Probabilistic Instance Dependent Label Refinement for Noisy Label Learning

Hao-Yuan He, Yu Liu, Ren-Biao Liu, Zheng Xie, Ming Li*
Nanjing, Nanjing University
{hehy,liuy,liurb,xiez,lim}@lamda.nju.edu.cn

Abstract

Label refinement methods are designed to improve the quality of training labels by incorporating model predictions into the original training labels. By adjusting the combination coefficient of the noisy label, the impact of noise is reduced, which in turn makes the training process more robust. However, previous label refinement methods are unable to model instance-dependent noise, which is the most realistic type of noise. To address this limitation, we propose a simple approach, probabilistic instance-dependent label refinement (denoted as π-LR). Inspired by the fact that humans are more likely to make mistakes when annotating confusing instances, we propose to estimate the probability of whether a sample is confusing, which can be beneficial for modeling noise generation. Our approach utilizes this concept by assigning a confusing probability η_i to each instance $\boldsymbol{x}_i$ from a probabilistic perspective. This provides a clear understanding of how instance-dependent noise affects true labels. Empirical evaluations show that π-LR enhances the robustness of the model in the presence of label noise and outperforms all compared methods on both realistic and synthetic label noise datasets.

1 Introduction

Various models have been proposed to tackle the issue of label noise and comprehend its generation. The Random Classification Noise (RCN) model posits that labels are randomly corrupted [1, 2], while the Class-Conditional Noise (CCN) model suggests that certain pairs of classes are more susceptible to mislabeling [3, 4]. However, both models have been criticized for their simplicity in representing the complexities of real-world label noise. To overcome these limitations, researchers have proposed the Instance-Dependent Noise (IDN) model, which suggests that label noise is highly dependent on the instance's features [5].

Label refinement (also known as label refurbishment) is a commonly used technique in machine learning to refine and prevent overfitting to noisy labels. Typically, label refinement methods generate new training targets by computing a convex combination of the original training labels and the current model's predictions [6–8]. By adjusting the combination coefficient of the noisy label, the impact of noise can be reduced, which can make the training process more robust. Previous methods have determined the combination coefficient based on heuristic arguments concerning the different behaviors of clean and corrupted samples during the training process [9–11]. This coefficient is commonly referred to as the "clean probability" [7, 12, 13].

However, methods that rely on the concept of "clean probability" are inadequate for accounting for the noise generation process. In other words, if we already know the clean probability of samples, we can detect the noise. Meanwhile, these methods often require additional assumptions to estimate it, such as the well-known small loss assumption (i.e., samples with small loss tend to be clean) [14].

*Corresponding author

Preprint. Under review.

Hence, there is still a dearth of label refinement methods from the IDN perspective that can effectively handle more realistic scenarios.

In this study, our aim is to address the issue of noisy labels by incorporating a noise model based on the IDN assumption. We build upon the concept of "confusing samples" [15] and estimate the confusing probability, η_i, for each instance, $\boldsymbol{x}_i$. Essentially, an instance can only be mislabeled if it is a confusing sample. For example, a husky may be mistaken for a wolf, but a chihuahua is not. Instead of using the commonly employed concept of clean probability, estimating the confusing probability is more intuitive for understanding the source of the noise and is easier to estimate, as we will demonstrate in Section 2.3.

Overview of our method. Using this idea, we derive our main result, represented by

$$\boldsymbol{q}_i = \boldsymbol{v}_i \cdot \Big(\eta_i \cdot \hat{\boldsymbol{y}}_i + (1-\eta_i) \cdot \tilde{\boldsymbol{y}}_i \Big),$$

where Prediction by model, Potential noisy label, Instance transition ratio, and Confusing probability annotate the corresponding terms.

where $\boldsymbol{q}_i$ is the refined label, $\hat{\boldsymbol{y}}_i$ and $\tilde{\boldsymbol{y}}_i$ is the model's prediction and the noisy label respectively, and $\boldsymbol{v}_i \in \mathbb{R}^c$ is a vector that reflects the shift of class distribution induced by the label noise which we name it as the instance transition ratio in our study.

We use the above equation to refine potentially noisy labels. When the confusing probability η_i is high, the model's prediction will have a significant impact on the refined label. In contrast, when η_i is low, the refined label will remain the same as the original label from the training set. As we will show in Section 2.3, by introducing $\boldsymbol{v}_i$, the refined label can effectively model instance-dependent noise. Especially, when the noise is independent of the true label, $\boldsymbol{v}_i$ is a unit vector. We termed our approach probabilistic instance-dependent label refinement (π-LR for short).

Our contributions. (1) We propose π-LR, a novel probabilistic approach for addressing the problem of instance-dependent noise in labels. π-LR extends classical label refinement techniques to handle IDN problems by introducing the instance transition ratio $\boldsymbol{v}_i$. Unlike traditional clean probabilities used in label refinement techniques, π-LR introduces confusing probabilities, which are better suited for modeling IDN problems. Our method is highly adaptable and can be easily integrated with other approaches, such as semi-supervised learning or sample selection, to achieve even better performance.

(2) We evaluated the performance of π-LR on two realistic datasets, CIFAR-N [5] and Clothing1M [16], both of which contain instance-dependent noise. Our method outperformed other methods in all settings of realistic noise. In addition to the IDN setting, we also demonstrated that our method achieved comparable performance to most methods in this field under the CCN and RCN scenarios.

2 Instance Dependent Noise Modeling: A Probabilistic View

In this section, we propose a probabilistic framework π-LR for addressing the IDN problem. The core of π-LR is the derivation of the estimation of the true label based on the concept of "confusing probability" from a probabilistic perspective. To estimate this confusing probability, we introduce a Gaussian assumption as a reasonable approach. We then formulate the optimization objective using the evidence lower bound criterion and propose an alternating optimization algorithm as a solution to this objective. Finally, we summarize the overall algorithm in Algorithm 1.

2.1 Notations

In this study, we consider a dataset of training samples $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$, where each sample is associated with a true label $\boldsymbol{y}_i$ and a noisy label $\tilde{\boldsymbol{y}}_i$. The true labels are represented as $Y = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_N\}$ and the noisy labels as $\tilde{Y} = \{\tilde{\boldsymbol{y}}_1 \cdots, \tilde{\boldsymbol{y}}_N\}$. The dimension of the label space is c, where $\boldsymbol{y}_i \in \{0,1\}^c$. To better understand the source of noise in the labels, we introduce the concept of "confusing sample" [15].

Definition 2.1 (confusing sample)**.** A sample $\boldsymbol{x}_i$ is considered a confusing sample if and only if:

$$\Pr(\tilde{\boldsymbol{y}}_i = \boldsymbol{y}_i | \boldsymbol{x}_i, \boldsymbol{y}_i) < 1.$$

We denote s_i as:

$$\begin{cases} s_i = 1 & \text{if } \boldsymbol{x}_i \text{ is a confusing sample} \\ s_i = 0 & \text{if } \boldsymbol{x}_i \text{ is not a confusing sample} \end{cases}$$

In other words, a sample is considered a confusing sample if it is likely to be mislabeled. The confusing probability of a sample given its potentially noisy label is defined as $\Pr(s_i = 1|\tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i)$.

2.2 Estimation of true label distribution

In this section, we aim to derive the posterior distribution of the true label $\boldsymbol{q}_i$ given the sample $\boldsymbol{x}_i$. The estimated posterior probability of the true label is represented by the vector $\boldsymbol{q}_i = [\Pr(\boldsymbol{y}_i^1 = 1|\boldsymbol{x}_i), \cdots, \Pr(\boldsymbol{y}_i^c = 1|\boldsymbol{x}_i)]^\top$.

First, to estimate the individual probability $\boldsymbol{q}_i^j = \Pr(\boldsymbol{y}_i^j = 1|\boldsymbol{x}_i)$, we use Bayes' formula as follows:

$$\Pr(\boldsymbol{y}_i^j = 1|\boldsymbol{x}_i) = \frac{\Pr(\tilde{\boldsymbol{y}}_i, \boldsymbol{y}_i^j = 1|\boldsymbol{x}_i)}{\Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{y}_i^j = 1, \boldsymbol{x}_i)}. \tag{1}$$

Next, we expand $\Pr(\tilde{\boldsymbol{y}}_i, \boldsymbol{y}_i^j = 1|\boldsymbol{x}_i)$ as follows:

$$\Pr(\tilde{\boldsymbol{y}}_i, \boldsymbol{y}_i^j = 1|\boldsymbol{x}_i) = \Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{x}_i) \cdot \Pr(\boldsymbol{y}_i^j = 1|\tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i), \tag{2}$$

where $\Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{x}_i)$ is the posterior probability of the noisy label $\tilde{\boldsymbol{y}}_i$ given the sample $\boldsymbol{x}_i$. $\Pr(\boldsymbol{y}_i^j = 1|\tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i)$ represents the probability that the true label $\boldsymbol{y}_i$ corresponds to the j-th class, given the noisy label $\tilde{\boldsymbol{y}}_i$ and the sample $\boldsymbol{x}_i$. By introducing the concept of "confusing probability" and using η_i to represent $\Pr(s_i = 1|\tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i)$, we can expand $\Pr(\boldsymbol{y}_i^j = 1|\tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i)$ as follows:

$$\Pr(\boldsymbol{y}_i^j = 1|\tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i) = \Pr(\boldsymbol{y}_i^j = 1|s_i = 0, \tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i) \cdot (1 - \eta_i) + \Pr(\boldsymbol{y}_i^j = 1|s_i = 1, \tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i) \cdot \eta_i. \tag{3}$$

$\Pr(\boldsymbol{y}_i^j = 1|s_i = 0, \tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i)$ refers to the scenario where the sample $\boldsymbol{x}_i$ is not confusing, in which case we have $\boldsymbol{y}_i = \tilde{\boldsymbol{y}}_i$. This indicates that $\Pr(\boldsymbol{y}_i^j = 1|s_i = 0, \tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i) = \mathbb{I}(\boldsymbol{y}_i^j = \tilde{\boldsymbol{y}}_i^j) = \tilde{\boldsymbol{y}}_i^j$.

$\Pr(\boldsymbol{y}_i^j = 1|s_i = 1, \tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i)$ refers to the probability of the true label $\boldsymbol{y}_i$ belonging to the j-th class given $\boldsymbol{x}_i$, $\tilde{\boldsymbol{y}}_i$, and $s_i = 1$. In the context of noisy label learning, we can use a model (e.g., a neural network trained on $(S, \hat{Y})$) to estimate this probability, which we refer to as the model's prediction $\hat{\boldsymbol{y}}_i^j$. With this in mind, Equation 3 can be rewritten as follows:

$$\Pr(\boldsymbol{y}_i^j = 1|\tilde{\boldsymbol{y}}_i, \boldsymbol{x}_i) = (\eta_i \cdot \hat{\boldsymbol{y}}_i^j + (1 - \eta_i) \cdot \tilde{\boldsymbol{y}}_i^j). \tag{4}$$

Combining the above equations, $\boldsymbol{q}_i^j$ can be expressed as follows:

$$\boldsymbol{q}_i^j = \boldsymbol{v}_i^j \cdot (\eta_i \cdot \hat{\boldsymbol{y}}_i^j + (1 - \eta_i) \cdot \tilde{\boldsymbol{y}}_i^j), \tag{5}$$

where we use the notation $\boldsymbol{v}_i^j$ to represent the ratio between $\Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{x}_i)$ and $\Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{y}_i^j = 1, \boldsymbol{x}_i)$, which we refer to as the "instance transition ratio". This ratio reflects the shift of class distribution induced by the label noise. Finally, we can express the estimated true label $\boldsymbol{q}_i$ in vector form as follows:

$$\boldsymbol{q}_i = \boldsymbol{v}_i \cdot (\eta_i \cdot \hat{\boldsymbol{y}}_i + (1 - \eta_i) \cdot \tilde{\boldsymbol{y}}_i). \tag{6}$$

Instance transition ratio $\boldsymbol{v}_i$. In this study, we derive a new concept called "instance transition ratio" from the Bayes perspective. The denominator can be seen as a value that lies in the jth row of the instance-specific transition matrix.

$$\boldsymbol{v}_i^j = \frac{\Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{x}_i)}{\Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{y}_i^j = 1, \boldsymbol{x}_i)} \tag{7}$$

When the noise is independent of the ground truth label (i.e., the denominator is the same as the numerator), the instance transition ratio $\boldsymbol{v}_i$ is equal to $\mathbf{1}$. In this case, the posterior estimation of the ground truth label can be expressed as a linear combination of the predicted label and the noisy label, with coefficients determined by the confusing probability η_i. However, in the realistic world, the independence condition does not always hold, and consequently, the instance transition ratio is not equal to $\mathbf{1}$. Therefore, estimating the instance transition ratio can help improve the modeling of instance-specific noise. To mitigate this challenge, we treat the instance transition ratio as a learnable parameter and adopt an optimization process to estimate it.

Confusing probability η_i. When the noise is conditionally independent of the true label given $\boldsymbol{x}_i$, i.e., the instance transition ratio equals $\mathbf{1}$, the posterior estimation of the true label, $\boldsymbol{q}_i$, is given by

$$\boldsymbol{q}_i = (\eta_i \cdot \hat{\boldsymbol{y}}_i + (1 - \eta_i) \cdot \tilde{\boldsymbol{y}}_i)\,. \tag{8}$$

This equation is also known as "label refinement" in other literature, where η_i represents the probability that the sample is clean, which is often estimated using the model's prediction, the classification errors, the margin of samples, or other similar metrics [9–11].

For instance, Li et al.and Arazo et al. estimate the "clean probability" based on the small loss assumption [14]. Specifically, they use a two-component GMM to distinguish clean samples and corrupted samples from classification errors. Compared with estimating the clean probability based on heuristic metrics, using confusing probability has two advantages:

• Confusing probability is more *practical*. Modeling clean probability is equivalent to noise detection, which always relies on the small loss assumption. However, it is crucial to note that the small loss assumption is based on certain conditions [14], especially it may not hold when dealing with long-tail distributions. Also, it may suffer from confirmation bias, whereby the detection of noise depends on the model's output, leading the model to repeatedly confirm its prediction. In the following section, we will explore the concept of confusing probability, which is effective in handling these limitations.

• Confusing probability is more suitable for understanding the generation of noise. Using confusing probability, the generation process of noise labels is easy to understand. However, the clean probability is an ad-hoc perspective. Therefore, the confusing probability is better for modeling noise data.

2.3 Estimation of confusing probability

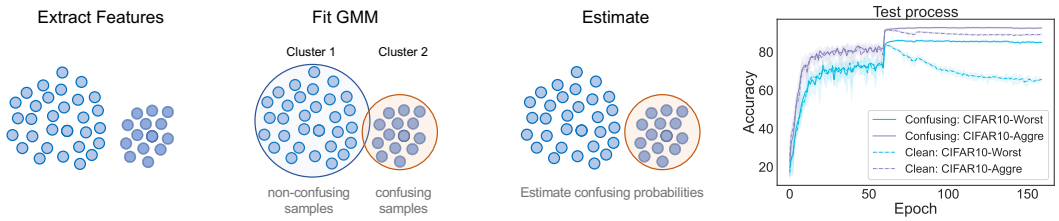

Figure 1: Left: The estimation process for one class, is repeated for all classes. Right: Performance comparison between our approach (using confusing probability) and previous approaches (using clean probability, c.f., [12]).

In this section, we estimate the confusing probability (as discussed in Section 2.2) instead of the clean probability (i.e., whether a sample's label is correct or not) to mitigate the issue of confirmation bias. However, estimating the confusing probabilities can be challenging due to the lack of supervision information. To tackle this challenge, we make an assumption that the distributions of confusing and non-confusing samples follow distinct Gaussian models. This assumption allows us to utilize a GMM for estimating the confusing probability without any supervision.

The estimation process is illustrated in Figure 1. We conduct the estimation process individually for each class indicated by the labels in the training set. The feature extractor is obtained from a trained neural network. It is crucial to note that the GMM fits two clusters, and one can choose one of them as a confusing probability estimator based on their assumptions. For instance, in this case, we assume that non-confusing samples are typically located closer to the class center. Therefore, we can select an estimator based on the distance between the cluster center and the class center.

The performance of a model is significantly reduced when using clean probability, as illustrated in Figure 1. Here we estimate the clean probability based on the model's classification error [12],

with the assumption that clean samples tend to have lower classification error (a.k.a., small loss assumption). Specifically, we changed the clean probability estimation to the confusing probability while keeping the remaining process the same. Due to the impact of confirmation bias, the model tends to confirm its prediction as correct, resulting in reduced performance. Conversely, estimating the confusing probability relies solely on the feature space of the model, which is more stable during training and more resilient to noisy labels, resulting in better performance.

2.4 Optimization object

Given the parameter $\Theta = \{\boldsymbol{v}_i | i \in [N]\} \cup \{\mathbf{w}\}$, where $\{\boldsymbol{v}_i | i \in [N]\}$ is approximate instance transition ratio and $\mathbf{w}$ is the parameters of the neural network, the learning process can be seen as the maximization of the log-likelihood:$\ell(\Theta) = \sum_{i\in[N]} \log \Pr(\tilde{\boldsymbol{y}}_i | \boldsymbol{x}_i; \Theta)$. However, this formula is hard to optimize because the inference process of the posterior is high-complexity. For efficient optimization, we would like to obtain the lower bound of this object, the log-likelihood $\ell(\Theta)$ can be written as:

$$\ell(\Theta) = \mathbb{E}_{i\in[N]} \log \mathbb{E}_{j\in[c]} [\Pr(\tilde{\boldsymbol{y}}_i, \boldsymbol{y}_i^j = 1 | \boldsymbol{x}_i; \Theta) \cdot \frac{\boldsymbol{q}_i^j}{\boldsymbol{q}_i^j}] \tag{9}$$

$$\geq \mathbb{E}_{i\in[N], j\in[c]} \left[\boldsymbol{q}_i^j \cdot \log[\Pr(\tilde{\boldsymbol{y}}_i, \boldsymbol{y}_i^j = 1 | \boldsymbol{x}_i; \Theta) / \boldsymbol{q}_i^j] \right] \tag{10}$$

$$= \mathbb{E}_{i\in[N], j\in[c]} \left[\boldsymbol{q}_i^j \cdot \log[\Pr(\tilde{\boldsymbol{y}}_i, \boldsymbol{y}_i^j = 1 | \boldsymbol{x}_i; \Theta)] \right] + \text{const.} \tag{11}$$

Combine Equation 9 and Equation 10, using Jensen's inequality, we can derive a lower-bound of primitive optimization object:

$$\arg\max_{\Theta} \sum_{i\in[N]} \sum_{j\in[c]} \boldsymbol{q}_i^j \log \Pr(\tilde{\boldsymbol{y}}_i, \boldsymbol{y}_i^j = 1 | \boldsymbol{x}_i; \Theta). \tag{12}$$

2.5 Alternating optimization

As shown in section 2.2, the posterior $\boldsymbol{q}_i$ is dependent on the prediction of the model and the estimation of η_i, which is related to Θ. Thus the optimization framework is naturally alternatively updated with the posterior $\boldsymbol{q}_i$ and parameters Θ.

Predicting step. Refer to the discussion in the section 2.2, the posterior $\boldsymbol{q}_i$ is predict as:

$$\boldsymbol{q}_i = \boldsymbol{v}_i \cdot (\eta_i \cdot \hat{\boldsymbol{y}}_i + (1 - \eta_i) \cdot \tilde{\boldsymbol{y}}_i).$$

$\boldsymbol{q}_i$ is a refined label $\eta_i \cdot \hat{\boldsymbol{y}}_i + (1 - \eta_i) \cdot \tilde{\boldsymbol{y}}_i$ further multiply by the "instance transition ratio" $\boldsymbol{v}_i$.

Updating step. The updating step is optimized by minimizing the following given loss functions through stochastic gradient descent. The loss used in this part is composed of the following parts.

• The classification loss with refinement posterior $\boldsymbol{q_i}$. This term in the loss function aims to minimize the classification error with respect to the refined target $\boldsymbol{q}_i$.

$$\mathcal{L}_c = \frac{1}{N} \cdot \sum_{i\in[N]} \text{CrossEntropy}(\boldsymbol{q_i}, \hat{\boldsymbol{y}}_i) \tag{13}$$

• The regularization part refers to ELR [17]. To mitigate the memorization effects of neural networks [17] during training, we introduce this loss component. The vector $\mathbf{t}_i$ is the historical smoothed version of $\hat{\boldsymbol{y}}_i$, generated using the exponential moving average (EMA) technique, a commonly used approach in the field of NLL. The function $\langle \cdot, \cdot \rangle$ calculates the inner product, which can be interpreted as the similarity between two vectors.

$$\mathcal{L}_r = \frac{1}{N} \cdot \sum_{i\in[N]} \log(1 - \langle \hat{\boldsymbol{y}}_i, \mathbf{t}_i \rangle) \tag{14}$$

• The neg-likelihood derived from optimization Equation 12 with Equation 2 and 4. This term of the loss function aims to optimize the instance transition ratio $\boldsymbol{v}_i$. Here we denote the posterior $\Pr(\tilde{\boldsymbol{y}}_i | \boldsymbol{x}_i)$

as ψ_i, which can be estimated by a pre-trained model's prediction (details are shown in the appendix). Specifically, this can be seen as a one-step maximization process in the expectation-maximization (EM) algorithm.

$$\mathcal{L}_v = -\frac{1}{N \cdot c} \sum_{i \in [N]} \sum_{j \in [c]} \boldsymbol{q}_i^j \log \left(\psi_i \cdot (\eta_i \cdot \hat{\boldsymbol{y}}_i + (1 - \eta_i) \cdot \tilde{\boldsymbol{y}}_i) \right) \tag{15}$$

Updating rules. For the parameters $\mathbf{w}$ in the neural network, the updating step follows common gradient descent with respect to $\mathcal{L}_c + \lambda \mathcal{L}_r$, where λ is a tunable coefficient. The optimization Equation 12 can be solved by projected gradient descent, as the instance transition ratio, $\boldsymbol{v}_i$, is naturally subject to the convex set $\boldsymbol{v}_i : \boldsymbol{v}_i \succ \mathbf{0}, i \in [N]$. In each step, we first update each $\boldsymbol{v}_i$ by gradient descent and then project each $\boldsymbol{v}_i$ to the constraint set, where α_1 is the learning rate.

$$\boldsymbol{v}_i \leftarrow \text{ElementMax}(\mathbf{0}, \boldsymbol{v}_i - \alpha_1 \nabla_{\boldsymbol{v}_i} \mathcal{L}_v) \tag{16}$$

2.6 Overall algorithm

In the alternating optimization approach, the initial predictions of the model may not be reliable. To address this issue, the initial values of η_i are set to zero. This means that the posterior $\boldsymbol{q}_i$ is primarily determined by the noisy label, in order to avoid unreliable model predictions at the beginning of the training process. Additionally, it is a well-established fact that deep neural networks (DNNs) are less prone to learning from label noise during the early training phase when the learning rate is high [18]. Therefore, a high learning rate is used during the initial phase of the training process to warm up the model.

As the training process continues, the feature extractor of the model becomes more reliable and we can use it to estimate the confusing probability of the noisy labels. It is important to note that after the estimation of the confusing probability, the target $\boldsymbol{q}_i$ is refined, which leads to further improvement in the model's performance. The overall learning process is summarized in Algorithm 1, which is both straightforward and efficient.

Algorithm 1 Overall Algorithm of π-LR

Input: training set $(S, \tilde{Y})$; estimated probabilities $\{\psi_1, \cdots, \psi_N\}$; training steps *num_steps*; hyper-parameter λ; step list of estimation $\mathcal{T}$;
Output: the optimal parameters Θ
Initialize $\eta_i = 0, \forall i$;
Initialize $\Theta = \{\boldsymbol{v}_i | i \in [N]\} \cup \{\mathbf{w}\}$;
for $t = 1$ **to** *num_steps* **do**
  $\boldsymbol{q} = \boldsymbol{v} \cdot (\eta \cdot \hat{\boldsymbol{y}} + (1 - \eta) \cdot \tilde{\boldsymbol{y}})$
  Compute $\mathcal{L}_c$ c.f., Equation 13
  Compute $\mathcal{L}_r$ c.f., Equation 14
  Compute $\mathcal{L}_v$ c.f., Equation 15
  Update Θ c.f., Section 2.5
  if $t \in \mathcal{T}$ **then**
    estimate $\{\eta_i | i \in [N]\}$ c.f., Section 2.3
  end if
end for

3 Experiments

3.1 Experimental setup

We first evaluate the effectiveness of π-LR on two realistic noisy datasets. Then, we further test the performance of π-LR on the synthetic noise tasks as in previous works.

Realistic label noise. Two realistic noisy datasets are adopted for experiments.

(1) CIFAR-N [5] is consist of CIFAR-10N and CIFAR-100-N, which are human-annotated versions of the CIFAR-10 and CIFAR-100 datasets [25]. The CIFAR-10N dataset includes five different noise settings: Aggregate, Random 1, Random 2, Random 3, and Worst, with noise rates of 9.03%, 17.23%, 18.12%, 17.64%, and 40.21% respectively. Specifically, each image in CIFAR-10N contains three submitted labels (i.e., Random 1,2,3) which are further combined to create an Aggregate and a Worst label. The CIFAR-100-N dataset has one noise setting, named Noisy, with a 40.20% noise rate.

(2) Clothing1M [16] is a large-scale realistic noisy dataset consisting of more than one million images from 14 classes with roughly 40% wrong labels. Images were obtained from several online shopping websites and labels were generated by their surrounding texts. For a fair comparison, as the previous works, we do not use strong data augmentations.

Table 1: Test accuracies on CIFAR-N with realistic label noise. All methods use ResNet34 as the backbone. Mean and standard deviation over 5 independent runs with seeds $\{2021, 2022, 2023, 2024, 2025\}$ are reported.

Methods	CIFAR-10N Random 1	Random 2	Random 3	Aggregate	Worst	CIFAR-100N Noisy
CE	$85.02_{\pm 0.65}$	$86.46_{\pm 1.79}$	$85.16_{\pm 0.61}$	$87.77_{\pm 0.38}$	$77.69_{\pm 1.55}$	$55.50_{\pm 0.66}$
Forward [3]	$86.88_{\pm 0.50}$	$86.14_{\pm 0.24}$	$87.04_{\pm 0.35}$	$88.24_{\pm 0.22}$	$79.79_{\pm 0.46}$	$57.01_{\pm 1.03}$
Backward [3]	$87.14_{\pm 0.34}$	$86.28_{\pm 0.80}$	$86.86_{\pm 0.41}$	$88.13_{\pm 0.29}$	$77.61_{\pm 1.05}$	$57.14_{\pm 0.92}$
GCE [19]	$87.61_{\pm 0.28}$	$87.70_{\pm 0.56}$	$87.58_{\pm 0.29}$	$87.85_{\pm 0.70}$	$80.66_{\pm 0.35}$	$56.73_{\pm 0.30}$
Peer Loss [20]	$89.06_{\pm 0.11}$	$88.76_{\pm 0.19}$	$88.57_{\pm 0.09}$	$90.75_{\pm 0.25}$	$82.53_{\pm 0.52}$	$57.59_{\pm 0.61}$
VolMinNet [21]	$88.30_{\pm 0.12}$	$88.27_{\pm 0.09}$	$88.19_{\pm 0.41}$	$89.70_{\pm 0.21}$	$80.53_{\pm 0.20}$	$57.80_{\pm 0.31}$
F-div [22]	$89.70_{\pm 0.40}$	$89.79_{\pm 0.12}$	$89.55_{\pm 0.49}$	$91.64_{\pm 0.34}$	$82.53_{\pm 0.52}$	$57.10_{\pm 0.65}$
ELR [17]	$91.46_{\pm 0.38}$	$91.61_{\pm 0.16}$	$91.41_{\pm 0.44}$	$92.38_{\pm 0.64}$	$83.58_{\pm 1.13}$	$58.94_{\pm 0.92}$
CAL [23]	$90.93_{\pm 0.31}$	$90.75_{\pm 0.30}$	$90.74_{\pm 0.24}$	$91.97_{\pm 0.32}$	$85.36_{\pm 0.16}$	$61.73_{\pm 0.42}$
CORES^2 [24]	$89.66_{\pm 0.32}$	$89.91_{\pm 0.45}$	$89.79_{\pm 0.50}$	$91.23_{\pm 0.11}$	$83.60_{\pm 0.53}$	$61.15_{\pm 0.73}$
π-LR (w/o. CP)	$90.17_{\pm 0.30}$	$90.16_{\pm 0.16}$	$90.02_{\pm 0.15}$	$91.36_{\pm 0.20}$	$83.85_{\pm 0.62}$	$60.51_{\pm 0.49}$
π-LR (w/o. $\mathcal{L}_r$)	$90.58_{\pm 0.13}$	$90.81_{\pm 0.25}$	$90.72_{\pm 0.25}$	$92.11_{\pm 0.20}$	$84.18_{\pm 0.13}$	$62.13_{\pm 0.25}$
π-LR (w/o. $\mathcal{L}_v$)	$91.48_{\pm 0.27}$	$91.59_{\pm 0.41}$	$91.78_{\pm 0.16}$	$92.89_{\pm 0.24}$	$86.28_{\pm 0.30}$	$62.47_{\pm 0.63}$
π-LR (Ours)	$\mathbf{92.02}_{\pm 0.32}$	$\mathbf{91.96}_{\pm 0.28}$	$\mathbf{92.09}_{\pm 0.12}$	$\mathbf{92.99}_{\pm 0.24}$	$\mathbf{86.76}_{\pm 0.42}$	$\mathbf{62.73}_{\pm 0.46}$

Synthetic label noise. As the commonly used setting in the previous works, we also adopt CIFAR-10 and CIFAR-100 with symmetric and asymmetric noise types with different noise ratios.

Baselines. Baselines are adopted from two perspectives:

(1) Methods that focus on learning with noisy labels, such as estimation of the transition matrix such as Forward-T/Backward-T [3] and VolMinNet [21], design of robust loss functions like GCE [19], Peer Loss [20], F-div [22], CAL [23], SL [28], and label noise correction using techniques like ELR [17], PM [15], and CORES^2 [24].

(2) Methods that use semi-supervised technologies (e.g., consistency regularization), ensemble two models for sample selection, or both, such as DivideMix [12], Co-Teaching [26], JoCoR [29], CORES^{2*} [24], SOP+[27], and ProMix [30].

The results of these methods are based on their original papers, and we use the same training backbone and data pre-processing for a fair comparison. We conducted our experiments using an NVIDIA Tesla A100, and detailed information about the experimental setup can be found in the appendix.

Table 2: Test accuracies on Clothing-1M with realistic label noise.

Methods	Accuracy(%)
Bootstrapping [6]	67.6
Forward [3]	69.8
Co-teaching [26]	69.2
PENCIL [18]	73.5
ELR [17]	72.9
CORES^2 [24]	73.2
PM [15]	74.0
SOP [27]	73.5
π-LR (Ours)	**74.3**

Table 3: Test accuracies on CIFAR-N with realistic label noise, semi-supervised scenario. Comparison with methods that use two network ensembles and semi-supervised technologies. Mean and standard deviation over 5 independent runs with seeds $\{2021, 2022, 2023, 2024, 2025\}$ are reported.

Methods	CIFAR-10N Random 1	Random 2	Random 3	Aggregate	Worst	CIFAR-100N Noisy
JoCoR [29]	$90.30_{\pm 0.20}$	$90.21_{\pm 0.19}$	$90.11_{\pm 0.21}$	$91.44_{\pm 0.05}$	$83.37_{\pm 0.30}$	$59.97_{\pm 0.24}$
Co-Teaching [26]	$90.33_{\pm 0.13}$	$90.30_{\pm 0.17}$	$90.15_{\pm 0.18}$	$91.20_{\pm 0.13}$	$83.83_{\pm 0.13}$	$60.37_{\pm 0.27}$
DivideMix [12]	$95.16_{\pm 0.19}$	$95.23_{\pm 0.07}$	$95.21_{\pm 0.14}$	$95.01_{\pm 0.71}$	$92.56_{\pm 0.42}$	$71.13_{\pm 0.48}$
CORES^{2*} [24]	$94.45_{\pm 0.14}$	$95.23_{\pm 0.07}$	$94.74_{\pm 0.03}$	$95.25_{\pm 0.09}$	$91.66_{\pm 0.09}$	$55.72_{\pm 0.42}$
SOP+[27]	$95.28_{\pm 0.13}$	$95.31_{\pm 0.10}$	$95.39_{\pm 0.11}$	$95.61_{\pm 0.13}$	$93.24_{\pm 0.21}$	$67.81_{\pm 0.23}$
ProMix [30]	$96.89_{\pm 0.05}$	-	-	$97.08_{\pm 0.06}$	$95.91_{\pm 0.25}$	$72.74_{\pm 0.16}$
π-LR+ (Ours)	$\mathbf{96.99}_{\pm 0.03}$	$\mathbf{96.98}_{\pm 0.10}$	$\mathbf{97.07}_{\pm 0.05}$	$\mathbf{97.28}_{\pm 0.10}$	$\mathbf{95.92}_{\pm 0.11}$	$\mathbf{73.11}_{\pm 0.07}$

Table 4: Test accuracies with synthetic label noise. All methods use ResNet34 as the backbone. Mean and standard deviation over 5 independent runs with seeds $\{2021, 2022, 2023, 2024, 2025\}$ are reported.

	CIFAR-10			CIFAR-100		
Methods	Symm. 0.4	Symm. 0.6	Asym. 0.3	Symm. 0.4	Symm. 0.6	Asym. 0.3
CE	$81.88_{\pm 0.29}$	$74.14_{\pm 0.56}$	$86.14_{\pm 0.40}$	$48.20_{\pm 0.65}$	$37.41_{\pm 0.94}$	$51.40_{\pm 0.16}$
Forward [3]	$83.25_{\pm 0.38}$	$74.96_{\pm 0.65}$	$86.79_{\pm 0.36}$	$31.05_{\pm 1.44}$	$19.12_{\pm 1.95}$	$38.13_{\pm 2.97}$
GCE [19]	$87.13_{\pm 0.22}$	$82.54_{\pm 0.23}$	$85.45_{\pm 0.74}$	$61.77_{\pm 0.24}$	$53.16_{\pm 0.78}$	$61.45_{\pm 0.26}$
SL [28]	$87.13_{\pm 0.26}$	$82.81_{\pm 0.61}$	$88.48_{\pm 0.46}$	$62.27_{\pm 0.22}$	$54.82_{\pm 0.57}$	$72.12_{\pm 0.24}$
ELR [17]	$89.15_{\pm 0.17}$	$86.12_{\pm 0.49}$	$91.89_{\pm 0.22}$	$68.28_{\pm 0.31}$	$59.28_{\pm 0.67}$	$73.71_{\pm 0.22}$
π-LR (Ours)	$\mathbf{89.92}_{\pm 0.38}$	$\mathbf{86.44}_{\pm 0.23}$	$\mathbf{92.91}_{\pm 0.30}$	$\mathbf{73.01}_{\pm 0.52}$	$\mathbf{72.95}_{\pm 0.51}$	$\mathbf{73.98}_{\pm 0.85}$

3.2 Empirical results

Realistic label noise, CIFAR-N. Table 1 reports the performance of our method with realistic instance-dependent label noise on CIFAR-N. It can be seen that our method is robust to this realistic instance-dependent label noise setting and achieve superior performance. For a fair comparison, this table does not involve methods that use extra information provided by strong data augmentation and semi-supervised learning.

We report three kinds of ablations of our method in Table 1, (a) setting confusing probability to 0 for all samples (w/o. CP); (b) without using regularization part $\mathcal{L}_r$ (w/o. $\mathcal{L}_r$); (c) without updating the instance transition ratio, i.e., setting $\boldsymbol{v}_i$ to $\mathbf{1}$, (w/o. $\mathcal{L}_v$).

For further improved performance, we report the performance of our method with additional information in Table 3. With semi-supervised technology and the two-model ensemble, our method can achieve state-of-the-art. Details of the implementations can be seen in the appendix.

Realistic label noise, Clothing1M. Table 2 shows the performance of our method with real-life noise on the Clothing1M dataset. Due to the dataset's large size, we did not adopt a semi-supervised scenario under this setting. As demonstrated in the table, our simple method achieves comparable performance with state-of-the-art methods.

Synthesis label noise. Table 4 reports the performance of π-LR on CIFAR-10 and CIFAR-100 with different levels of symmetric and asymmetric label noise. Even though π-LR is based on the IDN assumption, which is not true in this setting, it is still comparable to most of the methods in the field. Moreover, this experiment further proves the effectiveness of a probabilistic view of noise modeling when dealing with IDN scenarios.

It is important to emphasize that the primary objective of this research is not solely to achieve state-of-the-art results, but rather to contribute to the advancement of the field by: (a) Providing a novel framework to extend the existing label refinement technologies in IDN scenarios. (b) Highlighting the effectiveness of utilizing a probabilistic perspective for modeling instance-dependent noise.

4 Related Work and Discussions

4.1 Related works

Our paper is closely aligned with the topic of label refinement technology and the concept of confusing samples [15]. In this section, we will provide an overview of label refinement technology, followed by an exploration of other popular methods within this area of research. For a comprehensive introduction to the NLL field, we recommend the following surveys [31, 32].

Label refinement . The ability to effectively refine noisy labels is crucial in preventing overfitting to false labels. One of the pioneering methods that introduced the concept of label refinement is Bootstrapping [6], which provides the learner with justification to "disagree" with a perceptually-inconsistent training label. Specifically, the learner bootstraps itself by using a convex combination of training labels and the current model's predictions to generate the training targets. Subsequent methods have aimed to improve upon this technique, such as Dynamic Bootstrapping [7], which

adapts the confidence α of individual training examples on the fly, and Self-adaptive training [8], which addresses the instability issue of using instantaneous predictions of the current deep neural network (DNN) by utilizing exponential moving averages. Previous techniques for label refinement have primarily been based on heuristic arguments concerning the different behaviors of clean and corrupted samples during the training process. These include but are not limited to properties of learned representations [9], prediction consistency [10], and margin [11]. Besides, AdaCorr [33] selectively refine the noisy label and provides a theoretical error-bound. Our proposed framework, π-LR, generalizes the form of label refinement by introducing the instance transition ratio term and utilizing confusing probability in place of clean probability or other heuristic arguments. This allows π-LR to provide more comprehensive modeling of instance-dependent noise (IDN) problems.

Other techniques. The noise transition matrix [34] describes a corruption process that transitions from the latent clean label $\boldsymbol{y}$ to the observed noisy label $\tilde{\boldsymbol{y}}$. This process is represented by a *noise transition matrix* T, where $T(i,j) = \Pr(\tilde{\boldsymbol{y}}^j = 1 | \boldsymbol{y}^i = 1)$. Several works aim to estimate the transition matrix [2, 35–38] for loss correction. Considering that deep neural networks (DNNs) tend to overfit to label noise [39], other techniques focus on designing loss functions and avoiding model overfitting [19, 28, 40–42]. These techniques aim to leverage DNNs' memorization capacity. For instance, the small-loss criterion [12] filters out a clean subset from the noisy origin examples based on this principle to alleviate the overfitting problem. The Co-teaching method [26] utilizes the simultaneous training of two networks with different initializations to facilitate the teaching process between the networks by selecting clean examples. DivideMix [12] employs two models to separate samples with the MixMatch [43] strategy and leverage unselected examples. Similar ideas have been adopted by other recent studies, including [44–49]. Also, adversarial training is another approach to regularize the behavior of the model [50, 51].

4.2 Discussions

Confusing probability. In this paper, we utilize this concept and estimate it using a two-component Gaussian mixture model (GMM) based on the assumption that non-confusing samples are typically located closer to the class center. Nevertheless, we would like to emphasize that this assumption only holds when the noise type is instance-dependent, our method may not perform optimally when the noise type changes. In such cases, combining our method with techniques to prevent overfitting or memorization effects [9, 17] could be helpful. Besides, the estimation process of the confusing probability relies on the GMM, which may become inefficient as the amount of training samples increases. Therefore, an efficient estimation method is a potential future research direction.

Instance transition ratio. The estimation of the instance transition ratio $\boldsymbol{v}_i$ is a crucial aspect of our proposed method. However, the current implementation using Projected Gradient Descent (PGD) lacks convergence guarantees for the optimal solution. We have observed that the denominator of $\boldsymbol{v}_i^j = \Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{x}_i) / \Pr(\tilde{\boldsymbol{y}}_i|\boldsymbol{y}_i^j = 1, \boldsymbol{x}_i)$ is closely related to an instance-dependent transition matrix T_i. Here, $T_i(a,b)$ represents the probability of $\Pr(\tilde{\boldsymbol{y}}_i^b = 1 | \boldsymbol{y}_i^a = 1, \boldsymbol{x}_i)$. Incorporating instance-dependent transition matrix estimation techniques in our framework is a promising approach to improve performance. However, the estimation problem could be ill-posed, leading to numerical overflow issues, as the denominator requires non-zero values. Therefore, the estimation of $\boldsymbol{v}_i$ remains a challenge and an important future direction in this framework.

5 Conclusion

In this study, we introduce a novel probabilistic framework called π-LR, which addresses the problem of instance-dependent noise (IDN) and extends the scope of previous label refinement technologies. π-LR leverages both the confusing probability η_i and the instance transition ratio $\boldsymbol{v}_i$ to refine the potentially noisy label $\tilde{\boldsymbol{y}}_i$ and construct a posterior distribution of true labels $\boldsymbol{q}_i$ under IDN. By estimating the true label distribution $\boldsymbol{q}_i$, the training process becomes more robust and flexible, allowing our method to seamlessly integrate with other technologies such as consistency regularization and sample selection. The future directions include further improving the optimization process of the instance transition ratio $\boldsymbol{v}_i$ and investigating a more efficient estimation process of the confusing probability η_i.

References

[1] Naresh Manwani and P. S. Sastry. Noise tolerance under risk minimization. *IEEE Transactions on Cybernetics*, 2013.

[2] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Conference on Neural Information Processing Systems*, 2015.

[3] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[4] Pengfei Chen, Junjie Ye, Guangyong Chen, Jingwei Zhao, and Pheng-Ann Heng. Robustness of accuracy metric and its inspirations in learning with noisy labels. *CoRR*, 2020.

[5] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations*, 2022.

[6] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *International Conference on Learning Representations*, 2015.

[7] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. *CoRR*, 2019.

[8] Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. In *Conference on Neural Information Processing Systems*, 2020.

[9] Taehyeon Kim, Jongwoo Ko, JinHwan Choi, Se-Young Yun, et al. Fine samples for learning with noisy labels. In *Conference on Neural Information Processing Systems*, 2021.

[10] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. Selfie: Refurbishing unclean samples for robust deep learning. In *International Conference on Machine Learning*, 2019.

[11] Jason Z Lin and Jelena Bradic. Learning to combat noisy labels via classification margins. *CoRR*, 2021.

[12] Junnan Li, Richard Socher, and Steven C. H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

[13] Mingcai Chen, Hao Cheng, Yuntao Du, Ming Xu, Wenyu Jiang, and Chongjun Wang. Two wrongs don't make a right: Combating confirmation bias in learning with label noise. *CoRR*, 2021.

[14] Xian-Jin Gui, Wei Wang, and Zhang-Hao Tian. Towards understanding deep learning from noisy labels with small-loss criterion. In *International Joint Conference on Artificial Intelligence*, 2021.

[15] Qizhou Wang, Bo Han, Tongliang Liu, Gang Niu, Jian Yang, and Chen Gong. Tackling instance-dependent label noise via a universal probabilistic model. In *AAAI Conference on Artificial Intelligence*, 2021.

[16] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[17] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. In *Conference on Neural Information Processing Systems*, 2020.

[18] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[19] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Conference on Neural Information Processing Systems*, 2018.

[20] Yang Liu and Hongyi Guo. Peer loss functions: Learning from noisy labels without knowing noise rates. In *International Conference on Machine Learning*, 2020.

[21] Xuefeng Li, Tongliang Liu, Bo Han, Gang Niu, and Masashi Sugiyama. Provably end-to-end label-noise learning without anchor points. In *International Conference on Machine Learning*, 2021.

[22] Jiaheng Wei and Yang Liu. When optimizing f-divergence is robust with label noise. In *International Conference on Learning Representations*, 2021.

[23] Zhaowei Zhu, Tongliang Liu, and Yang Liu. A second-order approach to learning with instance-dependent label noise. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[24] Hao Cheng, Zhaowei Zhu, Xingyu Li, Yifei Gong, Xing Sun, and Yang Liu. Learning with instance-dependent label noise: A sample sieve approach. In *International Conference on Learning Representations*, 2021.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[26] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Conference on Neural Information Processing Systems*, 2018.

[27] Sheng Liu, Zhihui Zhu, Qing Qu, and Chong You. Robust training under label noise by over-parameterization. In *International Conference on Machine Learning*, 2022.

[28] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *International Conference on Computer Vision*, 2019.

[29] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[30] Haobo Wang, Ruixuan Xiao, Yiwen Dong, Lei Feng, and Junbo Zhao. Promix: Combating label noise via maximizing clean sample utility. *CoRR*, 2022.

[31] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *CoRR*, 2020.

[32] Bo Han, Quanming Yao, Tongliang Liu, Gang Niu, Ivor W Tsang, James T Kwok, and Masashi Sugiyama. A survey of label-noise representation learning: Past, present and future. *CoRR*, 2020.

[33] Songzhu Zheng, Pengxiang Wu, Aman Goswami, Mayank Goswami, Dimitris N. Metaxas, and Chao Chen. Error-bounded correction of noisy labels. In *International Conference on Machine Learning*, 2020.

[34] Brendan van Rooyen and Robert C Williamson. A theory of learning with corrupted labels. *Journal of Machine Learning Research*, 2017.

[35] Bo Han, Jiangchao Yao, Gang Niu, Mingyuan Zhou, Ivor Tsang, Ya Zhang, and Masashi Sugiyama. Masking: A new perspective of noisy supervision. In *Conference on Neural Information Processing Systems*, 2018.

[36] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[37] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Conference on Neural Information Processing Systems*, 2018.

[38] Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? In *Conference on Neural Information Processing Systems*, 2019.

[39] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[40] Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. Robust loss functions under label noise for deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2017.

[41] Xinshao Wang, Elyor Kodirov, Yang Hua, and Neil Martin Robertson. Improving MAE against CCE under label noise. *CoRR*, 2019.

[42] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels. In *International Conference on Machine Learning*, 2020.

[43] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Conference on Neural Information Processing Systems*, 2019.

[44] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. Self: Learning to filter noisy labels with self-ensembling. In *International Conference on Learning Representations*, 2020.

[45] Hao Cheng, Zhaowei Zhu, Xing Sun, and Yang Liu. Mitigating memorization of noisy labels via regularization between representations. In *International Conference on Learning Representations*, 2023.

[46] Yan Yan and Yuhong Guo. Mutual partial label learning with competitive label noise. In *International Conference on Learning Representations*, 2023.

[47] Jichang Li, Guanbin Li, Feng Liu, and Yizhou Yu. Neighborhood collective estimation for noisy label identification and correction. In *European Conference on Computer Vision*, 2022.

[48] Tohar Lukov, Na Zhao, Gim Hee Lee, and Ser-Nam Lim. Teaching with soft label smoothing for mitigating noisy labels in facial expressions. In *European Conference on Computer Vision*, 2022.

[49] Chao Liang, Zongxin Yang, Linchao Zhu, and Yi Yang. Co-learning meets stitch-up for noisy multi-label visual recognition. *IEEE Transactions on Image Processing*, 2023.

[50] Chengyu Dong, Liyuan Liu, and Jingbo Shang. Label noise in adversarial training: A novel perspective to study robust overfitting. In *Conference on Neural Information Processing Systems*, 2022.

[51] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[52] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Conference on Neural Information Processing Systems*, 2020.

[53] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical data augmentation with no separate search. *CoRR*, 2019.

[54] Ethan Harris, Antonia Marcu, Matthew Painter, Mahesan Niranjan, Adam Prügel-Bennett, and Jonathon S. Hare. Understanding and enhancing mixed sample data augmentation. *CoRR*, 2020.

[55] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Conference on Neural Information Processing Systems*, 2020.

Appendices

This appendix is organized as follows. In Section A we provide additional details for reproducing experimental results. In Section B we provide more empirical studies of this method. In Section C we discuss of efficiency of our method.

A Experimental Details

A.1 Generation of synthetic label noise

In this section, we introduce how the synthetic label noise setup in our study. Specifically, we employed Algorithm 2 as previous works [17, 27] to generate a dataset with synthetic noisy labels.

Algorithm 2 Generate Synthetic Label Noise to CIFAR-10 / CIFAR-100

Input: dataset $\mathcal{D}$, noise rate r, noise type t, pre-defined transformation $\mathcal{T}$
Output: dataset with synthetic label $\mathcal{D}^{\text{noise}}$
if noise type t is symmetry **then**
  for each class i **do**
    Randomly select $\lceil r \cdot n_i \rceil$ samples $\mathcal{S}_i^{\text{sym}}$, where n_i is the number of samples in class i
    for s in $\mathcal{S}_i^{\text{sym}}$ **do**
      Random assign a label to s.
    end for
  end for
end if
if noise type t is asymmetry **then**
  for each class i **do**
    Randomly select $\lceil r \cdot n_i \rceil$ samples $\mathcal{S}_i^{\text{asm}}$, where n_i is the number of samples in class i
    for s in $\mathcal{S}_i^{\text{asm}}$ **do**
      Assign a new label to s based on the pre-defined transformation $\mathcal{T}$.
    end for
  end for
end if

Transformation under the asymmetry label noise. For CIFAR-10, we flip labels as follows: TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, CAT $\leftrightarrow$ DOG, while the other classes remain unchanged. For CIFAR-100, we flip the class label within the same super-class in the CIFAR-100. Specifically, the transformation can be expressed as: $a \rightarrow ((\lfloor a/20 \rfloor + a \bmod 20 + 1) \bmod 5) * \lfloor a/20 \rfloor$, where a denotes the class index, starting from 0. In this section, we will describe the details of our experiments and methods.

A.2 Implementation details of π-LR

In this section, we describe the details of our framework. We first revisited the main result of π-LR.

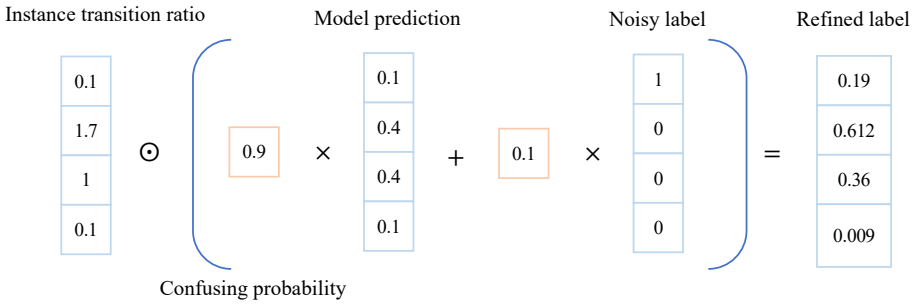

Figure 2: Illustration of the main result of π-LR.

$$\boldsymbol{q}_i = \boldsymbol{v}_i \cdot \left(\eta_i \cdot \hat{\boldsymbol{y}}_i + (1 - \eta_i) \cdot \tilde{\boldsymbol{y}}_i \right),$$

Prediction by model; Potential noisy label; Instance transition ratio; Confusing probability

We use this equation to refine potentially noisy labels. When the confusing probability η_i is high, the model's prediction will have a significant impact on the refined label. In contrast, when η_i is low, the refined label will remain the same as the original label from the training set. If the noise is independent of the true label, $\boldsymbol{v}_i$ is a unit vector. Otherwise, it reflects the change in confidence when the true class of the instance is known. By incorporating $\boldsymbol{v}_i$, the refined label can effectively model instance-dependent noise.

Algorithm 1 Overall Algorithm of π-LR

Input: training set $(S, \tilde{Y})$; estimated probabilities $\{\psi_1, \cdots, \psi_N\}$; training steps *num_steps*; hyper-parameter λ; step list of estimation $\mathcal{T}$;
Output: the optimal parameters Θ
Initialize $\eta_i = 0, \forall i$;
Initialize $\Theta = \{\boldsymbol{v}_i | i \in [N]\} \cup \{\mathbf{w}\}$;
for $t = 1$ **to** *num_steps* **do**
  $\boldsymbol{q} = \boldsymbol{v} \cdot (\eta \cdot \hat{\boldsymbol{y}} + (1 - \eta) \cdot \tilde{\boldsymbol{y}})$
  Compute $\mathcal{L}_c$
  Compute $\mathcal{L}_r$
  Compute $\mathcal{L}_v$
  Update Θ
  if $t \in \mathcal{T}$ **then**
    estimate $\{\eta_i | i \in [N]\}$
  end if
end for

Calculation of ψ_i. This term refers to the posterior probability distribution $\Pr(\tilde{\boldsymbol{y}}_i | \boldsymbol{x}_i)$, which can be estimated using a pre-trained neural network. Specifically, this probability distribution is calculated based on the model's prediction for the corresponding position in the output vector, which is determined by the label of the corresponding training example after passing through the softmax layer. In practice, we have found that this term can be approximated as a constant near 1 (e.g., 0.95) without a significant loss of accuracy, which can help to reduce the computational cost during training.

Estimation of η_i. The confusing probability η_i remains the crucial part of our framework. The estimation process in detail can be seen in the following Python-style pseudo-code as shown in Figure. 3.

A.2.1 Experimental details

Realistic label noise. For the CIFAR-N dataset, the backbone of the model is ResNet-34 with Adam optimizer using 0.9 momentum and 5e-4 weight decay rate. The alternating optimization process is executed for 100 epochs. The initial learning rate for the model is 0.1 decayed with a factor of 10 at the 60th and 80th epochs. The EMA ratio used in $\mathcal{L}_r$ is 0.3 and λ is 3. The learning rate for $\boldsymbol{v}_i$ is 1. And there is no weight decay for $\boldsymbol{v}_i$. The batch size of the training process is 128. The list of estimation epochs is set as [60, 80].

For the Clothing1M dataset, we follow the previous work, the backbone of the model is a pretrained ResNet-50 with SGD optimizer using 0.9 momentum and 1e-3 weight decay rate. The alternating optimization process is executed for 20 epochs. The initial learning rate for the model is 0.005 decayed with a factor of 10 at every 5 epochs. The EMA ratio used in $\mathcal{L}_r$ is 0.3 and λ is 1. The learning rate for $\boldsymbol{v}_i$ is 1. And there is no weight decay for $\boldsymbol{v}_i$. The batch size of the training process is 32. The list of estimation epochs is set as [7, 14].

Synthetic label noise. For the CIFAR-10/CIFAR-100 dataset, the backbone of the model is ResNet-34 with SGD optimizer using 0.9 momentum and 5e-4 weight decay rate. The alternating optimization

```
"""
Estimation Process of confusing probability
"""

# Get features provided by the model
with torch.no_grad():
    for (x, y, ids) in dataloader:
        ys[ids], features[ids] = y, model.feature(x)

# Get class centers
for c in range(num_classes):
    ids = (ys == c)
    centers[c] = torch.mean(features[ids], dim=0, keepdim=True)

# Fitting a Gaussian Mixture Model for each class
for c in range(num_classes):
    ids = (ys == c)
    gm = GaussianMixture(n_components=2).fit(features[ids])

    # get the index of cluster2 by the distance
    i = torch.sum((gm.means_ - centers[c])**2, axis=1).argmax()

    # predict confusing probabilities
    probs = gm.predict_proba(features[ids])[:,i]
    etas[ids] = probs
```

Figure 3: Pseudo-code for describing the process of estimating the confusing probabilities $\{\eta_i | i \in [N]\}$.

process is executed for 160 epochs. The initial learning rate for the model is 0.1 decayed with a factor of 10 at the 60th, 80th, and 100th epochs.The EMA ratio used in $\mathcal{L}_r$ is 0.3 and λ is 6. The learning rate for $\boldsymbol{v}_i$ is 1. And there is no weight decay for $\boldsymbol{v}_i$. The batch size of the training process is 128.

A.3 Implementation details of π-LR+

In this section, we describe the details of π-LR+. Our approach is based on previous state-of-the-art works, specifically, [30]. We use the Semi-Supervised Learning (SSL) method on both labeled and unlabeled data.

Consistency regularization. To improve performance, we incorporate consistency regularization, as previously done in [12]. This method assumes that a classifier should produce similar predictions for a data point and its local neighbors. We use two data augmentation techniques, SimAugment [52] and RandAugment [53], to obtain a weakly-augmented image $\boldsymbol{x}^w$ and a strongly-augmented counterpart $\boldsymbol{x}^s$. We then define the CR loss as $l_{\text{cr}} = l_{\text{CE}}(\boldsymbol{x}_i^s, \boldsymbol{q}_i)$, which is the cross-entropy loss on the strongly-augmented example and the refined labels.

Mixup. To further improve performance, we incorporate mixup training. We take a pair of weakly-augmented examples, $\boldsymbol{x}_i^w$ and $\boldsymbol{x}_j^w$, from the set $\mathcal{D}_c$, and create a virtual training example by linearly interpolating both:

$$\begin{aligned} \boldsymbol{x}_i^m &= \sigma \boldsymbol{x}_i^w + (1-\sigma)\boldsymbol{x}_j^w \\ \boldsymbol{q}_i^m &= \sigma \boldsymbol{q} i^w + (1-\sigma)\boldsymbol{q}_j^w \end{aligned} \tag{17}$$

where σ is sampled from a Beta distribution with parameter ς, which we set to 4 without further tuning. We define the mixup loss, $l$mix, as the cross-entropy loss on $\boldsymbol{x} i^m$ and $\boldsymbol{q}_i^m$. In addition to interpolation-based data augmentation, we also use a masking-based mixed sample data augmentation approach called FMix [54] to exploit local consistency.

Data set extension. We perform label guessing on the dirty set $\mathcal{D}_d$. Since there are no free labels to be matched on falsely-labeled data, we follow previous works and train two peer networks to produce

potential labels for each other. We denote the networks by f^1 and f^2, and select a new subset $\mathcal{D}'_c$ using the following condition:

$$(\max_j f_j^1(\boldsymbol{x}) \geq \tau) \wedge (\max_j f_j^2(\boldsymbol{x}) \geq \tau) \wedge (\arg\max_j f_j^1(\boldsymbol{x}) == \arg\max_j f_j^2(\boldsymbol{x})) \tag{18}$$

This means that the two networks should have high confidence in the same label. However, since the two networks are trained in similar environments, they are likely to agree on false labels, so the guessed labels should be treated with caution. In practice, we enable label guessing in the last K epochs when the networks are accurate enough. Additionally, we set a maximal ratio of total selected samples by ρ' (including the clean set), and those examples with smaller mean confidence values $\frac{\max_j f_j^1(\boldsymbol{x})+\max_j f_j^2(\boldsymbol{x})}{2}$ are discarded. After that, we take the shared prediction $\arg\max_j f_j^1(\boldsymbol{x})$ as the guessed labels for $\mathcal{D}'_c$. This procedure is similar to FixMatch [55], which also trains the classifier on a combination of labeled data and unlabeled data with high confidence.

Experimental details. We utilize two 18-layer ResNet as the backbone and train them for 420 epochs using a standard SGD optimizer with a momentum of 0.9 and weight decay of $5e_{-4}$. The batch size is fixed at 256, and the initial learning rate is 0.05, which is then decreased according to a cosine learning rate schedule. We set the ratio parameter ρ to 0.5, and fix the threshold parameter τ to 0.99 for CIFAR-10N and τ to 0.95 for CIFAR-100N. The parameter ς is set to 4 for mixup training. We also include a linear warm-up phase in the first 10 epochs and gradually increase β from 0 to 1 over the first 50 epochs to prevent the overfitting of false labels. The data set extension happens during the last 150 epochs. The estimation epoch list is set as [100, 200, 300, 400].

B More Experiments and Details

In this section, we provide more experiments and details of this study.

B.1 Estimation of Gaussian Mixture Models

Here we validate the estimation results via classical clustering metrics. When evaluating the performance of a clustering algorithm, three commonly used metrics are Adjusted Rand Index (ARI), Fowlkes-Mallows Index (FMI), and Normalized Mutual Information (NMI).

Adjusted Rand Index. ARI measures the similarity between the true clustering and the clustering produced by the algorithm, taking into account the possibility of chance agreement. It is defined as:

$$\text{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}$$

where n_{ij} is the number of pairs of data points that are in the same cluster in both the true and predicted clusterings, a_i is the number of data points in cluster i in the true clustering, b_j is the number of data points in cluster j in the predicted clustering, and n is the total number of data points.

Fowlkes-Mallows Index. FMI is a geometric mean of precision and recall, calculated based on the number of true positives, false positives, and false negatives. It is defined as:

$$\text{FMI} = \frac{\text{TP}}{\sqrt{(\text{TP}+\text{FP})\cdot(\text{TP}+\text{FN})}}$$

where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives.

Normalized Mutual Information. NMI measures the mutual information between the true clustering and the clustering produced by the algorithm, normalized by the entropy of the two clusterings. It is defined as:

$$\mathrm{NMI} = \frac{2I(Y,C)}{H(Y)+H(C)}$$

where Y is the true clustering, C is the predicted clustering, $I(Y,C)$ is the mutual information between Y and C, and $H(Y)$ and $H(C)$ are the entropies of Y and C, respectively.

It should be noted that the aforementioned metrics are bounded within the range of [-1, 1] or [0, 1], with higher values indicating better performance. In particular, a value of 1 for a given metric implies that the resulting cluster solutions are identical.

Table 5: Metrics of clustering results provided by GMM, the indicator is the clean indicator (i.e., one sample's label is clean or not.) Feature space is provided by a trained ResNet34 except for the last FC layer.

Dataset	FMI	NMI	ARI
CIFAR-100N	0.864	0.568	0.675
CIFAR-10N(Worst)	0.840	0.613	0.556

Evaluation. We first trained a ResNet34 model using the standard implementation of the CrossEntropy loss function on the trainset. We then applied the estimation process to the feature space of this model. Finally, we validated the estimation results corresponding to the clean indicator of the trainset (i.e., whether the label of the sample is clean or not) using the clustering metrics mentioned above.

As demonstrated in Table 5, the estimation results exhibit a significant degree of similarity when compared to the clean indicator. However, we would like to point out that,it is important to note that the ground truth for the confused samples is not available. Therefore, the clustering metrics should not be expected to be excessively high.

C Efficiency of π-LR

The size of the training dataset significantly impacts the time and space efficiency of π-LR due to the need to maintain the confusing probability η_i and instance transition ratio $\boldsymbol{v}_i$ for each sample $\boldsymbol{x}_i$. In our method, estimating the confusing probability involves fitting a GMM for each class, which incurs extra time and space complexity costs (c.f. Figure 4). However, as shown in this picture, our method is highly efficient and fast compared to other methods that require extra models and unsupervised consistency regularization.

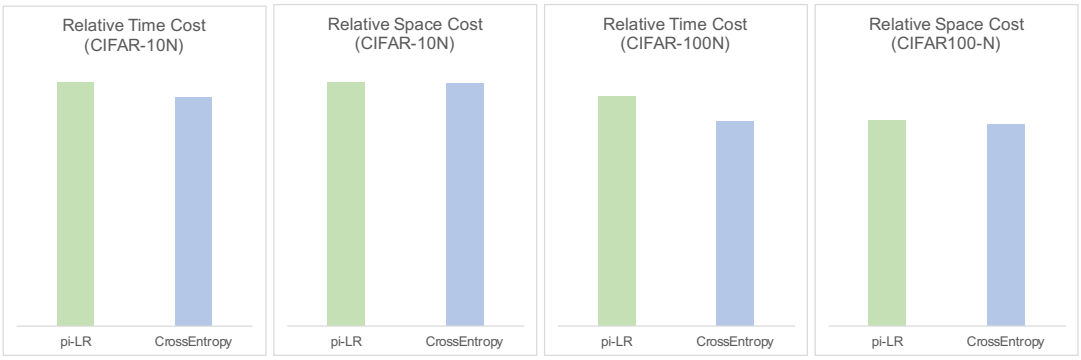

Figure 4: Efficiency: The time and space overheads are almost negligible for CIFAR-10N and CIFAR-100N. Besides, the vertical measures of the above graphs all begin at 0.