Positive-Unlabeled Learning by Latent Group-Aware Meta Disambiguation (Appendix)

Lin Long[1*] Haobo Wang[1*] Zhijie Jiang[1] Lei Feng[2] Chang Yao[1†] Gang Chen[1] Junbo Zhao[1]
[1] Zhejiang University, China [2] Singapore University of Technology and Design, Singapore
{llong, wanghaobo, zjjjj882, changy, cg, j.zhao}@zju.edu.cn, lfengqaq@gmail.com

A. Additional Clarifications

A.1. Notations

Notation	Description
$\mathcal{D}_{\text{train}}$	Training set consisting of PU data.
$\mathcal{D}_{\text{sup}}$	Support set consisting of a small number of labeled data, disjoint from $\mathcal{D}_{\text{train}}$.
$\mathcal{B}_{\text{train}}$	Training mini-batch sampled from $\mathcal{D}_{\text{train}}$.
$\mathcal{B}_{\text{sup}}$	Support mini-batch sampled from $\mathcal{D}_{\text{sup}}$.
$\tilde{\boldsymbol{y}}$	Pseudo-labels for unlabeled data in $\mathcal{D}_{\text{train}}$.
$f_\theta(\cdot)$	Classifier parameterized with θ. $f_\theta(\boldsymbol{x}) : \mathbb{R}^d \mapsto [0,1]$ indicates the probability for the input $\boldsymbol{x}$ to be positive.
$g_\theta(\cdot)$	Encoder parameterized with θ, which consists of the representation layers from f_θ with a projection head. $g_\theta(\boldsymbol{x}) : \mathbb{R}^d \mapsto \mathbb{R}^{128}$ maps the input $\boldsymbol{x}$ to an embedding of length 128.
$\mathcal{L}(\cdot,\cdot)$	Total loss for model (actual-)training.
$\mathcal{L}_{\text{cls}}(\cdot,\cdot)$	Classification loss for model training or evaluation. In LaGAM, BCE is used in particular.
$\mathcal{L}_{\text{cont}}(\cdot,\cdot)$	Contrastive loss for model (actual-)training.

Table 7. Additional explanations of the notations used in this article.

A.2. Pseudo-Code of LaGAM

As shown in Algorithm 1.

Algorithm 1: Pseudo-code of LaGAM.

Input: Training dataset $\mathcal{D}_{\text{train}}$, support dataset $\mathcal{D}_{\text{sup}}$, classifier f, encoder g, learning rate λ, EMA parameter ϵ, weighting parameter β, number of epochs n.

Initialization: model parameter θ, pseudo-labels $\tilde{\boldsymbol{y}} \leftarrow \mathbf{0}$;
for $1, 2, \ldots, n$ **do**
  perform k-means on $\mathcal{D}_{\text{train}}$ and determine the corresponding cluster center of each sample;
  for *mini-batches* $\mathcal{B}_{train} \subseteq \mathcal{D}_{train}$ **do**
    // embeddings generation
    // Aug_q and Aug_k refer to random augmentations
    $B_q \leftarrow \{\boldsymbol{q}_i = g(\text{Aug}_q(\boldsymbol{x}_i)) | (\boldsymbol{x}_i, y_i) \in \mathcal{B}_{\text{train}}\}$;
    $B_k \leftarrow \{\boldsymbol{k}_i = g(\text{Aug}_k(\boldsymbol{x}_i)) | (\boldsymbol{x}_i, y_i) \in \mathcal{B}_{\text{train}}\}$;
    $A \leftarrow B_q \cup B_k$;
    // 1^{st} backward: Meta-Train
    $\theta' \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}_{\text{cls}}(\mathcal{B}_{\text{train}}, \theta)$;
    // 2^{nd} backward: Label-Update
    $\boldsymbol{\delta} \leftarrow -\nabla_{\tilde{\boldsymbol{y}}} \mathcal{L}_{\text{cls}}(\mathcal{D}_{\text{sup}}, \theta')$;
    // projected gradients generation
    for $i = 1, 2, \ldots, |\mathcal{B}_{train}|$ **do**
      $s_i \leftarrow \mathbb{I}(\delta_i > 0)$;
    end
    // EMA update
    $\tilde{\boldsymbol{y}} \leftarrow \epsilon \tilde{\boldsymbol{y}} + (1-\epsilon)\boldsymbol{s}$;
    // actual training
    $\theta \leftarrow \theta - \lambda \nabla_\theta [\mathcal{L}_{\text{cls}} + \beta \mathcal{L}_{\text{cont}}]$;
  end
end

A.3. Implementation Settings

For fair comparisons, most of our experimental settings remain consistent with [1, 7, 13]. Specifically, our LaGAM is trained for 400 epochs (with 20 epochs of warm-up) on each setup, optimized using momentum SGD with a batch size of 64 and an initial learning rate of 0.001, which decays as training progresses. The temperature τ for contrastive loss in Eq. (2) is set as 0.07. Notably, the encoder g used for embedding generation in contrastive learning shares the same representation layers as the classifier f, followed by a projection head that maps the representations to embeddings of length 128. For convenience, when performing k-means, we uniformly set the number of clusters to 100 for all datasets. Besides, kNN local neighbor smoothing is activated after the 50^{th} epoch, and the number of neighbors is set as 10. We follow [7] for the implementations of the baselines and use mostly default parameters. We will make a sensitivity analysis on the weighting parameter β in Eq. (13) later. More implementation details can be referred to in the source code.

Notably, for all ablation studies and sensitivity analyses, the results are obtained using ResNet-18 as the default back-

*Joint first authors.
†Corresponding author.

bone and a particular set of seeds for eliminating the interference of irrelevant factors. "CIFAR-10" and "CIFAR-100" in the result tables refer to CIFAR-10-1 and CIFAR-100-2, respectively, unless otherwise specified.

B. Theoritical Insights

B.1. Motivation

Challenges. Generally, most of the existing PU learning methods can be divided into two categories, termed cost-sensitive methods [1–3, 5, 12] and sample-selection methods [1, 8, 9, 12]. Cost-sensitive methods initially treat all unlabeled samples as noisy negative ones, followed by estimation bias correction with specific misclassification risks, the goal of which is to downweight the contribution of those false negatives. Among them, Self-PU [1] creatively pioneers the technique of meta-learning for sample-wise loss reweighting, to directly boost the model's generalizability with a golden support set. Despite the promising results, assigned with low weights, the unlabeled positive samples, i.e., the false negatives, actually cannot provide valid supervision signals and thus are not effectively utilized, which greatly weakens the intensity of supervision and might easily cause the model to overfit. Sample-selection methods, as the name suggests, focus on identifying confident positive or negative samples from the unlabeled set by hand-crafted heuristics or standard semi-supervised learning methods, to produce solid supervision signals. However, the accuracy of sample-selection cannot always be promised, where exploiting the model's output confidence scores for selecting reliable samples may lead to confirmation bias or accumulated error, greatly affecting the model's generalizability.

Solutions: from trade-off breaking to semantic understanding. Under such a trade-off, it is evident that previous works are not able to simultaneously satisfy the needs for supervision intensity and generalizability. This phenomenon inspires us to propose the idea of meta-disambiguation, which (i)-ensures generalizability with the same golden support set; and (ii)-allows the false negatives to be actually corrected instead of just being filtered out by directly meta-learning the pseudo-labels. However, from Table 4 we can see that, despite that meta-disambiguation can indeed effectively mitigate the label noise, the overall performance still largely lags behind the supervised counterpart. This result suggests that solving the PU problem not only requires a reasonable and robust label disambiguation strategy, but also a deeper understanding of the underlying semantics within PU data to help the model stay on the correct disambiguation trajectory. This motivates us to conduct more fundamental research on how to produce more semantically discriminative representations for PU data, which ultimately becomes the core idea of LaGAM.

How is the lack of semantics the bottleneck? We would like to exemplify this with a case of human decision. As humans, we are typically unaware of many subconscious reasoning steps involved in solving "simple" binary questions. To answer "Do you like sushi?", we'll consider various factors including texture, taste, and price. Intuitively, we expect our model also to learn such implicit "features" from which the binary output can be easily inferred, namely latent semantics. However, we notice that **the overly coarse-grained** binary labels cannot provide enough supervision that reflects the criteria for classification. As shown in Figure 5, with data distribution being better semantically aligned, the decision boundary naturally becomes more distinct, which explains the effectiveness of LaGAM.

B.2. Meta-Disambiguation from an Influence Function Perspective

To better understand the underlying logic of meta-disambiguation, we provide theoretical derivations to explain the intrinsic principle of meta-disambiguation from the perspective of influence function [6]. The influence function is a powerful tool for analyzing the robustness of a trained model, which studies two essential problems: how would the model's prediction change if a training input were removed (modified)[6]? Intuitively, these two questions investigate how the model makes use of a certain training sample and the quality of a certain training sample, respectively. Therefore, in the context of PU learning, we will focus on the latter, investigating the impacts of applying small perturbations to a sample on the model's output to examine the correctness of the given training sample, thus being able to detect those false negatives and refine them in the right direction. Formally, following [6], given a sample $z_k = (\boldsymbol{x}_k, y_k) \in \mathcal{D}_{\text{train}}$, consider the perturbation $z_k \mapsto z_{k,\Delta}$ where $z_{k,\Delta} = (\boldsymbol{x}_k, y_k + \Delta)$, and let $\hat{\theta}_{z_{k,\Delta}, -z_k}$ be the model parameter trained on the dataset $\mathcal{D}_{\text{train}} \backslash \{z_k\} \cup \{z_{k,\Delta}\}$. To approximate the effect of such a perturbation, define the parameters resulting from moving ε from z_k to $z_{k,\Delta}$ as:

$$\begin{aligned} \hat{\theta}_{\varepsilon, z_{k,\Delta}, -z_k} &= \arg\min_{\theta} [\mathcal{L}_{\text{cls}}(\mathcal{B}_{\text{train}}, \theta) \\ &\quad + \varepsilon \ell(f_\theta(\boldsymbol{x}_k), \tilde{y}_k + \delta) - \varepsilon \ell(f_\theta(\boldsymbol{x}_k), \tilde{y}_k)] \\ &= \theta(\tilde{\boldsymbol{y}} + \varepsilon \boldsymbol{\Delta}), \text{where } \boldsymbol{\Delta}_i = \mathbb{I}(i = k)\Delta \end{aligned} \tag{14}$$

Then we can apply the chain rule to measure how the functions of $\hat{\theta}$ are affected with respect to the perturbation Δ. In particular, according to [6], the influence on the evaluation

loss $\mathcal{L}(\mathcal{B}_{\text{sup}}, \hat{\theta}_{\varepsilon,z_{k,\Delta},-z_k})$ can be approximated by:

$$\begin{aligned}\mathcal{I}_{\text{pert}}^{\top} &= \nabla_{\Delta}\mathcal{L}_{\text{cls}}(\mathcal{B}_{\text{sup}}, \hat{\theta}_{z_{k,\Delta},-z_k})^{\top} \\ &= \left.\frac{\partial\hat{\theta}_{\varepsilon,z_{k,\Delta},-z_k}}{\partial\varepsilon}\right|_{\varepsilon=0} \nabla_{\theta}\mathcal{L}_{\text{cls}}(\mathcal{B}_{\text{sup}}, \hat{\theta}_{\varepsilon,z_{k,\Delta},-z_k})^{\top}/\Delta\end{aligned} \tag{15}$$

where $\left.\frac{\partial\hat{\theta}_{\varepsilon,z_{k,\Delta},-z_k}}{\partial\varepsilon}\right|_{\varepsilon=0}$ can be rewrote as:

$$\begin{aligned}\left.\frac{\partial\hat{\theta}_{\varepsilon,z_{k,\Delta},-z_k}}{\partial\varepsilon}\right|_{\varepsilon=0} &= \left.\frac{\partial\theta(\tilde{\boldsymbol{y}}+\varepsilon\boldsymbol{\Delta})}{\partial\varepsilon}\right|_{\varepsilon=0} \\ &= \lim_{\varepsilon\to 0}\frac{\theta(\tilde{\boldsymbol{y}}+\varepsilon\boldsymbol{\Delta})-\theta(\tilde{\boldsymbol{y}})}{\varepsilon} \\ &= \lim_{\varepsilon\to 0}\frac{\theta(\tilde{\boldsymbol{y}}+\varepsilon\boldsymbol{\Delta})-\theta(\tilde{\boldsymbol{y}})}{\varepsilon\Delta}\Delta \\ &= \frac{\partial\theta(\tilde{\boldsymbol{y}})}{\partial y_k}\Delta\end{aligned} \tag{16}$$

By substituting it back into the original equation, we have:

$$\begin{aligned}\mathcal{I}_{\text{pert}}^{\top} &= \frac{\partial\theta(\tilde{\boldsymbol{y}})}{\partial\tilde{y}_k}\nabla_{\theta}\mathcal{L}_{\text{cls}}(\mathcal{B}_{\text{sup}}, \hat{\theta}_{\varepsilon,z_{k,\Delta},-z_k})^{\top} \\ &= \frac{\partial\mathcal{L}_{\text{cls}}(\mathcal{B}_{\text{sup}}, \hat{\theta}_{\varepsilon,z_{k,\Delta},-z_k})}{\partial\tilde{y}_k}^{\top} = -\delta_k^{\top}\end{aligned} \tag{17}$$

According to the Lagrange's Mean Value Theorem, $\mathcal{I}_{\text{pert}}^{\top}\Delta$ tells us the approximation effect that $z_k \mapsto z_{k,\Delta}$ has on the evaluation loss, which indicates that whether $z_{k,\Delta}$ is a better or worse substitution of z_k. By setting Δ in the direction opposite to $\mathcal{I}_{\text{pert}}$, we can obtain the optimal z_{k,Δ^*} that minimizes the evaluation loss, where the perturbation Δ^* indicates exactly the same updating direction for specific $\tilde{y}_k$ as we have done with δ_k in Eq. (10). Since the new training sample z_{k,Δ^*} that lowers the evaluation loss is more likely to be the sample with a correct label, it is also reasonable for us to believe that the corresponding perturbation δ_k is the right updating direction that better guides $\tilde{y}_k$ towards the ground-truth. By applying such influence function analysis to all samples in $\mathcal{B}_{\text{train}}$, we can obtain the optimal updating direction for each sample, which is equal to directly calculating the gradient of $\mathcal{L}_{\text{cls}}(\mathcal{B}_{\text{sup}}, \theta(\tilde{\boldsymbol{y}}))$ w.r.t. $\tilde{\boldsymbol{y}}$, just the same as how we calculate $\boldsymbol{\delta}$ in Eq. (10).

In implementations, we actually adopt the concept of the influence function, transforming the computation of the gradient of the loss w.r.t. $\tilde{\boldsymbol{y}}$ into the computation of the gradient of the loss w.r.t. a perturbation applied on $\tilde{\boldsymbol{y}}$. This approach allows us to control that $\tilde{\boldsymbol{y}}$ always maintains the form of a probability distribution, thus avoiding the invalid labels that would result from directly computing the gradient w.r.t. $\tilde{\boldsymbol{y}}$.

In meta-disambiguation, our starting point is to improve the model's generalizability on the support set by adjusting pseudo-labels, being outcome-oriented. From the perspective of the influence function, instead, we return to the essence of label disambiguation, observing the different impacts caused by perturbations in various directions on a sample, which allows us to determine whether there are any substitutions that are more likely to be the ground-truth. The influence function perspective provides better interpretability for such an iterative algorithm, allowing us to more intuitively understand how the labels of unlabeled samples are progressively distilled by constantly finding a substitution that prompts the model to make better predictions. Besides, by establishing such a connection between meta-disambiguation and influence function, some relevant studies can also provide inspiration for accelerating the calculation of δ, such as [6, 10], which may further improve the efficiency of our algorithm.

From the perspective of influence function, it also explains why the effectiveness of meta-disambiguation is superior to the prevalent reweighting strategy, which adopts a different loss function:

$$\hat{\theta}_{\varepsilon,z_k} = \arg\min_{\theta}[\mathcal{L}_{\text{cls}}(\mathcal{B}_{\text{train}}, \theta) + \varepsilon\ell(f_{\theta}(\boldsymbol{x}_k), \tilde{y}_k)] \tag{18}$$

which essentially investigates the influence of the existence of a particular sample z_k. Though this can also be considered as a way to identify false negatives, the existence of which is likely to have a negative influence on the evaluation loss, the adjustment on $\tilde{\boldsymbol{y}}$ is not able to take place until the training ends, since the weights of the samples must be non-negative, i.e., $\varepsilon \in [-\frac{1}{n}, +\infty]$, hence false negatives can only be filtered out during training, rather than corrected. It means that estimation bias cannot be completely avoided with the reweighting strategy, which may reduce the robustness of the end-to-end training process. In contrast, the influence function of perturbation allows us to iteratively determine and apply the optimal updates on each sample during training, which is more efficient and effective.

B.3. Additional Explanations of the Support Set

To emphasize that the introduction of the golden support set, which consists of both labeled positives and negatives, will not cause data leakage or result in unfair comparisons, we provide more detailed explanations in this section, to prove that such a setting is legitimate from various aspects:

1. First and foremost, we are not the first to apply the technique of meta-learning to PU learning, instead, we simply follow the setups of receptive work Self-PU [1], where the validation set is used as the support set for sample reweighting.
2. Technically, we do not actually update the model parameters in the **Label-Update** stage, as it is called "virtually updated" in Section 4.2. Instead, from the perspective of the influence function, we simply inspect its behavior when perturbing the pseudo-labels $\tilde{\boldsymbol{y}}$ with the support

Setup	# Train	# Test	Input Size	Class Prior	Task	Backbone
CIFAR-10	50,000	10,000	$3 \times 32 \times 32$	0.4	Vehicle Recognition	7-layer CNN
CIFAR-100-1	50,000	10,000	$3 \times 32 \times 32$	0.1	Vehicle Recognition	13-layer CNN
CIFAR-100-2	50,000	10,000	$3 \times 32 \times 32$	0.5	Animal Recognition	13-layer CNN
STL-10	105,000	8,000	$3 \times 96 \times 96$	-	Vehicle Recognition	7-layer CNN
Alzheimer	5,890	1,279	$3 \times 224 \times 224$	0.5	Alzheimer's Disease Diagnosis	ResNet-50

Table 8. Specifications of different setups.

Setup	Positive Set	Negative Set
CIFAR-10-1	0,1,8,9	2,3,4,5,6,7
CIFAR-10-2	2,3,4,5,6,7	0,1,8,9
CIFAR-100-1	18,19	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17
CIFAR-100-2	0,1,7,8,11,12,13,14,15,16	2,3,4,5,6,9,10,17,18,19
STL-10-1	0,2,3,8,9	1,4,5,6,7
STL-10-2	1,4,5,6,7	0,2,3,8,9
Alzheimer	Moderate Demented	Mild Demented, Non Demented, Very Mild Demented

Table 9. Specifications of different data partitions.

set, through which an approximation for ground-truth labels is maintained for subsequently actual training.

3. The support set for meta-disambiguation can be considered as a special usage of the validation set, which also contains both labeled positives and negatives, and is commonly used in other PU learning methods for selecting the best model that exhibits the highest generalizability as the last step of training to avoid overfitting. In LaGAM, we simply adopt a different approach to utilizing the validation set for enhancing model generalizability, specifically by leveraging the gradient information provided by the evaluation loss to guide the label disambiguation process. In return, the model trained with LaGAM exhibits high generalizability throughout such a robust training process and thus can be directly applied to the test set without any additional steps of model selection. According to the testing results, this will not cause any accuracy loss, which saves data used for validation in common methods.
4. The actual training of the classifier takes place after meta-disambiguation, where the model parameters are optimized using another objective function $\mathcal{L}_{\text{cls}} + \beta \mathcal{L}_{\text{cont}}$, which is different from the process of Meta-Train and will not be affected by the support set.
5. To support the soundness of LaGAM, we report the effects of **(i)-*model directly trained on the valid set***; and **(ii)-*FixMatch [11], leveraging the valid set for semi-supervised learning***, both of which yield poor performance. We suppose that the supervision provided by such **limited and biased** data is too weak to train a robust model, further proving that LaGAM has not gained benefits from any **unfair misuse** of the validation set and sticks to PU setting.
6. To better support our perspective, we also provide sensitivity analysis on the size of the support set in Section C.4, where we can see that minimal support samples are enough to have a significant effect, which is far beyond what this amount of data could achieve through providing supervision signals in binary classification loss. This result inversely illustrates that the superiority of LaGAM is not due to the involvement of extra labeled data in model updating, but the robustness of the whole training process, where the support set only serves for data alignment.

Method	CIFAR-10	CIFAR-100-1	CIFAR-100-2
LaGAM	96.2	92.1	86.6
Dist-PU	88.8	65.8	69.1
D.T.	84.4	Degenerated	66.7
FixMatch	91.1	52.4	57.5

Table 10. Comparisons between LaGAM and unfair misuses of the validation set.

Ablation	CIFAR-10	CIFAR-100
LaGAM	**96.6**	**89.8**
Only CL	60.9	54.2
P^{3}MIX-C+CL	83.9	65.7
Dist-PU+CL	91.2	73.1
Only P^{3}MIX-C	76.1	51.4
Only Dist-PU	88.8	69.1

Table 11. Classification accuracy of different methods combined with group-aware contrastive learning (using ResNet-18 as the backbone).

B.4. Synergy between Group-Aware Contrastive Learning and Meta-Disambiguation

While seemingly separated from each other, the two key components of LaGAM (i.e., meta-disambiguation and group-aware contrastive learning) work in a collaborative fashion. As detailed procedure shown in Algorithm 1, within such a framework, on one hand, the discriminative representations obtained from contrastive learning provide more comprehensive semantics for label disambiguation, optimizing the interpretability of gradient backpropagation to improve the accuracy of label updates. On the other hand, the results of label disambiguation further enhance the accuracy of representation learning in the manner of dichotomized cut-off, effectively aligning the latent groups learned by unsupervised clustering with the actual sub-categories that contribute to binary classification.

Experimental results displayed in Table 3 also demonstrate that the two components are complementary to each other and achieve an effect of "$1 + 1 > 2$", where the performance of LaGAM is significantly reduced when either of them is removed. To demonstrate the excellent compatibility of the two components, we also experiment with the performance of employing group-aware contrastive learning individually in other state-of-the-art methods. As shown in Table 11, apparently, the representations learned by the contrastive learning module alone in the absence of any form of label disambiguation lack effectiveness, leading to poor performance on both CIFAR-10 and CIFAR-100. While being jointly used with P^{3}MIX-C [7] and Dist-PU [14], group-aware contrastive learning significantly improves their learning capacity and the overall performance, when using ResNet-18 as the backbone. Despite the improvements, LaGAM still holds the best performance, due to the significant data alignment provided by robust meta-disambiguation, for which the effect of dichotomized cut-off can be better promised.

C. Additional Experiments and Analyses

C.1. Experimental Settings

Detailed statistics are shown in Table 8, 9.

Backbone	Method	CIFAR-100-1	CIFAR-100-2
7-Layer	VPU	79.1±4.3	68.1±1.6
	P^{3}MIX-E	73.7±2.7	66.4±1.8
	P^{3}MIX-C	73.6±2.4	67.8±2.0
13-Layer	VPU	90.1±0.1	50.0±0.1
	P^{3}MIX-E	87.3±1.3	54.9±1.6
	P^{3}MIX-C	88.1±0.9	52.9±1.2

Table 12. Classification accuracy of the same methods with different CNN backbones.

C.2. Additional Analyses on the Comparison Results

Learning capacity makes difference. From Table 1, we can notice that most of the baselines suffer serious degradation on CIFAR-100, where methods like VPU [4] and P^{3}MIX [7] even degenerate into a trivial solution, assigning every input to the same class, while LaGAM still maintains high performance, even approaching the supervised counterpart. We suspect that this is due to a mismatch between the complexity of the network architecture, the learning capacity of the model, and the quality of the training data. To verify our hypothesis, we test the performance of VPU and P^{3}MIX on CIFAR-100 with a shallower 7-layer CNN, which is the same as the architecture used for CIFAR-10 and STL-10. As the results show in Table 12, despite the overall poor performance caused by the inherent difficulty of the CIFAR-100 tasks, at least all three baselines have escaped the dilemma of the trivial solution, exhibiting certain signs of learning, especially on CIFAR-100-2, which is consistent with our hypothesis. This also indicates that common baselines do indeed lack sufficient learning capacity to adapt to deeper neural networks, thereby failing to extract more complex connections between features and labels from the training samples when facing more challenging tasks, while LaGAM is able to alleviate this problem by a well-designed group-aware representation learning module.

Neccessity of group-awareness. To show that awareness of latent groups can indeed improve the model's understanding of specific binary classification tasks under PU scenarios, we also wonder: will the classification accuracy increase if we actively inject more knowledge about the latent categories into the model? Therefore, we conduct a validation experiment on the supervised baseline, firstly pretraining a ResNet-18 on each dataset with labels of original

categories for 200 epochs, followed by another 200 epochs of fine-tuning using the binary labels. As results show in Table 13, the classification accuracies get significantly improved under all setups, indicating that the classifier trained only with binary labels does not yet have a sufficiently comprehensive understanding of the data's semantics, while the intention of our well-designed representation learning module for mining the grouping patterns underlying PU data can effectively compensate for this weakness.

Setup	w/o Pretraining	Pretrained	LaGAM
CIFAR-10-1	98.7±0.5	98.9±0.2	96.2±0.5
CIFAR-10-2	98.7±0.5	98.9±0.2	96.1±0.3
CIFAR-100-1	92.6±0.7	97.8±0.3	92.1±0.4
CIFAR-100-2	85.9±0.1	94.4±0.2	86.6±0.3

Table 13. Classification accuracy of supervised baseline before and after pertaining, compared with our LaGAM.

Ablation	ID	UC	DC	NS	Acc.
LaGAM	✓	✓	✓	✓	**89.8**
w/o Neighor Smoothing	✓	✓	✓	✗	89.4
w/o NS + Dichotomized Cutoff	✓	✓	✗	✗	87.1
Only Instance Discrimination	✓	✗	✗	✗	81.9

Table 14. Additional ablation study on positive set constructions in $\mathcal{L}_{\text{cont}}$ on CIFAR-100-2.

C.3. Additional Ablation Study

Considering that the results shown in Table 5 may not be significant enough to reflect the effects of dichotomized cut-off and local neighbor smoothing, here we supplement it with additional ablation results on CIFAR-100-2. As shown in Table 14, it is obvious that dichotomized cut-off can significantly improve the classification accuracy, while local smoothing can also slightly boost the model's performance.

β	CIFAR-10	CIFAR-100
0.0	**96.6**	**89.8**
0.1	96.2	89.4
0.2	95.8	89.4
0.5	96.2	89.0

Table 15. Sensitivity analysis on β (default 0).

C.4. Additional Sensitivity Analyses

Balance between $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{cont}}$. By examining the effects of adopting different β as shown in Table 15, we can draw a simple conclusion that it is necessary to ensure enough intensity of contrastive representation learning to achieve the optimal performance, for which we set the default value of β to 0 in actual implementation.

$\mathcal{D}_{\text{sup}}$ **Size**	CIFAR-10	CIFAR-100	STL-10
60	**96.7**	**91.7**	**93.3**
100	94.1	86.0	91.1
200	94.5	89.5	90.5
500	96.6	89.8	89.9
800	95.4	88.7	89.1

Table 16. Sensitivity analysis on the size of the support set (default 500).

Minimal support data is enough. Intuitively, to prove that the effectiveness of meta-disambiguation does not depend on a large amount of support data, which is sometimes infeasible in real-world practice, we examine the performance of LaGAM using $\mathcal{D}_{\text{sup}}$ of different sizes. Surprisingly, as results show in Table 16, with only 60 support samples, LaGAM can achieve the optimal performance. Less doesn't mean worse, conversely, more support data is also not necessarily helpful for better label disambiguation: the classification accuracy actually does not improve with the increase in the size of $\mathcal{D}_{\text{sup}}$, exhibiting a rather unstable state instead. We suspect that the effectiveness of meta-disambiguation is related to the consistency of label updates across different iterations. Specifically, though larger $\mathcal{D}_{\text{sup}}$ can more precisely represent the real sample distribution, the inconsistency in label updates arises since the evaluation losses for different training batches $\mathcal{B}_{\text{train}}$ are calculated on different support batches $\mathcal{B}_{\text{sup}}$ (according to Algorithm 1), which are randomly sampled from $\mathcal{D}_{\text{sup}}$. This means that with larger $\mathcal{D}_{\text{sup}}$ being separated into different support batches, training samples may actually be aligned with different distributions, leading to a lack of consistency in the updates of labels. Additionally, from the perspective of the influence function, the gradient of the evaluation loss actually reflects the influence of the training samples on all instances from $\mathcal{B}_{\text{sup}}$, which can be flattened out by a large amount of evaluating samples and thus lead to weaker guidance on label updating. However, one of the reasons why we still set the default size of $\mathcal{D}_{\text{sup}}$ as 500 is to keep consistent with the experimental setups used in other baselines [7], where the size of validation set is usually 500. More importantly, the outstanding performance achieved with only 60 support samples in our experiments can partly be attributed

to our prior knowledge about the latent groups when uniformly selecting support data from each sub-category. In other words, in real-world scenarios, when the quality of $\mathcal{D}_{\text{sup}}$ cannot be well promised, we may need a larger $\mathcal{D}_{\text{sup}}$ to ensure the model's generalizability. In future work, we may further explore how to more reasonably allocate validation data for model selection and meta-disambiguation.

References

[1] Xuxi Chen, Wuyang Chen, Tianlong Chen, Ye Yuan, Chen Gong, Kewei Chen, and Zhangyang Wang. Self-pu: Self boosted and calibrated positive-unlabeled training. In *ICML*, pages 1510–1519, 2020. 1, 2, 3

[2] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *ICML*, pages 1386–1394, 2015.

[3] Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *NeurIPS*, pages 703–711, 2014. 2

[4] Wenpeng Hu, Ran Le, Bing Liu, Feng Ji, Jinwen Ma, Dongyan Zhao, and Rui Yan. Predictive adversarial learning from positive and unlabeled data. In *AAAI*, pages 7806–7814, 2021. 5

[5] Ryuichi Kiryo, Gang Niu, Marthinus Christoffel du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *NeurIPS*, pages 1675–1685, 2017. 2

[6] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, pages 1885–1894. PMLR, 2017. 2, 3

[7] Changchun Li, Ximing Li, Lei Feng, and Jihong Ouyang. Who is your right mixup partner in positive and unlabeled learning. In *ICLR*. OpenReview.net, 2022. 1, 5, 6

[8] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, pages 587–594. Morgan Kaufmann, 2003. 2

[9] Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. Partially supervised classification of text documents. In *ICML*, pages 387–394. Morgan Kaufmann, 2002. 2

[10] Zhongzheng Ren, Raymond A. Yeh, and Alexander G. Schwing. Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. In *NeurIPS*, 2020. 3

[11] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020. 4

[12] Daiki Tanaka, Daiki Ikami, and Kiyoharu Aizawa. A novel perspective for positive-unlabeled learning via noisy labels. *CoRR*, abs/2103.04685, 2021. 2

[13] Xinrui Wang, Wenhai Wan, Chuanxin Geng, Shaoyuan LI, and Songcan Chen. Beyond myopia: Learning from positive and unlabeled data through holistic predictive trends. *arXiv preprint arXiv:2310.04078*, 2023. 1

[14] Yunrui Zhao, Qianqian Xu, Yangbangyan Jiang, Peisong Wen, and Qingming Huang. Dist-pu: Positive-unlabeled learning from a label distribution perspective. In *CVPR*, pages 14461–14470, 2022. 5