boldsymbol{z}_i}}. \tag{18}$$

Let $P_j(\boldsymbol{x})$ denote the class-conditional distribution, i.e., $P_j(\boldsymbol{x}) = P(\boldsymbol{x}|y=j)$, and $[j] = \{i|y_i = j\}$ denote the sample indices corresponding to class j. We define the margin for a class j as follows:

$$\gamma_j = \min_{i\in[j]} \boldsymbol{\theta}_j^\top \boldsymbol{z}_i - \max_{j'\neq j} \boldsymbol{\theta}_j'^\top \boldsymbol{z}_i. \tag{19}$$

Let $\mathcal{L}_{\gamma_j,j}$ denote the margin loss on examples from class j:

$$\mathcal{L}_{\gamma_j,j}[f] = \mathbb{E}_{\boldsymbol{x}\sim P_j(\boldsymbol{x})}\mathbb{I}[\max_{j'\neq j} f_{j'}(\boldsymbol{x}) > f_j(\boldsymbol{x}) - \gamma_j], \tag{20}$$

and $\hat{\mathcal{L}}_{\gamma,j}$ denote its empirical variant:

$$\hat{\mathcal{L}}_{\gamma,j}[f] = \frac{1}{|[j]|}\sum_{i\in[j]} \mathbb{I}[\max_{j'\neq j} f_{j'}(\boldsymbol{x}_i) > f_j(\boldsymbol{x}_i) - \gamma_j]. \tag{21}$$

Here, for convenience, we let $f_j(\boldsymbol{x}_i) = \boldsymbol{\theta}_j^\top \boldsymbol{z}_i$ denote the logits of the model without the `softmax` operation. For a hypothesis class $\mathcal{F}$, let $\hat{\mathfrak{R}}_j(\mathcal{F})$ denote the empirical Rademacher complexity of its class j margin:

$$\hat{\mathfrak{R}}_j(\mathcal{F}) = \frac{1}{|[j]|}\mathbb{E}_\sigma\left[\sup_{f\in\mathcal{F}}\sum_{i\in[j]}\sigma_i[f_j(\boldsymbol{x}_i) - \max_{j'\neq j} f_{j'}(\boldsymbol{x}_i)]\right] \tag{22}$$

where σ is a vector of i.i.d. uniform from $\{+1, -1\}$ bits. According to (Cao et al., 2019), we have the following theorem of class-balanced generalization error bound:

Theorem 1. *With probability $1-\delta$ over the randomness of the training data, for class sample margins $\gamma_1, \ldots, \gamma_c > 0$, let*

$$G(\mathcal{F}, \mathcal{X}, \mathcal{Y} = \frac{1}{c}\sum_{j=1}^{c}\left(\hat{\mathcal{L}}_{\gamma_j,j}[f] + \frac{4}{\gamma_j}\hat{\mathfrak{R}}_j(\mathcal{F}) + \epsilon_j(\gamma_j)\right), \tag{23}$$

where $\epsilon_j(\gamma_j) = \sqrt{\frac{\log\log_2(\frac{2\max_{x\in\mathcal{X},f\in\mathcal{F}}|f(\boldsymbol{x})|}{\gamma_j}) + \log\frac{2c}{\delta}}{|[j]|}}$ is typically a low-order term in $|[j]|$. For all hypotheses $f\in\mathcal{F}$, we will have balanced-class generalization bounded by:

$$P_{\boldsymbol{x},y}[f_y(\boldsymbol{x}) < \max_{j\neq y} f_j(\boldsymbol{x})] \leq G(\mathcal{F}, \mathcal{X}, \mathcal{Y}, \{\gamma_1, ..., \gamma_c\}). \tag{24}$$

Lemma A.1. *Suppose that the γ_p in $\{\gamma_1, ..., \gamma_c\}$ is replace by γ_p' to obtain $G'(\mathcal{F}, \mathcal{X}, \mathcal{Y})$, which satisfies $\gamma_p' > \gamma_p$, then we have $G'(\mathcal{F}, \mathcal{X}, \mathcal{Y}) < G(\mathcal{F}, \mathcal{X}, \mathcal{Y})$.*

Proof. According to the definition of the class margin in Eq. (19), for the first term of Eq. (23) we have

$$\begin{aligned}
\hat{\mathcal{L}}_{\gamma_j,j}[f] &= \frac{1}{|[j]|}\sum_{i\in[j]}\mathbb{I}[\max_{j'\neq j} f_{j'}(\boldsymbol{x}_i) > f_j(\boldsymbol{x}_i) - \gamma_j] \\
&= \frac{1}{|[j]|}\sum_{i\in[j]}\mathbb{I}[f_j(\boldsymbol{x}_i) - \max_{j'\neq j} f_{j'}(\boldsymbol{x}_i) < \min_{i\in[j]}\boldsymbol{\theta}_j^\top\boldsymbol{z}_i - \max_{j'\neq j}\boldsymbol{\theta}_j'^\top\boldsymbol{z}_i] \\
&= \frac{1}{|[j]|}\sum_{i\in[j]}\mathbb{I}[\frac{1}{|[j]|}\sum_{i\in[j]}\mathbb{I}[\min_{i\in[j]}\boldsymbol{\theta}_j^\top\boldsymbol{z}_i - \max_{j'\neq j}\boldsymbol{\theta}_j'^\top\boldsymbol{z}_i < \min_{i\in[j]}\boldsymbol{\theta}_j^\top\boldsymbol{z}_i - \max_{j'\neq j}\boldsymbol{\theta}_j'^\top\boldsymbol{z}_i] \\
&= 0.
\end{aligned} \tag{25}$$

For the second term, the Rademacher complexity $\hat{\mathfrak{R}}_j(\mathcal{F})$ will typically scale as $\sqrt{\frac{C(\mathcal{F})}{|[j]|}}$, which $C(\mathcal{F})$ denotes some complexity measure of $\mathcal{F}$, only related to $\mathcal{F}$ and the number of the instances belonging to class j. Hence, $\frac{4}{\gamma_j}\hat{\mathfrak{R}}_j(\mathcal{F})$ will increase as γ_j decreases. So does the third term $\epsilon_j(\gamma_j)$.

Then, we have

$$\begin{aligned}
G(\mathcal{F},\mathcal{X},\mathcal{Y}) &= \frac{1}{c}\left(\sum_{j\neq p}\hat{\mathcal{L}}_{\gamma_j,j}[f] + \frac{4}{\gamma_j}\hat{\mathfrak{R}}_j(\mathcal{F}) + \epsilon_j(\gamma_j) + \big(\hat{\mathcal{L}}_{\gamma_p,p}[f] + \frac{4}{\gamma_p}\hat{\mathfrak{R}}_p(\mathcal{F}) + \epsilon_p(\gamma_p)\big)\right) \\
&= \frac{1}{c}\left(\sum_{j\neq p}\hat{\mathcal{L}}_{\gamma_j,j}[f] + \frac{4}{\gamma_j}\hat{\mathfrak{R}}_j(\mathcal{F}) + \epsilon_j(\gamma_j) + \big(\frac{4}{\gamma_p}\hat{\mathfrak{R}}_p(\mathcal{F}) + \epsilon_p(\gamma_p)\big)\right) \\
&> \frac{1}{c}\left(\sum_{j\neq p}\hat{\mathcal{L}}_{\gamma_j,j}[f] + \frac{4}{\gamma_j}\hat{\mathfrak{R}}_j(\mathcal{F}) + \epsilon_j(\gamma_j) + \big(\frac{4}{\gamma_p'}\hat{\mathfrak{R}}_p(\mathcal{F}) + \epsilon_p(\gamma_p')\big)\right) \\
&= \frac{1}{c}\left(\sum_{j\neq p}\hat{\mathcal{L}}_{\gamma_j,j}[f] + \frac{4}{\gamma_j}\hat{\mathfrak{R}}_j(\mathcal{F}) + \epsilon_j(\gamma_j) + \big(\hat{\mathcal{L}}_{\gamma_p',p}[f] + \frac{4}{\gamma_p'}\hat{\mathfrak{R}}_p(\mathcal{F}) + \epsilon_p(\gamma_p')\big)\right) \\
&= G'(\mathcal{F},\mathcal{X},\mathcal{Y})
\end{aligned} \tag{26}$$

Hence, we obtain $G'(\mathcal{F},\mathcal{X},\mathcal{Y}) < G(\mathcal{F},\mathcal{X},\mathcal{Y})$, and complete the proof.

We make an assumption on the distribution of the extracted features as follows:

Assumption 1. *For $p \in \mathcal{Y}$, let $\boldsymbol{v}_p = \frac{\frac{1}{|[p]|}\sum_{i\in[p]}\boldsymbol{z}_i}{||\frac{1}{|[p]|}\sum_{i\in[p]}\boldsymbol{z}_i||}$. The extracted features $\{\boldsymbol{z}_i|1\leq i\leq n\}$ satisfy: (1) $\forall 1\leq i\leq n$, $||\boldsymbol{z}_i|| = 1$ and $\forall i,j\in[p]$, $\boldsymbol{z}_i^\top\boldsymbol{z}_j > 0$. (2) $\forall i\in[p]$, $\exists j\in[p]$, $\boldsymbol{z}_j = ||\boldsymbol{z}_j+\boldsymbol{v}_p||\boldsymbol{v}_p - \boldsymbol{z}_i$. (3) $\forall p,q\in\mathcal{Y}$ with $p\neq q$, $\forall i\in[p]$ and $j\in[q]$, $\boldsymbol{v}_p^\top\boldsymbol{z}_i \geq \boldsymbol{v}_q^\top\boldsymbol{z}_i$ and $\boldsymbol{v}_p^\top\boldsymbol{z}_i \geq \boldsymbol{v}_p^\top\boldsymbol{z}_j$. (4) $\forall p\in\mathcal{Y}$, $\forall\boldsymbol{d}\in\Delta^{c-1}$ with $p = \arg\max_k d^k$, $\sum_{i\in[p]}\sum_{j\in[p]} d^m\boldsymbol{z}_i^\top\boldsymbol{z}_j > \sum_{i\notin[p]}\sum_{j\in[p]}(1-d^p)\boldsymbol{z}_i^\top\boldsymbol{z}_j$.*

Here, $\boldsymbol{v}_p$ is a normalized center of the extracted features belongs to class p. (1) means that the extracted feature $\boldsymbol{z}_i$ has been normalized to a hypersphere with a radius of 1 (Zhou et al., 2022), and for two extracted features belongs to the same class, they do not go into opposite directions (Wang & Ma, 2022). (2) means that for each extracted feature of each class p, there is a symmetric point $\boldsymbol{z}_j$ about the vector $\boldsymbol{v}_p$. (3) means that $\boldsymbol{v}_q$ can be considered as the optimal prototype of each class q, and the extracted features $\{\boldsymbol{z}_i|1\leq i\leq n\}$ have the separability (Li & Liang, 2018). (4) is an assumption that the correlation between the extracted features belonging to the same class is stronger than that ones belonging to different classes.

Let $\angle(\cdot,\cdot)$ denote the angle between two vectors. Let $o_p = \arg\min_{i\in[p]}\boldsymbol{v}_p^\top\boldsymbol{z}_i$, and $\boldsymbol{z}_{o_p}$ can be seen as the edge sample of the class p. According to Assumption 1, there also exists $o_p'\in[p]$, such that $\boldsymbol{v}_p^\top\boldsymbol{z}_{o_p} = \boldsymbol{v}_p^\top\boldsymbol{z}_{o_p'}$. Then, we make the following assumption about the prototypes:

Assumption 2. *$\forall k\in\mathcal{Y}$ with $k\neq p$, $\boldsymbol{\theta}_k = \boldsymbol{v}_k$, and the normalized prototype $\boldsymbol{\theta}_p$ satisfy: (1) $||\boldsymbol{\theta}_p|| = 1$; (2) $\angle(\boldsymbol{\theta}_p,\boldsymbol{z}_{o_p}) + \angle(\boldsymbol{\theta}_p,\boldsymbol{v}_p) = \angle(\boldsymbol{z}_{o_p},\boldsymbol{v}_p)$; (3) $\forall q = \arg\min_{q\neq p}\angle(\boldsymbol{v}_p,\boldsymbol{v}_q)$, $\angle(\boldsymbol{v}_q,\boldsymbol{z}_{o_p'}) < \angle(\boldsymbol{v}_q,\boldsymbol{z}_{o_p})$. (4) $\forall q\in\mathcal{Y}$ with $q\neq p$, $\forall i\in[q]$, for $q' = \arg\max_{q'\neq q}\boldsymbol{\theta}_q'^\top\boldsymbol{z}_i$, $p\neq q'$.*

Here, (1) means that the final prototypes used to classify the extracted features are also on a hypersphere with a radius of 1 (Zhou et al., 2022). (2) means that the prototype $\boldsymbol{\theta}_p$ lies between the $\boldsymbol{v}_p$ and $\boldsymbol{z}_{o'_p}$. (3) means that the prototype $\boldsymbol{v}_q$, which is nearest to $\boldsymbol{v}_p$, is closer to $\boldsymbol{z}_{o'_p}$ than $\boldsymbol{z}_{o_p}$. (4) means that the change of $\boldsymbol{\theta}_p$ will only have effect on γ_p.

Then, we could obtain the following lemma:

Lemma A.2. *Under Assumption 2, suppose that $\boldsymbol{\Theta}$ with $\boldsymbol{\theta}_p$ and $\boldsymbol{\Theta}'$ with $\boldsymbol{\theta}'_p$ satisfy Assumption 2, and $\boldsymbol{\theta}_p^\top \boldsymbol{v}_p - \boldsymbol{\theta}_p'^\top \boldsymbol{v}_p > 0$, we have $\gamma_p - \gamma'_p > 0$.*

Proof. The class margin γ_p and γ'_p could be computed as $\gamma_p = \boldsymbol{\theta}_p^\top \boldsymbol{z}_{o'_p} - \boldsymbol{\theta}_q^\top \boldsymbol{z}_{o'_p}$ and $\gamma'_p = \boldsymbol{\theta}_p'^\top \boldsymbol{z}_{o'_p} - \boldsymbol{\theta}_q^\top \boldsymbol{z}_{o'_p}$ with some $q \neq p$, according to Assumption 2. Then we have:

$$\begin{aligned}\gamma_p - \gamma'_p &= \boldsymbol{\theta}_p^\top \boldsymbol{z}_{o'_p} - \boldsymbol{\theta}_p'^\top \boldsymbol{z}_{o'_p} \\ &= \cos\angle(\boldsymbol{\theta}_p, \boldsymbol{z}_{o'_p}) - \cos\angle(\boldsymbol{\theta}'_p, \boldsymbol{z}_{o'_p}) \\ &= \cos\Big(\angle(\boldsymbol{\theta}_p, \boldsymbol{v}_p) + \angle(\boldsymbol{v}_p, \boldsymbol{z}_{o'_p})\Big) - \cos\Big(\angle(\boldsymbol{\theta}'_p, \boldsymbol{v}_p) + \angle(\boldsymbol{v}_p, \boldsymbol{z}_{o'_p})\Big)\end{aligned} \tag{27}$$

Since $\boldsymbol{\theta}_p^\top \boldsymbol{v}_p - \boldsymbol{\theta}_p'^\top \boldsymbol{v}_p > 0$, we could get $\angle(\boldsymbol{\theta}_p, \boldsymbol{v}_p) < \angle(\boldsymbol{\theta}'_p, \boldsymbol{v}_p)$. Hence, $\gamma_p - \gamma'_p > 0$.

Theorem 2. *Let $\boldsymbol{\Theta}$ be the parameters of the last model layer trained by Gradient Descend starting from random initialization. Suppose that at epoch T, $\forall k \in \mathcal{Y}$ with $k \neq p$, $\boldsymbol{\theta}_k$ arrives at the optimal prototype $\boldsymbol{v}_k$. Then, we fix the extracted features $\{\boldsymbol{z}_i | 1 \leq i \leq n\}$ and the prototype except $\boldsymbol{\theta}_p$ to continue the training process for $\boldsymbol{\theta}_p$. Let $G^s(\mathcal{F}, \mathcal{X}, \mathcal{Y}), G^h(\mathcal{F}, \mathcal{X}, \mathcal{Y})$ denote the generalization error bound based on the class margin derived from the empirical risk estimators with soft pseudo-labels and initial hard labels, respectively. At epoch $T' > T$, we could have*

$$G^s(\mathcal{F}, \mathcal{X}, \mathcal{Y}) < G^h(\mathcal{F}, \mathcal{X}, \mathcal{Y}).$$

.

Proof. $\forall p \in \mathcal{Y}$, the gradient of $\mathcal{L}(t)$ with respect to $\boldsymbol{\theta}_p^\top(t)$ can be derived as follows:

$$\frac{\partial \mathcal{L}(t)}{\partial \boldsymbol{\theta}_p^\top(t)} = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j\neq p} w_i^j \frac{e^{(\boldsymbol{\theta}_p(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}{1+\sum_{k\neq j} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}} - w_i^p \frac{\sum_{k\neq p} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}{1+\sum_{k\neq p} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}\right)\boldsymbol{z}_i^\top \tag{28}$$

Let $\mathcal{L}^h$ and $\mathcal{L}^s$ denote the risk estimators using the hard labels and soft labels, respectively. The gradient difference $\Delta(t)$ with respect to the prototype $\boldsymbol{\theta}_p(t)$ between them can be derived as follows:

$$\begin{aligned}\Delta(t) &= \frac{\partial \mathcal{L}^h(t)}{\partial \boldsymbol{\theta}_p^\top(t)} - \frac{\partial \mathcal{L}^s(t)}{\partial \boldsymbol{\theta}_p^\top(t)} \\ &= \frac{1}{n}\sum_{i=1}^{n}\Big(\sum_{j\neq p}(l_i^j - s_i^j)\frac{e^{(\boldsymbol{\theta}_p(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}{1+\sum_{k\neq j} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}} - (l_i^p - s_i^p)\frac{\sum_{k\neq p} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}{1+\sum_{k\neq p} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}\Big)\boldsymbol{z}_i^\top \\ &= \frac{1}{n}\Big(\sum_{i\in[m]}\Big(\sum_{j\neq p}(l_i^j - s_i^j)\frac{e^{(\boldsymbol{\theta}_p(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}{1+\sum_{k\neq j} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}} - (l_i^p - s_i^p)\frac{\sum_{k\neq p} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}{1+\sum_{k\neq p} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}\Big)\boldsymbol{z}_i^\top \\ &+ \sum_{i\notin[m]}\Big(\sum_{j\neq p}(l_i^j - s_i^j)\frac{e^{(\boldsymbol{\theta}_p(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}{1+\sum_{k\neq j} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}} - (l_i^p - s_i^p)\frac{\sum_{k\neq p} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}{1+\sum_{k\neq p} e^{(\boldsymbol{\theta}_k(t)-\boldsymbol{\theta}_j(t))^\top \boldsymbol{z}_i}}\Big)\boldsymbol{z}_i^\top\Big) \\ &= \frac{1}{n}\Big(\sum_{i\in[p]}(s_i^p - 1)\boldsymbol{z}_i^\top + \sum_{i\notin[p]} s_i^p \boldsymbol{z}_i^\top\Big)\end{aligned} \tag{29}$$

We investigate the difference between the prototype $\boldsymbol{\theta}_p(T')$ and $\boldsymbol{\theta}_p(T)$ when the parameters are trained by Gradient Decent with the step size $\eta > 0$:

$$\begin{aligned}\boldsymbol{\theta}_p(T') &= \boldsymbol{\theta}_p(T'-1) - \eta\frac{\partial\mathcal{L}(T'-1)}{\partial\boldsymbol{\theta}_p(T'-1)} \\ &= \boldsymbol{\theta}_p(T'-2) - \eta\frac{\partial\mathcal{L}(T'-1)}{\partial\boldsymbol{\theta}_p(T'-1)} - \eta\frac{\partial\mathcal{L}(T'-2)}{\partial\boldsymbol{\theta}_p(T'-2)} \\ &= \boldsymbol{\theta}_p(T) - \sum_{r=T}^{T'-1}\eta\frac{\partial\mathcal{L}(r)}{\partial\boldsymbol{\theta}_p(r)}.\end{aligned} \tag{30}$$

Hence, we could obtain:

$$\boldsymbol{\theta}_p(T') - \boldsymbol{\theta}_p(T) = \sum_{r=T}^{T'-1}\eta(-\frac{\partial\mathcal{L}(r)}{\partial\boldsymbol{\theta}_p(r)}). \tag{31}$$

We transpose both sides of Eq. (31) and then right-multiply with the vector $\boldsymbol{v}_p$:

$$\boldsymbol{\theta}_p^\top(T')\boldsymbol{v}_p - \boldsymbol{\theta}_p^\top(T)\boldsymbol{v}_p = \sum_{r=T}^{T'-1}\eta(-\frac{\partial\mathcal{L}(r)}{\partial\boldsymbol{\theta}_p(r)})^\top\boldsymbol{v}_p. \tag{32}$$

Let $\boldsymbol{\theta}_p^h$ and $\boldsymbol{\theta}_p^s$ denote the prototypes updated by $\mathcal{L}^h$ and $\mathcal{L}^s$, respectively. Based on Eq. (32), we could obtain:

$$\boldsymbol{\theta}_p^{s\top}(T')\boldsymbol{v}_p - \boldsymbol{\theta}_p^{h\top}(T')\boldsymbol{v}_p = \sum_{r=T}^{T'-1}\eta(\frac{\partial\mathcal{L}^h(r)}{\partial\boldsymbol{\theta}_p^h(r)} - \frac{\partial\mathcal{L}^s(r)}{\partial\boldsymbol{\theta}_p^s(r)})^\top\boldsymbol{v}_p. \tag{33}$$

Due to that $\boldsymbol{s}_i \in \Delta^{c-1}$, we could obtain the following according to Assumption 1.(4)

$$\boldsymbol{\theta}_p^{s\top}(T')\boldsymbol{v}_p - \boldsymbol{\theta}_p^{h\top}(T')\boldsymbol{v}_p > 0. \tag{34}$$

According to Lemma A.1 and A.2, we have $\gamma_p^{\mathrm{s}} > \gamma_p^{\mathrm{h}}$ and finally obtain $G^{\mathrm{s}}(\mathcal{F},\mathcal{X},\mathcal{Y}) < G^{\mathrm{h}}(\mathcal{F},\mathcal{X},\mathcal{Y})$.