# What Makes Partial-Label Learning Algorithms Effective?

**Jiaqi Lv**[1,5], **Yangfan Liu**[1], **Shiyu Xia**[1], **Ning Xu**[1,5], **Miao Xu**[2], **Gang Niu**[3,1], **Min-Ling Zhang**[1,5], **Masashi Sugiyama**[3,4], **Xin Geng**[1,5]*

[1]Southeast University [2]The University of Queensland

[3]RIKEN Center for Advanced Intelligence Project [4]The University of Tokyo

[5]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China

`{is.jiaqi.lv, gang.niu.ml}@gmail.com`,
`miao.xu@uq.edu.au`, `sugi@k.u-tokyo.ac.jp`,
`{liuyangfan, shiyu_xia, xning, zhangml, xgeng}@seu.edu.cn`

## Abstract

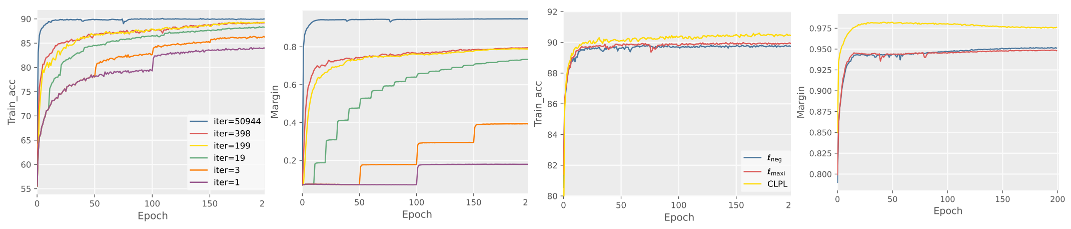
A *partial label* (PL) specifies a set of candidate labels for an instance and *partial-label learning* (PLL) trains multi-class classifiers with PLs. Recently, many methods that incorporate techniques from other domains have shown strong potential. The expectation that stronger techniques would enhance performance has resulted in prominent PLL methods becoming not only highly complicated but also quite different from one another, making it challenging to choose the best direction for future algorithm design. While it is exciting to see higher performance, this leaves open a fundamental question: *what makes a PLL method effective?* We present a comprehensive empirical analysis of this question and summarize the success of PLL so far into some *minimal algorithm design principles*. Our findings reveal that high accuracy on benchmark-simulated datasets with PLs can misleadingly amplify the perceived effectiveness of some general techniques, which may improve representation learning but have limited impact on addressing the inherent challenges of PLs. We further identify the common behavior among successful PLL methods as a progressive transition from uniform to one-hot pseudo-labels, highlighting the critical role of *mini-batch PL purification* in achieving top performance. Based on our findings, we introduce a *minimal working algorithm* that is surprisingly simple yet effective, and propose an improved strategy to implement the design principles, suggesting a promising direction for improvements in PLL.

## 1 Introduction

*Partial-label learning* (PLL) [14, 12, 42] has been an established discipline in the weakly supervised learning field [50, 51] for decades. It aims to train multi-class classifiers from instances with *partial-labels* (PLs)—a PL for an instance is a set of candidate labels, where a *fixed but unknown* candidate is the true label.

As benchmarking on simulated PLL versions of vision datasets becomes standard practice for evaluating PLL methods, new PLL approaches are emerging that integrate diverse advanced techniques to enhance the performance. While many methods show great promise and some headway has been made in understanding the methodology against ambiguous label assignments [24], we find that state-of-the-art (SOTA) approaches *look quite complicated and differ significantly from each other* (Table 1), making it challenging to choose the best direction for better algorithm design.

*Corresponding author.



(a) Acc of IBS (b) Margin of IBS (c) Acc of ABS (d) Margin of ABS

Figure 1: Training accuracy and confidence margin of predicted pseudo-labels for traditional IBS and ABS methods on FMNIST with PLs: (a-b) show the granularity of EM execution in classical IBS methods is refined from a single step to an entire epoch, facilitating a smoother transition from uniform to one-hot pseudo-labels; (c-d) demonstrate that when ABS methods are optimized using SGD, the optimization targets for candidate labels can gradually become distinct.

In this paper, we aim to summarize the success of PLL so far, and understand the indispensable elements in a well-performing PLL method, which can be condensed into *minimal algorithm design principles*. For this purpose, we think of the taxonomy of PLL approaches first, comb the evolution and trends in this field, thereby uncovering key factors that drive their effectiveness and motivating future research.

The widely accepted PLL taxonomy [47, 24] divides methods based on the treatment of PLs into *identification-based strategy* (IBS) and *average-based strategy* (ABS):

- IBS disambiguates each PL to select the most likely true label for training;
- ABS treats all candidate labels equally for training.

However, the boundary between these two categories is vague, resulting in a lack of consensus on the category of many recent approaches which *purifies each PL on the fly during model training* [25, 37, 39]. They initially look like ABS, since uniform targets are always used to prepare for true-label selection, and as training progresses, the optimization targets for candidate labels become distinct gradually, such that they exhibit IBS-like characteristics. The tricky fact is that such approaches are classified as ABS or IBS hinging on which definition of category researchers are willing to relax (e.g., [45] versus [35])! In light of the evolving approaches within PLL, is there a need to establish a third category within the taxonomy to capture the "hybrid strategy", as attempted by recent work like [43, 5]?

We suggest that the answer is in the negative, since our analysis confirms the fluidity of method categorization within PLL, emphasizing that IBS, ABS, and so-called hybrid strategy often overlap due to the dynamic nature of their implementation:

- Training manner of IBS. Typical IBS approaches [48, 13] perform one-step EM to identify the true label in each PL (E step) and then train a supervised classifier (M step). To mitigate overfitting to wrongly identify labels, multi-step EM IBS [11, 10] are proposed. As iterations are executed more frequently, the identifocation of true labels becomes increasingly smooth (Figure 1).
- Implicit differentiation in ABS. Typical ABS approaches [19, 6] design a loss function that does not differentiate candidate labels, e.g., enforcing the softmax outputs of them to sum to 1 [19]. However, stochastic optimization algorithms can *implicitly* lead to distinct outputs for each candidate label and show a progressive purification characteristic (Figure 1 and Lemma 3.1).
- Execution matters. Whether a method truly exhibits progressive characteristics and the extent of progression also depend on the optimizers used and hyperparameters like the learning rate and batch size (Figure 4).

Therefore, predefining a method's category and then asserting its utility can limit our understanding of its true nature and effectiveness. Further, our empirical incestigations reveal a key insight: all successful PLL algorithms exhibit a common behavior characterized by a *progressive transition from uniform to one-hot pseudo-labels*, facilitated by the combination of PL purification and model updates in a *mini-batch-wise manner*. Building on this core strategy, modern methods also integrate various techniques from other domains, exemplified by Match from semi-supervised learning [3, 29], aiming to further enhance model performance. While intuitively, better performance could be achieved if stronger techniques are employed, our findings indicate that these enhancements often yield marginal gains when compared to the primary benefits derived from mini-batch PL purification. Notably, even

Table 1: Comparison of techniques used in eight prominent PLL methods. ✓/× indicates whether a technique is used, and an underline denotes the key components of the respective methods.

| PLL methods | Mini-b. purif. | Mixup | Data augment. | Exp.Mov Average | Match | | Main assumption |
|---|---|---|---|---|---|---|---|
| | | | | | DA | DM | |
| PRODEN [25] | ✓ | × | × | × | × | ✓ | DNNs learn pattern first |
| CC [12] | ✓ | × | × | × | × | × | PLs are generated uniformly |
| LWC [37] | ✓ | × | × | × | × | ✓ | PLs are class-dependent |
| PiCO [34] | ✓ | × | ✓ | ✓ | ✓ | ✓ | Same class representations cluster |
| DPLL [38] | ✓ | × | ✓ | × | ✓ | × | Input is invarient to the translations |
| SoLar [33] | ✓ | ✓ | ✓ | ✓ | ✓ | × | High-confid. sample is likely correct |
| PaPi [40] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | Same class representations cluster |
| CroSel [30] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | Stable high-confid. sample is correct |

when these additional techniques result in significant improvements, they tend to boost the model's ability to learn representations rather than resolving the inherent ambiguities of PLs.

The main contributions of our paper can be summarized as follows: (i) We advance the understanding of PLL taxonomy and establish minimal algorithm design principles. At the core of these principles is mini-batch PL purification, a fundamental aspect that goes beyond using supervised information. These principles not only enhance the efficiency of algorithm development but also act as a conclusive work to prevent redundant efforts in future research. (ii) We analyze the design philosophies and component frameworks of SOTA PLL methods, conducting comprehensive studies on benchmark-simulated datasets with PLs. Building on this, we highlight a *minimal working algorithm* that adheres to our design principles, and propose an enhancement strategy to mini-batch PL purification that have the potential to elevate performance across all existing SOTA methods.

## 2 Preliminaries

### 2.1 Notation

Consider a $k$-class classification problem. Let $\boldsymbol{x} \in \mathcal{X}$ be features and $y \in \mathcal{Y} \doteq \{1, 2, \ldots, k\}$ be labels. Then one has $(\boldsymbol{x}, y)$ sampled from the ground-truth joint density $p(\boldsymbol{x}, y)$ over $\mathcal{X} \times \mathcal{Y}$ in supervised learning. PLL deals with PLL data $(\boldsymbol{x}, S)$, which is independently drawn from a corrupted distribution $p(\boldsymbol{x}, S)$ of $p(\boldsymbol{x}, y)$ with $p(\boldsymbol{x})$ unchanged. $S \in \{2^{[k]} \backslash \emptyset \backslash [k]\}$ denotes a PL, and $\mathcal{D} = (\boldsymbol{x}_i, S_i)_{i=1}^n$ is a PLL dataset. The key assumption of PLL is that the latent true label of an instance is always included in its PL, i.e., $p(y \in S | \boldsymbol{x}, S) = 1$. Let $\Delta^{k-1} \subset [0, 1]^k$ denote the $k$-dimensional simplex. Let $f : \mathcal{X} \to \Delta^{k-1}$ be a multi-label classifier to be trained, specifically, a composite of a backbone (e.g., ResNet [18]) and an inverse link function $\psi^{-1}$ [27] (e.g., `softmax`), so that $f(\boldsymbol{x})$ can be interpreted as probabilities. Let $\ell : \Delta^{k-1} \times \mathcal{Y} \to \mathbb{R}_+$ be a surrogate loss function, e.g., cross-entropy loss. The classification risk of $f$ is defined as $\mathcal{R}(f) = \mathbb{E}_{p(\boldsymbol{x},y)}[\ell(f(\boldsymbol{x}), y)]$, which is the performance measure we would like to optimize.

### 2.2 Backgrounds

In this section, we review recent prominent PLL works. In PLL, a common practice is to adopt a weighted objective of the form

$$\mathcal{R}(f) = \mathbb{E}_{p(\boldsymbol{x},S)}\big[\textstyle\sum_{z \in S} w(\boldsymbol{x}, z)\ell(f(\boldsymbol{x}), z)\big], \tag{1}$$

with the optimal weights $w(\boldsymbol{x}, z) = p(z|\boldsymbol{x})$. If $\ell$ is the cross-entropy loss, the weight can be integrated directly into the loss function, acting as optimization target (pseudo-label) for $\boldsymbol{x}$ directly. In classical taxonomy for PLL [47], IBS trains predictive models based on fixed weights assigned to the training instances, i.e., $\hat{w}(\boldsymbol{x}, z) = \{0, 1\}^k$, while ABS sets uniform weights during training, i.e., $\hat{w}(\boldsymbol{x}, z) = 1/|S|$.

**Definition 2.1** (Mini-batch PL purification)**.** Mini-batch PL purification is a process where for each mini-batch $\mathcal{B} \subset \mathcal{D}$ selected at iteration $t$, the weights are updated such that the distinction among candidate labels' contributions increases over iterations:

$$w_{t+1}(\boldsymbol{x}; f, S) = g(\text{model's confidence for } \boldsymbol{x} \text{ based on current and previous iterations}), \quad (2)$$

with $g$ being a strictly increasing function that increases the weight for more likely candidate labels according to the model's confidence. The model's parameters $\theta_t$ are updated by optimizing a weighted loss over $\mathcal{B}$:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \sum\nolimits_{(\boldsymbol{x},S)\in\mathcal{B}} \ell(f(\boldsymbol{x};\theta_t), S; w_{t+1}(\boldsymbol{x})). \quad (3)$$

The standard practice is initializing the weights uniformly $w_i^0 = 1/|S|$ if $i \in S$ and $w_i^0 = 0$ otherwise, and let $f^0$ be initialized randomly. The model $f^0$ is then updated for at least one epoch to perform a preliminary training phase. Then in each mini-batch of $t$-epoch, $w(\boldsymbol{x})$ is computed where $f$ is fixed, and then $f$ is updated by the weighted objective where $w$ keeps fixed in backpropagation. An instantiation of mini-batch PL purification was first introduced by [25]. They use a delayed mechanism, i.e., $w(\boldsymbol{x})$ is computed by the output of historical model $f^{t-1}$ on $\boldsymbol{x}$, implicitly assuming that DNNs learn pattern first [2], and the delayed mechanism mitigates the accumulation of errors. Then, some methods replace the delayed mechanism with Match techniques to estimate $w$, which may rely on either Siamese networks [4] applied to two or more inputs (dual-augmentation match, i.e., DA), or co-teaching networks [16] where one network's outputs serve as targets for the other (dual-model match, i.e., DM). Furthermore, some methods advocate DA+DM framework; for example, PiCO [34] involves mutual guidance of two heterogeneous classifiers, with one built on top of a supervised contrastive learning architecture [20]. In addition techniques borrowed from various communities have propelled PLL methods to top performance, such as mixup [46] and data augmentation like cropping and flipping [28], which have become mainstream.

As shown in Table 1, prominent methods in recent years rely on the mini-batch PL purification strategy and specific enhanced tools. However, our research reveals a significant disparity in the impact of these components, at least on current benchmarks. Mini-batch PL purification is sufficient to provide a reliable guarantee of performance, while the additional tools contribute relatively little. Although techniques like data augmentation can enhance the performance by improving the model's robustness to input variations, they primarily boost representation learning, but not benefits the disambiguition for PLL.

## 3 Understanding Minimal Algorithm Design Principles

In this section, we inverstigate four SOTA PLL methods that have consistently demonstrated top accuracy across various benchmark tasks. By methodically dissecting these methods and analyzing the components credited for their robustness against PLs, we distill the essential elements contributing to their success. We defer the experiments details to Appendix.

### 3.1 PLL with DA Match

**Algorithm details.** The key contribution of DPLL [38] lies in incorporating neighborhood consistency, a technique adapted from semi-supervised learning, into PLL. This technique maximizes the similarity among several perturbed views of the same instance, thereby inducing smoothness in the structure of learned representations, referred to as dual-augmentation match (DA Match).

Specifically, DA Match instantiates Eq. 1 by specifying the loss function as the KL divergence and estimating the weights by a weighted sum of outputs of all augmentations in $\mathcal{A}(\boldsymbol{x})$:

$$\mathcal{R}(\boldsymbol{x}, f) = -\sum\nolimits_{\mathcal{A}(\boldsymbol{x})} \hat{w}(\boldsymbol{x})^\top \log f(\mathcal{A}(\boldsymbol{x})), \quad (4)$$

$$\hat{w}(\boldsymbol{x}, z) = \begin{cases} \dfrac{\left(\prod_{\alpha\in\mathcal{A}(x)} f_z(\alpha)\right)^{1/|\mathcal{A}(x)|}}{\sum_{j\in S}\left(\prod_{\alpha\in\mathcal{A}(x)} f_j(\alpha)\right)^{1/|\mathcal{A}(x)|}} & z \in S, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

$\hat{w}$ is updated in a mini-batch-wise manner along with the model parameters and is initialized uniformly (mini-batch PL purification). In addition, the learning objective of DPLL includes another loss function as

$$\ell(\boldsymbol{x}, f) = -\sum\nolimits_{i\notin S} \log(1 - f_i(\mathcal{A}(\boldsymbol{x}))), \quad (6)$$

where the vector subscript indicates the element is that position. It encourages the output of each non-candidate label to be zero.

**Evaluation 1. Specific implementations do not matter.** First, we replace the two terms in learning objective with alternative approaches. Eq. 6 is conceptually equivalent to encouraging the sum of the outputs for the candidate labels to be close to one, i.e,

$$\ell(\boldsymbol{x}, f) = -\log \sum\nolimits_{i \in S} f_i(\mathcal{A}(\boldsymbol{x})). \tag{7}$$

Instead of using a shared target for all views, we modifies the optimization target for each view separately based on the output of the other view:

$$\mathcal{R}(\boldsymbol{x}, f) = -\big(\hat{w}_2(\boldsymbol{x})^\top \log f(\boldsymbol{x}_1) + \hat{w}_1(\boldsymbol{x})^\top \log f(\boldsymbol{x}_2)\big), \tag{8}$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{A}(\boldsymbol{x})$, and

$$\hat{w}_{2(1)}(\boldsymbol{x}, z) = \begin{cases} f_z(\boldsymbol{x}_{1(2)}) / \sum_{j \in S} f_j(\boldsymbol{x}_{1(2)}) & z \in S, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

We combine two DA Match terms and two loss functions in pairs, respectively. As can be seen from Table 2, there was no significant difference between the results of these four combinations. This indicates that the specific implementation methods of DA Match, including the exact form of the loss function, is not critical.

**Evaluation 2. Additional losses do not matter.** We then split the combined learning objectives into two separate objectives to examine the difference in the contribution of these two components to learning. We found that Eq. 6 and Eq. 8 were comparable under the relatively simple settings, but Eq. 8 outperformed Eq. 4 in challenging scenarios (CIFAR-100 and mini-ImageNet). Note that Eq.6 is a traditionally considered ABS loss, and Eq. 8 implements mini-batch PL purification. It is commonly believed that ABS does not require identifying the true labels during training, leading to over-parameterized DNNs memorizing all candidate labels [44], which results in poor performance. However, as discussed in Section 1, ABS can still achieve acceptable performance because the optimization process may induce differentiated outputs. We will elaborate on this in more detail in Section 3.4. Another noteworthy observation is that Eq. 8 not only did not lead to a decrease in accuracy compared to the original DPLL, but even showed some improvement, which is likely because Eq. 8 can model neighborhood consistency better than Eq. 4.

So far, we have extracted a core unit from DPLL, which takes the form of Siamese networks [4]: a weight-sharing network applied on two (or more) inputs for comparing, and PLs prevent the model from collapsing, i.e., outputting a constant for all inputs. We call Eq. 8 *dual augmentations single model* (DASM), as depicted in Figure 5. However, it remains to be seen whether the effectiveness of DASM is due to mini-batch PL purification or neighborhood consistency.

**Evaluation 3. Mini-batch PL purification does matter.** We modify the learning strategy of DASM by either altering the mini-batch PL purification strategy or removing the consistency component. We compare them in Table 2, where the "-H", "-S" or "-E" suffix means using the hard pseudo-labels, one-step iteration or epoch-wise iteration. Specifically, DASM-H uses hard labels as optimization targets for Eq. 8:

$$\hat{w}_2(\boldsymbol{x}) = \boldsymbol{e}^i, \hat{w}_1(\boldsymbol{x}) = \boldsymbol{e}^j, \text{ where } i = \arg\max f(\boldsymbol{x}_1) \text{ and } j = \arg\max f(\boldsymbol{x}_2), \tag{10}$$

where $\boldsymbol{e}^i$ denotes the $i$-th standard canonical vector, i.e., $\boldsymbol{e}^i \in \{0,1\}^k, \mathbf{1}^\top \boldsymbol{e}^i = 1$. DASM-S simulates one-step EM methods by training the model with uniform targets for the first 50 epochs, and then transforming PLL into supervised learning by using one-hot pseudo-labels (i.e., the `argmax` of the model's output for each instance at the 50th epoch) for the next 450 epochs. The results indicated that the performance of DASM-H and DASM-S was inferior to DASM. This can be attributed to the models' inability to adjust learning targets at the *appropriate* time based on the underlying learned patterns: Since DNNs tend to fit easy patterns first and gradually memorize harder ones, a phenomenon known as memorization effects [2], DASM-H may remember unreliable information due to random initialization, and DASM-S may lead remember too much undesired memorization [15]. DASM-E shifts from a mini-batch-wise manner to an epoch-wise manner, performing pseudo-label estimation and model updates at the epoch level, which resulted in decreased accuracy. Compared to DASM-E, mini-batch manner benefits from using the up-to-date model for generating optimization objectives

Table 2: Conceptual and empirical comparisons (%) of various simplifications of DPLL. ✓/× indicates whether a technique is used.

| Methods | Mini-b. purif. | ABS loss | Data augment. | DA | FMNIST 0.3 | FMNIST 0.7 | CIFAR-100 0.05 | CIFAR-100 0.1 | mini-I.Net ins.-dep. |
|---|---|---|---|---|---|---|---|---|---|
| Eq. 4+6 DPLL | ✓ | ✓ | ✓ | ✓ | 93.82 | **92.68** | 76.81 | 75.93 | 52.22 |
| Eq. 4+7 | ✓ | ✓ | ✓ | ✓ | 93.80 | 92.44 | 79.35 | 78.85 | 53.40 |
| Eq. 8+6 | ✓ | ✓ | ✓ | ✓ | 93.49 | 92.19 | **79.75** | 78.87 | 53.78 |
| Eq. 8+7 | ✓ | ✓ | ✓ | ✓ | 93.57 | 92.20 | 79.53 | 78.85 | 54.09 |
| Eq. 6 | × | ✓ | ✓ | × | 93.58 | 92.12 | 76.96 | 75.94 | 44.69 |
| Eq. 8 DASM | ✓ | × | ✓ | ✓ | 93.89 | 92.85 | 79.70 | **79.62** | 54.71 |
| DASM-H | × | × | ✓ | ✓ | 93.86 | 92.37 | 78.25 | 33.22 | 34.59 |
| DASM-S | × | × | ✓ | ✓ | 93.28 | 90.75 | 78.65 | 76.22 | 36.71 |
| DASM-E | × | × | ✓ | × | 93.80 | 92.35 | 79.30 | 79.11 | 53.44 |
| SASM | ✓ | × | ✓ | × | **93.83** | 92.18 | 79.38 | 78.19 | **55.45** |
| DASM w/o.aug | ✓ | × | × | ✓ | 90.86 | 89.60 | 60.18 | 56.39 | 30.85 |

and also improves computational efficiency. In addition, we change the dual feedforwarding to single. The optimization target of an input is modified in place according to its own output, which we term *single augmentations single model* (SASM). It was somewhat surprising that this method does not require additional components, performed remarkably well, implying DA may not be necessary.

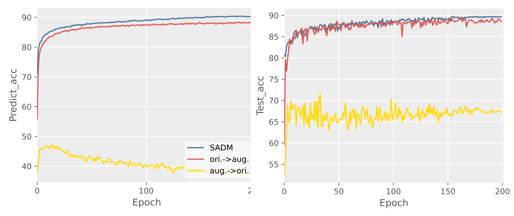
**Evaluation 4. Does data augmentation matter?** Additionally, removing data augmentation from DASM, as in DASM w/o.aug, causes a decrease in accuracy as expected. We will discuss its impact further in Section 3.3.

## 3.2 PLL with DA+DM Match

**Algorithm details.** PiCO [34] enhances representation learning by incorporating supervised contrastive learning into PLL. It utilizes two heterogeneous classifiers sharing a backbone (one linear and one contrastive-based), guiding each other to instantiate Eq. 1. PaPi [40] investigates PiCO and identifies limitations in the contrastive learning module. Thus PaPi adopts a more efficient approach inspired by the delayed mechanism in PRODEN, instantiating Eq. 1 without the need to maintain two separate models. Besides, PaPi also uses the zero-and-normalized outputs of historical models to guide a prototypical classifier that shares the same backbone with the linear classifier. Both methods feed forward different views of the same input. We refer to such franework as dual-augmentation and dual-model match (DA+DM Match). CroSel [30] is the latest PLL method that, in addition to using the DM framework to generate optimization targets for each other, also selects samples with more accurate pseudo-labels for the other model to compute supervised loss.

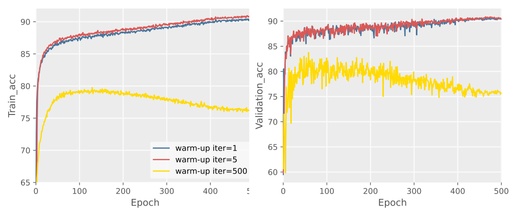
**Evaluation.** At a high level, PiCO draws inspiration from co-teaching [16]: instead of training a single classifier, it trains two classifiers simultaneously and lets them teach each other in every mini-batch. We simplify this idea by removing the contrastive-based classifier and using two networks with the same architecture but different initialization, which is also CroSel without its sample selection module. We call such method *dual augmentations dual models* (DADM). Taking it a step further, *single augmentation dual models* (SADM) removes one data augmentation, feeding both networks the same view of an instance within the same epoch. Conversely, if we cancel DADM from CroSel, the remaining implementation is co-teaching adapted for PLL (Coteaching in Tabel 4). For PaPi, we strip away the prototypical classifier, resulting in a streamlined version akin to PRODEN with added data augmentation (PRODEN+), to explore whether the specific instantiation of DM makes a difference. Figure 5 illustrates their basic workflow.

Our results are shown in Table 4. Both DADM and SADM outperformed PiCO and were comparable with CroSel, and PRODEN+ generally matches the performance of PaPi. Deleting the implementation of mini-batch PL purification, whether by altering the iteration frequency or replacing soft pseudo-



(a) Acc of pseudo-labels (b) Test Acc

Figure 2: Comparative results of different training setups of DASM on FMNIST with PLs. (a) shows the pseudo-label accuracy, while (b) presents the test accuracy.



(a) Train Acc (b) Validation Acc

Figure 3: Warm up different iterations on FMNIST with PLs. Red lines mean terminating the warm-up phase at a local maximum in validation accuracy.

labels with hard pseudo-labels, consistently leads to a decline in performance. The observations reaffirm that mini-batch PL purification is essential for achieving top performance. By comparing the performance of the DM framework with the SM framework, we discover that the DM framework's edge often comes from the diverse capabilities of the two networks, which help handle the noise introduced by PLs. On the other hand, the DA methods without any special design (DASM, DADM, PRODEN+) showed suboptimal performance than the SA methods, perhaps hinting at the importance of exploring different augmentation methods [31] to discover appropriate choices for class-invariant patterns, remaining for future research.

## 3.3 Does Data Augmentation Help Identification Better?

Our findings indicate that mini-batch PL purification and data augmentation are pivotal for PLL, achieving competitive performance when both techniques are implemented. Data augmentation is a well-established regularization tool, enhancing model robustness to input variations. In the following, we explore whether its role in PLL extends beyond this conventional purpose, specifically whether it facilitates the crucial task of identifying the true label among candidate labels. If data augmentation aids true-label identification, we would expect that augmented examples generate more accurate pseudo-labels, then enhancing the performance. Specifically, we experiment with DASM by setting one view as augmented and the other as non-augmented. We also use different training setups: (i) using zero-and-normalized pseudo-labels generated from original $\boldsymbol{x}$ to supervise augmented $\boldsymbol{x}$, and (ii) the reverse, using augmented $\boldsymbol{x}$ to supervise original $\boldsymbol{x}$. The results are presented in Figure 2.

Switching from dual paths to a single path while retaining supervised learning on augmented instances with pseudo-labels from the original instances showed little change in accuracy. However, when using pseudo-labels from augmented instance to supervise original one, performance drops significantly. This suggests that data augmentation alone is insufficient to preserve the mutual information between examples and their true labels, as augmented views may discard task-relevant information, thereby degrading performance. Following this reasoning, our results suggest that data augmentation indirectly benefits the classifiers built upon representation learning rather than aiding in label disambiguation. Therefore, it should NOT be considered a design principle for PLL.

## 3.4 PLL through Pseudo-Label Manipulation vs.Loss Minimization

Mainstream methods mentioned above conduct mini-batch PL purification primarily by directly manipulating the optimization targets of candidate labels. However, as analyzed in Section 1 and observed in Eq. 6 of Table 2, using stochastic algorithms to minimize loss functions, even ABS loss funtions that are traditionally considered to "treat all candidate labels equally", can also implicitly result in a progressive effect of pseudo-labels. To our knowledge, no existing research has explored the relationship between pseudo-label manipulation and loss minimization. Our findings raise the question: how do these two methods compare in terms of their effectiveness and mechanisms in the context of PL identification?

We investigate several representative PLL losses, which are traditionally thought as ABS:

- Modified negative log likelihood loss (Eq. 6) [38]: $\ell_{\text{neg}}(\boldsymbol{x}, f) = -\sum_{i \notin S} \log(1 - f_i(\boldsymbol{x}))$;
- APL loss [24]: $\ell_{\text{APL}}(\boldsymbol{x}, f) = \frac{1}{|S|} \sum_{i \in S} \tilde{\ell}(f(\boldsymbol{x}), i)$, with GCE loss [49] as the component $\tilde{\ell}$;

- Maximum likelihood loss [19]: $\ell_{\text{maxi}}(\boldsymbol{x}, f) = -\log \sum_{i \in S} f_i(\boldsymbol{x})$.

Denote the scores of $\boldsymbol{x}$ outputted by the last layer before `softmax` as $\boldsymbol{z}$, i.e., $\psi^{-1}(\boldsymbol{z}) = f(\boldsymbol{x})$ where $f = (\psi^{-1} \circ f^{(n)} \circ \cdots \circ f^{(1)})$. Let us look at the gradients of $\ell_{\text{neg}}$ and $\ell_{\text{APL}}$:

$$\frac{\partial \ell_{\text{neg}}}{\partial \boldsymbol{z}_k} = \begin{cases} -f_k \sum_{i \notin S} \frac{e^{\boldsymbol{z}_i}}{\sum_{j \in \mathcal{Y}} e^{\boldsymbol{z}_j} - e^{\boldsymbol{z}_i}} & k \in S, \\ f_k (1 - \sum_{i \notin S, i \neq k} \frac{e^{\boldsymbol{z}_i}}{\sum_{j \in \mathcal{Y}} e^{\boldsymbol{z}_j} - e^{\boldsymbol{z}_i}}) & \text{otherwise}, \end{cases} \tag{11}$$

$$\frac{\partial \ell_{\text{APL}}}{\partial \boldsymbol{z}_k} = \begin{cases} -\frac{1}{|S|} (f_k \sum_{i \in S, i \neq k} f_i^q - f_k^q (1 - f_k)) & k \in S, \\ \frac{1}{|S|} f_k \sum_{i \in S} f_i^q & \text{otherwise}, \end{cases} \tag{12}$$

where $q \in (0, 1]$ is a tunable parameter of GCE. From a gradient perspective, we observe that while the implicit optimization targets for all candidate labels are the same for each loss function, their optimization speeds differ. Specifically, the gradients of the candidate labels are consistently negative until one candidate label accumulates all the probabilities, which implies that both loss functions promote the output of each candidate label to converge to 1. As a result, the candidate labels compete for dominance. A label with a higher output probability experiences a larger gradient and thus becomes the winner. According to the memorization effects of DNNs, such a label is more likely to be the true label receiving a larger gradient at the beginning, which explains why the labels become distinguishable with these two losses. However, since the optimization targets for all candidate labels remain the same, minimizing these two losses does not align with our definition of mini-batch PL purification where the optimization targets are expected to diverge progressively.

Then we focus on the third loss $\ell_{\text{maxi}}$. We examine its gradients as

$$\frac{\partial \ell_{\text{maxi}}}{\partial \boldsymbol{z}_k} = \begin{cases} f_k - \frac{e^{\boldsymbol{z}_k}}{\sum_{i \in S} e^{\boldsymbol{z}_i}} & k \in S, \\ f_k & \text{otherwise}. \end{cases} \tag{13}$$

It is crucial to recognize two key points: (i) The philosophy of $\ell_{\text{maxi}}$ is to ensure that the output of all candidate labels sums to 1, which is *exactly the same* as that of $\ell_{\text{neg}}$, which requires the output of all non-candidates to sum to 0. However, their gradients are *completely different*, indicating that they lead to different optimal empirical solutions even with identical initialization and stochastic optimizer. (ii) When learning with $\ell_{\text{maxi}}$, the implicit optimization target of an instance is the zero-and-normalized outputs of current model on this instance itself, exhibiting *consistent behavior with pseudo-label manipulation* in SASM.

**Lemma 3.1.** *Suppose using the same stochastic optimizer, then performing SASM is mathematically equivalent to minimizing $\ell_{\text{maxi}}$.*

Empirical results in Table 5 also verified the theoretical findings. However, pseudo-label manipulation offers greater flexibility as it allows for more arbitrary modifications to the optimization targets, as DASM, SADM, etc. have done, and additional steps over generating soft pseudo-labels, such as sharpening [29], but the targets of loss functions are fixed.

Now, our empirical investigations have firmly established the necessity of mini-batch PL purification along with using supervision. Supervision can be used in two ways: directly within the loss function as it in supervised learning, or to manipulate pseudo-labels on-the-fly. We identify SASM is the minimal working algorithm. It achieved superior or at least comparable performance compared with SOTA PLL methods in most cases, while not requiring multiple forward propagations or additional components.

## 3.5 Probing the Implementation of Mini-Batch PL Purification

Here, we examine the implementation of mini-batch PL purification, raising at least two questions: Q1. Does different initialization methods impact the performance? Q2. Why does model confidence in PLs is effective to disambiguation?

In early training stages, our primary concern is if initialization methods might cause certain candidate labels to receive significantly higher confidence. Consider a neural network where weights are independently drawn from a standard normal distribution with zero mean and variance $\sigma^2$ (as in normal initialization, Xavier initialization [1], He initialization [17]). For an instance $\boldsymbol{x}$, the output $z$ of a neuron after a ReLU activation is given by $z = \max(0, \boldsymbol{w}\boldsymbol{x} + b)$, where $\boldsymbol{w}$ represents the weight vector and $b$ is a small constant bias. By the Central Limit Theorem, the mean of the weights $\bar{w} = 1/n \sum_{i=1}^{n} \boldsymbol{w}_i$ for a sufficiently large number of neurons $n$ approaches a normal distribution with mean zero and variance $\delta^2/n$. Applying Chebyshev's inequality, we have $P(|\bar{w}| \geq a) \leq \delta^2/(na^2)$. As $n \to \infty$, $\bar{w}$ is close to zero with high probability. Then the output for each neuron $z$ will be close to $b$. By a similar reasoning, this result generalizes to deeper layers, suggesting that the initial outputs across classes approximate a uniform distribution regardless of the number of layers.

**Lemma 3.2.** *For a neural network initialized with weights $\mathbf{w}$ drawn from a normal distribution $\mathcal{N}(0, \delta^2)$, the initial outputs across classes approximate a uniform distribution as the number of neurons $n \to \infty$.*

Then we explain why using model's output as a proxy for the probability that a candidate label is the true label works. This practice of using the high-confidence label as the true label, introduced to PLL by [25], has long been foundational in noisy-label learning (e.g., [16, 23]). Its success in PLL hinges on a key assumption: the true label has a higher probability of appearing among the candidate labels than any incorrect label. In fact, the experimental setups employed across PLL methods impose a further restriction on this assumption, that is, $p(y \in S|x, S) = 1, p(i \in S|x, S) > 1, \forall i \neq y$. This setup implies that, within any sufficiently small neighborhood in the data space, the true label will dominate. Consequently, when stochastic gradient-based optimizers are used, the true label tends to contribute more frequently to the objective function, making it more likely to be learned first.

## 4 Improving Mini-Batch PL Purification

### 4.1 Motivation

The standard practice involves initiating the process with uniform pseudo-labels for one epoch to bootstrap the classifier's basic capabilities. Our first observation is that the effect of warming up for one epoch is nearly indistinguishable from not warming up at all, prompting further investigation into the actual efficacy of the warm-up phase. Suppose a neural network is initialized with weights drawn from a Gaussian distribution with mean zero and a sufficiently small variance. If the inputs are normalized, the pre-activation values across a neural network tend to be small and centered around zero. When these values are input to a softmax function, the resulting distribution across classes tends to uniformity.

Our second observation is that a prolonged warm-up using uniform pseudo-labels often leads to the network overfitting to candidate labels, as illustrated in Figure 3, evident around 50 epochs with a decline in validation accuracy. While overfitting to the training data becomes apparent about 120 epochs, marked by a drop in training accuracy after previously reaching a peak.

We simply terminate the warm-up phase at a local maximum in validation accuracy, about 5 epoch, preventing excessive memorization while preserving more information beneficial for generalization. Then the results showed a little improvement. Several approaches [34, 24] have used multiple epochs for warm-up and treat the number of warm-up epochs as a hyperparameter. However, due to the varying difficulty levels across samples, using a uniform duration for the warm-up phase could result in performance disparities among different subgroups within the dataset. This suggests the need for an adaptive warm-up strategy.

### 4.2 StreamPurify: An Instance-Dependent Warm-Up Strategy

Building upon the analysis and our established design principles, we propose StreamPurify, a novel instance-dependent warm-up strategy, which fine-tunes the entry into the mini-batch PL purification phase based on each instance's readiness. During the initial training phase, it selectively channels instances that have higher confidence in the accuracy of their pseudo-labels into the PL purification phase, while others continue training with uniform targets until they meet the readiness criteria. This filtered progression helps prevent DNNs from harmfully memorizing incorrect pseudo-labels and mitigate the accumulation of errors from inadequately learned samples. Differing from sample selection techniques used in noisy-label learning [36] that reevaluated samples in every iteration, our filter approach is conducted without replacement. Once the training samples are transitioned to the purification phase, they do not revert to uniform targets, ensuring that the strategy remains in line with mini-batch PL purification.

Table 3: Classification accuracy (%) improvement of PLL methods with StreamPurify.

| Methods | FMNIST | | CIFAR-10 | | CIFAR-100 | | mini-I.Net |
|---|---|---|---|---|---|---|---|
| | 0.3 | 0.7 | 0.3 | 0.7 | 0.05 | 0.1 | ins.-dep. |
| SASM | 0.15 | 0.09 | 0.02 | 0.34 | 0.30 | 0.21 | 1.82 |
| SADM | 0.05 | 0.22 | 0.10 | 0.18 | 0.19 | 0.58 | 0.99 |
| DADM | 0.03 | 0.02 | 0.03 | 0.46 | 0.66 | 0.51 | 0.72 |
| DASM | 0.36 | 0.26 | 0.13 | 0.29 | 0.22 | 0.42 | 1.41 |
| DPLL | 0.11 | 0.00 | 0.06 | 0.40 | 0.85 | 0.47 | 0.95 |
| PiCO | 0.29 | 0.43 | 0.60 | 0.49 | 0.35 | 0.62 | 1.03 |
| PaPi | 0.17 | 0.36 | 0.30 | 0.20 | 0.68 | 0.47 | 0.89 |

StreamPurify is compatible to existing PLL methods and can absorb various sample selection criteria, such as small-loss trick. We adopt the sample selection method from CroSel [30] that chooses the samples with stable and high confidence as a filter criterion within StreamPurify, and combine it with the mini-batch PL purification approaches discussed in our paper. Empirical evaluations indicate that methods augmented with StreamPurify generally exceed the performance of their conventional counterparts, validating the effectiveness of StreamPurify, especially in complicated learning scenarios. This substantiates the robustness and adaptability of instance-dependent warm-up, suggesting it as a promising direction for future improvements in PLL.

## 5 Discussion and Future Work

We have systematically delineated the core components underlying successful PLL methods, centering our insights around the pivotal effect of mini-batch PL purification. We have proposed StreamPurify, an enhanced mini-batch PL purification strategy that tailors the learning path for each sample based on its state of readiness. We reaffirm that strictly categorizing PLL methods as IBS or ABS may oversimplify the dynamics of how these methods operate in practice. In the future, we wish to explore advanced PLL approaches guided by the minimal algorithm design principles. We also wish to extend the applicability and understanding of PLL methods to domains beyond vision tasks.

## References

[1] X. G. andnY. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of 13th International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, pages 249–256, 2010.

[2] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. In *Proceedings of 34th International Conference on Machine Learning (ICML'17)*, volume 70, pages 233–242, 2017.

[3] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems 32 (NeurIPS'19)*, pages 5050–5060, 2019.

[4] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature verification using a siamese time delay neural network. In *Advances in neural information processing systems 7 (NeurIPS'94)*, pages 737–744, 1994.

[5] X. Cheng, D. Wang, L. Feng, M. Zhang, and B. An. Partial-label regression. In *Proceedings of 37th AAAI Conference on Artificial Intelligence (AAAI'23)*, pages 7140–7147, 2023.

[6] T. Cour, B. Sapp, and B. Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12(5):1501–1536, 2011.

[7] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of 32nd IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, pages 113–123, 2019.

[8] T. Devries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[9] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[10] L. Feng and B. An. Partial label learning by semantic difference maximization. In *Proceedings of 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, pages 2294–2300, 2019.

[11] L. Feng and B. An. Partial label learning with self-guided retraining. In *Proceedings of 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, pages 3542–3549, 2019.

[12] L. Feng, J. Lv, B. Han, M. Xu, G. Niu, X. Geng, B. An, and M. Sugiyama. Provably consistent partial-label learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, pages 10948–10960, 2020.

[13] C. Gong, T. Liu, Y. Tang, J. Yang, J. Yang, and D. Tao. A regularization approach for instance-based superset label learning. *IEEE Transactions on Cybernetics*, 48(3):967–978, 2018.

[14] Y. Grandvalet and Y. Bengio. Learning from partial labels with minimum entropy. 2004.

[15] B. Han, G. Niu, X. Yu, Q. Yao, M. Xu, I. Tsang, and M. Sugiyama. Sigua: Forgetting may make learning with noisy labels more robust. In *Proceedings of 37th International Conference on Machine Learning (ICML'20)*, pages 4006–4016, 2020.

[16] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems 31 (NeurIPS'18)*, pages 8527–8537, 2018.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of 15th IEEE International Conference on Computer Vision (ICCV'15)*, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of 29th IEEE conference on Computer Vision and Pattern Recognition (CVPR'16)*, pages 770–778, 2016.

[19] R. Jin and Z. Ghahramani. Learning with multiple labels. In *Advances in Neural Information Processing Systems 16 (NeurIPS'03)*, pages 921–928, 2003.

[20] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, 2020.

[21] A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*. Citeseer, 2009.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] J. Li, R. Socher, and S. C. H. Hoi. Dividemix:learning with noisy labels as semi-supervised learning. In *Proceedings of 8th International Conference on Learning Representations (ICLR'20)*, 2020.

[24] J. Lv, B. Liu, L. Feng, N. Xu, M. Xu, B. An, G. Niu, X. Geng, and M. Sugiyama. On the robustness of average losses for partial-label learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):2569–2583, 2024.

[25] J. Lv, M. Xu, L. Feng, G. Niu, X. Geng, and M. Sugiyama. Progressive identification of true labels for partial-label learning. In *Proceedings of 37th International Conference on Machine Learning (ICML'20)*, pages 6500–6510, 2020.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS'19)*, pages 8024–8035, 2019.

[27] M. D. Reidand and R. C. Williamson. Composite binary losses. *The Journal of Machine Learning Research*, 11:2387–2422, 2010.

[28] C. Shorten and T. M. Khoshgoftaar. A survey on imagedata augmentation for deep learning. *Journal of Big Data*, 6:1146–1151, 2019.

[29] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. Raffel, E. D. Cubuk, A. Kurakin, and C. Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, 2020.

[30] S. Tian, H. Wei, Y. Wang, and L. Feng. Crosel: Cross selection of confident pseudo labels for partial-label learning. In *Proceedings of 37th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'24)*, 2024.

[31] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems 33 (NeurIPS'20)*, 2020.

[32] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *Advances in neural information processing systems 29 (NeurIPS'16)*, pages 3630–3638, 2016.

[33] H. Wang, M. Xia, Y. Li, Y. Mao, L. Feng, G. Chen, and J. Zhao. Solar: Sinkhorn label refinery for imbalanced partial-label learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS'22)*, 2022.

[34] H. Wang, R. Xiao, Y. Li, L. Feng, G. Niu, G. Chen, and J. Zhao. Pico: Contrastive label disambiguation for partial label learning. In *Proceedings of 10th International Conference on Learning Representations (ICLR'22)*, 2022.

[35] W. Wang and M. Zhang. Partial label learning with discrimination augmentation. In *Proceedings of 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'08)*, pages 1920–1928, 2022.

[36] Z. Wang, J. Jiang, B. Han, L. Feng, B. An, G. Niu, and G. Long. Seminll: A framework of noisy-label learning by semi-supervised learning. *Transactions on Machine Learning Research*, 2022, 2022.

[37] H. Wen, J. Cui, H. Hang, J. Liu, Y. Wang, and Z. Lin. Leveraged weighted loss for partial label learning. In *Proceedings of 36th International Conference on Machine Learning (ICML'21)*, pages 11091–11100, 2021.

[38] D. Wu, D. Wang, and M. Zhang. Revisiting consistency regularization for deep partial label learning. In *Proceedings of 37th International Conference on Machine Learning (ICML'22)*, volume 162, pages 24212–24225, 2022.

[39] Z. Wu, J. Lv, and M. Sugiyama. Learning with proper partial labels. *Neural Computation*, 35(1):58–81, 2022.

[40] S. Xia, J. Lv, N. Xu, G. Niu, and X. Geng. Towards effective visual representations for partial-label learning. In *Proceedings of 36th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'23)*, pages 15589–15598, 2023.

[41] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[42] N. Xu, C. Qiao, X. Geng, and M. Zhang. Instance-dependent partial label learning. In *Advances in Neural Information Processing Systems 34 (NeurIPS'21)*, pages 3615–3621, 2021.

[43] Y. Yao, C. Gong, J. Deng, and J. Yang. Network cooperation with progressive disambiguation for partial label learning. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 471–488, 2020.

[44] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of 5th International Conference on Learning Representations (ICLR'17)*, 2017.

[45] F. Zhang, L. Feng, B. Han, T. Liu, G. Niu, T. Qin, and M. Sugiyama. Exploiting class activation value for partial-label learning. In *Proceedings of 10th International Conference on Learning Representations (ICLR'22)*, 2022.

[46] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of 6th International Conference on Learning Representations (ICLR'18)*, 2018.

[47] M. Zhang. Disambiguation-free partial label learning. In *Proceedings of the 14th SIAM International Conference on Data Mining*, pages 37–45, 2014.

[48] M. Zhang, B. Zhou, and X. Liu. Partial label learning via feature-aware disambiguation. In *Proceedings of 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*, pages 1335–1344, 2016.

[49] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems 31 (NeurIPS'18)*, pages 8778–8788, 2018.

[50] Z. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2017.

[51] Z. Zhou. Open-environment machine learning. *National Science Review*, 2022.

## Appendix

## A Experiments Details

As benchmarking on partially labeled vision datasets has become standard practice in evaluating deep PLL methods, we conduct experiments on FMNIST [41], CIFAR-100 [21] and mini-ImageNet [32]. We generated PLs by the Flipping strategy [24] for FMNIST, CIFAR-10 and CIFAR-100. Each label $i$ is added into PL with a flipping probability $\eta_i^y = p(i \in S|y)$ independently and features are untouched: $p(s|x,y) = M \prod_{i\in S} \eta_i^y \prod_{i\notin S}(1-\eta_i^y)$ where $M = 1/\big(1-\prod_{i\neq y}\eta_i^y\big)$. We assumed $\eta_i^y = \eta, \forall i \neq y$ and $\eta_y^y = 1$, and set $\eta = \{0.3, 0.7\}$ on FMNIST and CIFAR-10, $\eta = \{0.05, 0.1\}$ on CIFAR-100. For mini-ImageNet and ablation experiments (Figure 1, 2, 3, 4), we simulated the real scenario by setting the flipping probability for each incorrect label individually for each instance. We first trained a classifier with clean labels, and then for each instance, set the confidence prediction of the classifier as the flipping probability [42].

Our explorations used three backbones: 5-layer LeNet [22] on FMNIST, 18-layer ResNet [18] on CIFAR-10 and CIFAR-100, and 34-layer ResNet [18] on mini-ImageNet. All the methods were trained for 500 epochs with a standard SGD optimizer [9] with a momentum of 0.9 and the batch size was 256 (128 for mini-ImageNet). We left out 10% of the corrupted training samples as a validation set, and searched the initial learning rate from $\{0.1, 0.07, 0.05, 0.03\}$ with cosine learning rate scheduling. We conducted 3 trials for each experiment, and recorded the mean test accuracy in percentage. There were two kinds of random augmentations involved. "Weak" augmentation was a random horizontal flips and crops [3]. For "strong" augmentation on FMNIST and CIFAR-10, we added Cutout [8] to the weak augmentation, and on CIFAR-100 and mini-ImageNet, we additional leveraged AutoAugment [7]. We denote the augmentation by $\mathcal{A}(\cdot)$, with method clear from context. The implementation was based on PyTorch [26] and experiments were carried out with GeForce RTX 4090 D.

Notably, we focused on the core components of the SOTA PLL methods mentioned in this paper, rather than strictly adhering to the settings detailed in their original implementations. This approach was taken to ensure fair and meaningful comparisons. For instance, we did not include techniques such as mixup in PaPi or the triple augmentation used in DPLL. As a result, the reported performance metrics in our paper might be slightly lower than those presented in the original publications.

## B Experimental Results

In Figure 1 (c-d), CLPL is a traditionally considered ABS loss from [6]. Figure 4 illustrates the dynamic changes in the same method under different hyperparameters and optimization methods. One can expect that with more extreme hyperparameters or optimization methods, the approach may degrade to one-step EM or persist with nearly uniform optimization targets. This indicates that method categorization must be assessed on a case-by-case basis post hoc. Figure 5 is the workflow of several key methods proposed in the paper. Table 4 provides conceptual and empirical comparisons of various simplifications of PiCO, PaPi and CroSel. Consistent with Table 2, the results reiterate that mini-batch PL purification is pivotal to achieve top performance. Table 5 presents the results of minimizing three loss functions. It is evident that $\ell_{\text{maxi}}$ yield similar results with SASM, and owing to the mini-batch PL purification, and due to the implementation of mini-batch PL purification, the accuracy is higher than the other two losses.

Table 4: Conceptual and empirical comparisons (%) of various simplifications of PiCO, PaPi and CroSel.

| Methods | Mini-b. purif. | Data augment. | DA | DM | FMNIST 0.3 | FMNIST 0.7 | CIFAR-100 0.05 | CIFAR-100 0.1 | mini-I.Net ins.-dep. |
|---|---|---|---|---|---|---|---|---|---|
| PiCO | ✓ | ✓ | ✓ | ✓ | 93.40 | 91.64 | 76.11 | 75.65 | 48.36 |
| DADM | ✓ | ✓ | ✓ | ✓ | 93.59 | 92.40 | **80.28** | 79.13 | 53.69 |
| DADM-H | × | ✓ | ✓ | ✓ | 92.60 | 85.13 | 79.99 | 35.72 | 36.30 |
| DADM-S | × | ✓ | ✓ | ✓ | 92.12 | 87.81 | 78.16 | 74.66 | 36.42 |
| DADM-E | × | ✓ | ✓ | ✓ | 93.66 | 92.22 | 80.08 | 78.49 | 52.18 |
| SADM | ✓ | ✓ | × | ✓ | **94.02** | 92.36 | 80.11 | **79.65** | 54.64 |
| SADM-E | × | ✓ | × | ✓ | 93.72 | 92.51 | 89.85 | 78.66 | 53.08 |
| PaPi | ✓ | ✓ | ✓ | ✓ | 93.54 | 91.54 | 80.10 | 79.62 | **57.10** |
| PRODEN+ | ✓ | ✓ | ✓ | ✓ | 93.70 | **92.52** | 79.85 | 79.51 | 52.39 |
| PRODEN | ✓ | × | × | ✓ | 91.61 | 90.45 | 62.47 | 59.10 | 29.21 |
| CroSel | ✓ | ✓ | ✓ | ✓ | 93.84 | 92.31 | 80.40 | 80.06 | 53.58 |
| Coteaching | × | ✓ | ✓ | ✓ | 93.86 | 92.37 | 79.25 | 33.22 | 41.34 |

Table 5: Average training / test accuracy (%) of learning with three loss functions.

| | FMNIST | | | | CIFAR-100 | | | | mini-I.Net | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0.3 | | 0.7 | | 0.05 | | 0.1 | | ins.-dep. | |
| | training | test | training | test | training | test | training | test | training | test |
| $\ell_{\text{neg}}$ | 94.98 | 92.28 | 92.05 | 91.29 | 89.66 | 74.91 | 86.35 | 74.20 | 30.40 | 28.04 |
| $\ell_{\text{APL}}$ | 93.53 | 93.01 | 88.10 | 89.09 | 87.15 | 72.68 | 78.31 | 68.10 | 45.92 | 43.01 |
| $\ell_{\text{maxi}}$ | **95.39** | **93.77** | **93.01** | **92.17** | **93.45** | **78.91** | **90.56** | **78.20** | **55.39** | **54.72** |

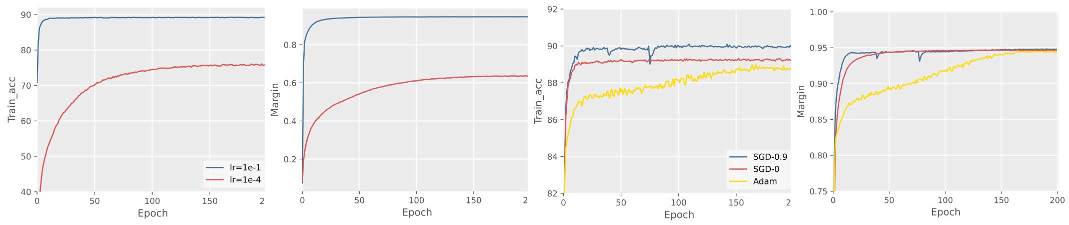


Figure 4: Training accuracy and confidence margin of predicted pseudo-labels for SASM with different learning rate or optimizer on FMNIST with PLs.

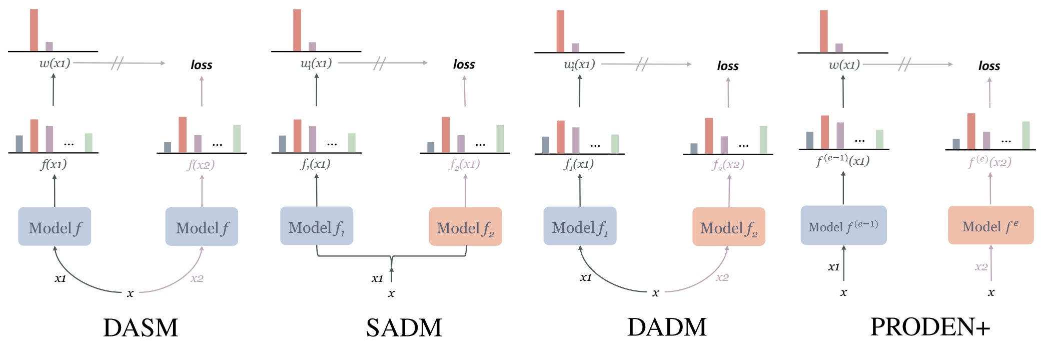


Figure 5: Illustrations of four effective units of SOTA PLL methods. We omit a symmetric loss from the other path, except PRODEN+ which can only compute one-path losses. ”//” means stop gradient.

## NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

**The checklist answers are an integral part of your paper submission.** They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: The paper discuss the limitations of the work performed by the authors.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper provide the full set of assumptions and a complete proof.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [NA]

   Justification: The paper focuses on understand existing algorithms rather than to fundamentally improve them.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specify all the training and test details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper report appropriate information about the statistical significance of the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper does not involve large-scale model training or extensive computational experiments.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper does not delve into the broader societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper has cited them appropriately.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper has cited them appropriately.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.