arXiv:submit/5339100 [cs.AI] 10 Jan 2024

CoLafier: Collaborative Noisy Label Purifier With Local Intrinsic Dimensionality Guidance

Dongyu Zhang[*] Ruofan Hu[*] Elke Rundensteiner[*]

Abstract

Deep neural networks (DNNs) have advanced many machine learning tasks, but their performance is often harmed by noisy labels in real-world data. Addressing this, we introduce CoLafier, a novel approach that uses Local Intrinsic Dimensionality (LID) for learning with noisy labels. CoLafier consists of two subnets: LID-dis and LID-gen. LID-dis is a specialized classifier. Trained with our uniquely crafted scheme, LID-dis consumes both a sample's features and its label to predict the label - which allows it to produce an enhanced internal representation. We observe that LID scores computed from this representation that effectively distinguish between correct and incorrect labels across various noise scenarios. In contrast to LID-dis, LID-gen, functioning as a regular classifier, operates solely on the sample's features. During training, CoLafier utilizes two augmented views per instance to feed both subnets. CoLafier considers the LID scores from the two views as produced by LID-dis to assign weights in an adapted loss function for both subnets. Concurrently, LID-gen, serving as classifier, suggests pseudo-labels. LID-dis then processes these pseudo-labels along with two views to derive LID scores. Finally, these LID scores along with the differences in predictions from the two subnets guide the label update decisions. This dual-view and dual-subnet approach enhances the overall reliability of the framework. Upon completion of the training, we deploy the LID-gen subnet of CoLafier as the final classification model. CoLafier demonstrates improved prediction accuracy, surpassing existing methods, particularly under severe label noise. For more details, see the code at `https://github.com/zdy93/CoLafier`.

Keywords: Noise Label, Label Correction.

1 Introduction

Motivation. Deep neural networks (DNNs) have achieved remarkable success in a wide range of machine learning tasks [39, 38, 40]. Their training typically requires extensive, accurately labeled data. However, acquiring such labels is both costly and labor-intensive [24, 41, 7]. To circumvent these challenges, researchers and practitioners increasingly turn to non-expert labeling sources, such as crowd-sourcing [9] or automated annotation by pre-trained models [23]. Although these methods enhance efficiency and reduce costs, they frequently compromise label accuracy [9]. The resultant 'noisy labels' may inaccurately reflect the true data labels. Studies show that despite their robustness in AI applications, DNNs are susceptible to the detrimental effects of such label noise, which risks impeding their performance and also generalization ability [1, 24].

State-of-the-Art. Recent studies on learning with noisy labels (LNL) reveal that Deep Neural Networks (DNNs) exhibit interesting memorization behavior [1, 31]. Namely, DNNs tend to first learn simple and general patterns, and only gradually begin to learn more complex patterns, such as data with noisy labels. Many methods thus leverage signals from the early training stage [13], such as loss or confidence scores, to identify potentially incorrect labels. For label correction, the identified faulty labels are either dropped, assigned with a reduced importance score, or replaced with generated pseudo labels [5, 12, 42].

However, these methods can suffer from accumulated errors caused by incorrect selection or miscorrection - with the later further negatively affecting the representation learning and leading to potential overfitting to noisy patterns [24, 27]. Worse yet, most methods require prior knowledge about the noise label ratio or the specific pattern of the noisy labels [5, 13]. In real-world scenarios, this information is typically elusive, making it difficult to implement these methods.

Local Intrinsic Dimensionality (LID), a measure of the intrinsic dimensionality of data subspaces [8], can be leveraged for training DNNs on noisy labels. Initially, LID decreases as the DNN models the low-dimensional structures inherent in the data. Subsequently, LID increases, indicating the model's shift towards overfitting the noisy label. Another study [19] applied LID to identify adversarial examples in DNNs, which typically increase the local subspace's dimensionality. These findings suggest LID's sensitivity to noise either from input features or labels. Nonetheless, previous research has utilized LID as a general indicator for the training

[*]Worcester Polytechnic Institute, {dzhang5, rhu, rundenst}@wpi.edu

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

stages or for detecting feature noise. While a promising direction for research, applying LID for detecting mislabeled samples has not been explored before.

Problem Definition. In this study, we propose a method for solving classification with noisy labeled training data. Given a set of training set with each item labeled with one noisy classification label, our goal is to train a robust classification model that solves the classification task accurately without any knowledge about the quality or correctness of the given labels.

Challenges. Classification with noisy labeled training data is challenging for the following reasons:

• *Lack of knowledge about noise ratio and noise pattern.* Without knowledge about the noise ratio and noise pattern of the given dataset, it is challenging to develop a universal method that can collect sufficient clean labels to train a strong model.

• *Compounding errors in the training procedure.* Incorrect selection or correction errors made early in the learning process can compound, leading to even larger errors as the model continues to be trained. This can result in a model that is far off from the desired outcome.

Proposed Method. In response to these challenges, we conduct an empirical study to evaluate the effectiveness of the Local Intrinsic Dimensionality (LID) score as a potential indicator for mislabeled samples. We design a specialized classifier, namely, LID-based noisy label discriminator (LID-dis). LID-dis processes both a sample's features and label to predict the label. Notably, its intermediate layer yields an enhanced representation encompassing both feature and label information. Our uniquely crafted training scheme for LID-dis reveals that the LID score of this representation can effectively differentiate between correctly and incorrectly labeled samples. This differentiation is consistent across various noise conditions.

To complement LID-dis, we introduce the LID-guided label generator (LID-gen), a regular classification model that operates solely on the data's features - not requiring access to the label. LID-dis and LID-gen together as two subnets form our proposed framework, CoLafier: Collaborative Noisy Label purifier with LID guidance. During training, we generate two augmented views of each instance's features, which are then processed by both LID-dis and LID-gen. CoLafier consider the consistency and discrepancy of the two views' LID scores as produced by LID-dis to determine weights for each instance in our adapted loss function. This reduces the risk of incorrect weight assignment. Both LID-dis and LID-gen undergo training using their respective weighted loss. Concurrently, LID-gen suggests pseudo-labels from these two augmented views for each training instance. LID-dis processes these pseudo-labels along with two views, deriving LID scores for them. These LID scores and the difference between prediction from LID-dis and LID-gen guide the decision on the label update. Information from the two views and two subnets together helps mitigate the risk of label miscorrection. After training is complete, LID-gen is utilized as the classification model to be deployed.

Contributions. Our contributions are as follows:

• We craft a pioneering approach to harness the LID score in the context of noisy label learning, leading to the development of LID-dis subnet. LID-dis processes not only a sample's features but also its label as input. This yields an enhanced representation adept at distinguishing between correct and incorrect labels across varied noise ratios and patterns.

• Drawing insights from the LID score, we introduce the CoLafier framework, a novel solution that integrates two LID-dis and LID-gen subnets. This framework utilizes two augmented views per instance, applying LID scores from the two views to weight the loss function for both subnets. LID scores from two views and the discrepancies in prediction from the two subnets inform the label correction decisions. This dual-view and dual-subnet approach significantly reduces the risk of errors and enhances the overall effectiveness of the framework.

• We conduct evaluation studies across varied noise conditions. Our findings demonstrate that, even in the absence of explicit knowledge about noise characteristics, CoLafier still consistently yields improved performance compared to state-of-the-art LNL methods.

2 Related Works

2.1 Learning With Noisy Labels. In recent studies, two primary techniques have emerged for training DNNs with noisy labels: sample selection and label correction. Sample selection approaches focus on identifying potentially mislabeled samples and diminishing their influence during training. Such samples might be discarded [5, 28], given reduced weights in the loss function [22, 10], or treated as unlabeled, with semi-supervised learning techniques applied [13, 12]. On the other hand, label correction strategies aim to enhance the training set by identifying and rectifying mislabeled instances. Both soft and hard correction methods have been proposed [22, 23, 30]. However, a prevalent challenge with these approaches is the amplification of errors during training. If the model makes incorrect selection or correction decisions, it can become biased and increasingly adapt to the noise. Another challenge arises when certain methods presuppose knowledge of the noise label ratio and pattern, using this information to inform their hyper-parameter settings [5, 12, 13]. However, in real-world scenarios, this information is typically unavail-

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

able, rendering these methods less practical for implementation.

2.2 Supervised Learning and Local Intrinsic Dimensionality. The Local Intrinsic Dimensionality (LID) [8] has been employed to detect adversarial examples in DNNs, as showcased by [19]. Their research highlights that adversarial perturbations, a specific type of input feature noise, tend to elevate the dimensionality of the local subspace around a test sample. As a result, features rooted in LID can be instrumental in identifying such perturbations. Within the Learning with Noisy Labels (LNL) domain, LID has been employed as a global indicator to assess a DNN's learning behavior and to develop adaptive learning strategies to address noisy labels [18]. However, it has not been utilized to identify samples with label noise.

In contrast to these applications, our study introduces a framework that leverages LID to detect and purify noisy labels at the sample level. Using LID, we can differentiate between samples with accurate and inaccurate labels, and its insights further guide the decision to replace noisy labels with more reliable ones.

3 Methodology

This section is organized as follows: we first introduce the problem definition, then we demonstrate the utilization of the LID score to differentiate between true-labeled and false-labeled instances. Finally, we present our proposed method, CoLafier: $\underline{\text{Co}}$llaborative Noisy $\underline{\text{La}}$bel puri$\underline{\text{fier}}$ with LID guidance.

3.1 Problem Definition. In this study, we address the problem of training a classification model amidst noisy labels. Let's define the feature space as $\mathcal{X}$ and $\mathcal{Y} = \{1, ..., N_c\}$ to be the label space. Our training dataset is represented as $\tilde{D} = \{(x_i, \tilde{y}_i)\}_{i=1}^{N}$, where each $\tilde{y}_i = [\tilde{y}_{i,1}, \tilde{y}_{i,2}, ..., \tilde{y}_{i,N_c}]$ is a one-hot vector indicating the *noisy label* for the instance x_i. Here, N_c denotes the total number of classes. If c is the noisy label class for x_i, then $\tilde{y}_{i,j} = 1$ when $j = c$; otherwise, $\tilde{y}_{i,j} = 0$. It is crucial to note that a noisy label, $\tilde{y}_i$, might differ from the actual ground truth label, y_i. An instance is termed a *true-labeled instance* if $\tilde{y}_i = y_i$, and a *false-labeled instance* if $\tilde{y}_i \neq y_i$. The set of all features in $\tilde{D}$ is given by $X = \{x_i | (x_i, \tilde{y}_i) \in \tilde{D}\}$. Our primary goal is to devise a classification method, denoted as $f(x; \Theta) \to \hat{y}$, which can accurately predict the ground-truth label of an instance. In this context, $\hat{y}_i = [\hat{y}_{i,1}, \hat{y}_{i,2}, ..., \hat{y}_{i,N_c}]$ is a probability distribution over the classes, with $\sum_{j=1}^{N_c} \hat{y}_{i,j} = 1$.

3.2 LID and Instance with Noisy Labels. In this section, we outline the use of a specially designed classifier: LID-based noisy label discriminator (LID-dis) f_{LD}, that employs LID as a feature to identify samples with incorrect labels. Prior research [19] has leveraged the LID scores from the final layer of a trained DNN classifier to characterize adversarial samples. However, in our context, the noise is present in the labels, not in the features. To ensure that f_{LD} can detect this noise, we input both the features and label into f_{LD}. f_{LD} consists of three components: a standard backbone model g_{LD} (which accepts x_i as input), a label embedding layer g_{LE} that processes the label $\tilde{y}_i$, and a classification head h_{LD} that takes the outputs of g_{LD} and g_{LE} to produce the final classification. The output from the backbone model $g_{\text{LD}}(x_i)$ and the label's embedding $g_{\text{LE}}(\tilde{y}_i)$ are merged as follows:

$$z(x_i, \tilde{y}_i) = \text{LayerNorm}\left(g_{\text{LD}}(x_i) + g_{\text{LE}}\left(\tilde{y}_i\right)\right) \tag{3.1}$$

The result, $z(x_i, \tilde{y}_i)$, the *enhanced representation* of $(x_i, \tilde{y}_i)$, is then passed to the classification head h_{LD} to predict $\tilde{y}_i$: $\hat{y}_i^D = h_{\text{LD}}\left(z(x_i, \tilde{y}_i)\right)$. Here, $\hat{y}_i^D$ represents the predicted value of $\tilde{y}_i$. If we train f_{LD} directly using the cross-entropy loss $\mathcal{L}_{\text{CE}}(\tilde{y}_i, \hat{y}_i^D) = -\sum_{j=1}^{N_C} \tilde{y}_{i,j} \log(\hat{y}_{i,j}^D)$, the model will consistently predict $\tilde{y}_i$. Using the noisy label as the "ground truth" label for measuring prediction accuracy would yield a 100% accuracy rate. This is because the model's predictions are solely based on the input label $\tilde{y}_i$. To compel the model to consider both the input features and label, we randomly assign a new label $\tilde{y}_i^*$ for each x_i, ensuring that $\tilde{y}_i^* \neq \tilde{y}_i$. We input the pair $(x_i, \tilde{y}_i^*)$ into f_{LD} to obtain another prediction $\hat{y}_i^{*D}$. We then employ the sum of the cross-entropy losses $\mathcal{L}_{\text{CE}}(\tilde{y}_i, \hat{y}_i^D) + \mathcal{L}_{\text{CE}}(\tilde{y}_i, \hat{y}_i^{*D})$ to train the network. This approach ensures that f_{LD} doesn't rely solely on the input label for predictions.

For LID calculation, we follow the method described in [18] (see Equation 1.5 in supplementary materials). Note that the input to f_{LD} is $(x_i, \tilde{y}_i)$. Assume that $(x_i, \tilde{y}_i) \in \tilde{D}_B, \tilde{D}_B \subset \tilde{D}$. Here, $\tilde{D}_B$ is the mini-batch drawn from $\tilde{D}$. Let $z(\tilde{D}_B) = \{z(x_i, \tilde{y}_i) | (x_i, \tilde{y}_i) \in \tilde{D}_B\}$. The equation to calculate LID score for $(x_i, \tilde{y}_i)$ can be presented below:

$$\widehat{\text{LID}}((x_i, \tilde{y}_i), \tilde{D}_B) = -\left(\frac{1}{k} \sum_{j=1}^{k} \log \frac{r_j(z(x_i, \tilde{y}_i), z(\tilde{D}_B))}{r_{\max}(z(x_i, \tilde{y}_i), z(\tilde{D}_B))}\right)^{-1}. \tag{3.2}$$

Here, the term $r_j(z(x_i, \tilde{y}_i), z(\tilde{D}_B))$ represents the distance of $z(x_i, \tilde{y}_i)$ to its j-th nearest neighbor in the set $\tilde{D}_B$, and $r_{\max}$ is the neighborhood's radius. Following the training procedure described above, in order to explore the properties of the LID score of $z(x_i, y_i)$ in f_{LD}, we conducted an empirical study on the

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

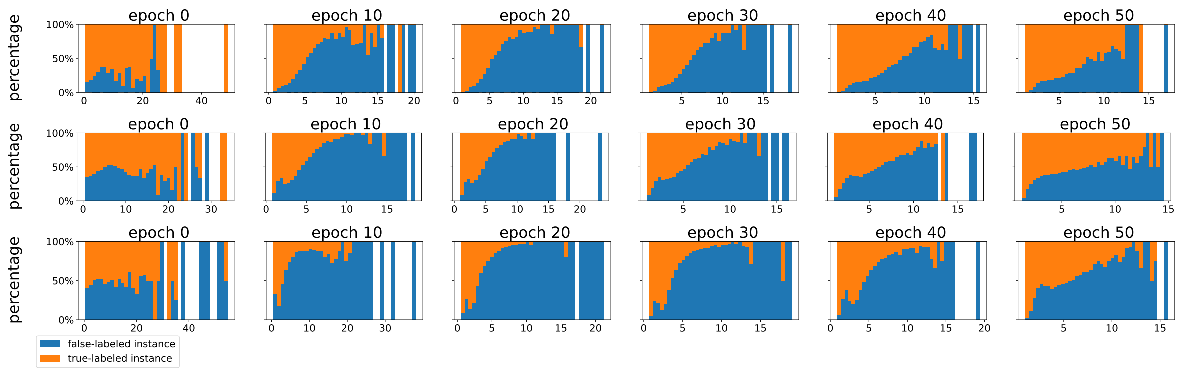

Figure 1: Distribution of LID scores for true-labeled versus false-labeled instances in three noise conditions. The heights of the orange and blue bars represent the proportions of true-labeled and false-labeled instances' LID scores within specific bins, respectively. LID scores are based on the enhanced representation of features and labels in the LID-dis. From top to bottom, the noise conditions for the three figures are: 20% instance-dependent noise, 40% instance-dependent noise, and 50% symmetric noise.

CIFAR-10 dataset with three types of noise conditions: 20% instance-dependent noise, 40% instance-dependent noise, and 50% symmetric noise. We used ResNet-34 [6] as the backbone network g_{LD}. During the training procedure, we recorded the estimation of the LID score (computed by Equation 3.2) for each instance $(x_i, \tilde{y}_i)$ at every epoch. We then split these LID scores into equal length bins and visualized the percentage distribution of false-labeled instances ($\tilde{y}_i \neq y_i$) and true-labeled instances ($\tilde{y}_i = y_i$) in each bin in Figure 1. As shown in this figure, for all three types of noise conditions, false-labeled instances tend to have higher LID scores compared to true-labeled instances. This observation underscores that, across various noise conditions, LID scores from LID-dis serve as a robust metric to differentiate between true-labeled and false-labeled instances.

3.3 Proposed Method: CoLafier. Informed by these observations, we introduce a collaborative framework, CoLafier: Collaborative Noisy Label purifier with LID guidance, tailored for learning with noisy labels. The comprehensive structure of CoLafier is depicted in Figure 2, and the pseudo-code of CoLafier is presented in the supplementary materials. This framework is bifurcated into two primary subnets: LID-dis f_{LD} and LID-guided label generator (LID-gen) f_{GE}. Specifically, f_{GE} operates as a conventional classification model, predicting $\hat{y}_i$ based on x_i. The training regimen of CoLafier unfolds in four distinct phases:

1. **Pre-processing:** For an instance $(x_i, \tilde{y}_i)$ drawn from batch $\tilde{D}_B$, we employ double augmentation to generate two distinct views: v_i^1 and v_i^2. Subsequently, a new label $\tilde{y}_i^*$ is assigned, ensuring it differs from $\tilde{y}_i$.

2. **Prediction and LID Calculation:** Post inputting the features and (features, label) pairs into LID-dis and LID-gen, predictions are derived from both subnets. Let's denote the predictions from f_{GE} as $\hat{y}_i^{1,G}$ and $\hat{y}_i^{2,G}$. Concurrently, CoLafier computes the LID scores for both $(v_i^1, \tilde{y}_i)$ and $(v_i^2, \tilde{y}_i)$.

3. **Loss Weight Assignment:** Utilizing the two LID scores from last step, CoLafier allocates weights to each instance. Every instance is endowed with three distinct weights: clean, noisy, hard weights, with each weight catering to a specific loss function.

4. **Label Update:** LID-dis processes $(v_i^1, \hat{y}_i^{1,G})$ and $(v_i^2, \hat{y}_i^{2,G})$, deriving LID scores for them. These scores and the difference between prediction from f_{LD} and f_{GE} subsequently guide the decision on whether to substitute $\tilde{y}_i$ with a combination of $\hat{y}_i^{1,G}$ and $\hat{y}_i^{2,G}$ for future epochs.

3.3.1 Pre-processing. Consider a mini-batch $\tilde{D}_B = \{(x_i, \tilde{y}_i)\}_{i=1}^{N_B}$ drawn from $\tilde{D}$. This batch can be partitioned into a feature set $X_B = \{x_i | (x_i, \tilde{y}_i) \in \tilde{D}_B\}$ and a label set $\tilde{Y}_B = \{\tilde{y}_i | (x_i, \tilde{y}_i) \in \tilde{D}_B\}$. For each $x_i \in X_B$, CoLafier generates two augmented views, v_i^1 and v_i^2. This two-augmentation design ensures that weight assignment and label update decisions in subsequent steps are not solely dependent on the original input. Such a design can lower the risk of error accumulation during the training procedure. The augmented views lead to: $V_B^1 = \{v_i^1 | v_i^1 = \text{augmentation1}(x_i), \forall x_i \in X_B\}, V_B^2 = \{v_i^2 | v_i^2 = \text{augmentation2}(x_i), \forall x_i \in X_B\}$. Same as Section 3.2, for each label $\tilde{y}_i$ in $\tilde{Y}_B$, a random new label $\tilde{y}_i^*$ is assigned, ensuring that $\tilde{y}_i^* \neq \tilde{y}_i$. The resulting set is given by: $\tilde{Y}_B^* = \{\tilde{y}_i^* | \tilde{y}_i^* = \text{assignNewLabel}(\tilde{y}_i), \tilde{y}_i \in$

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

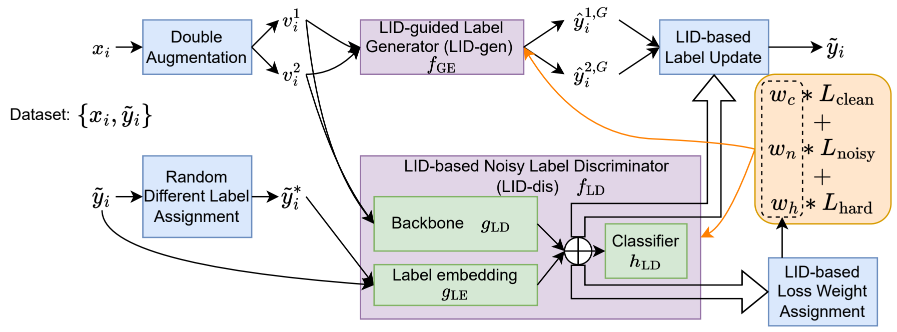
x_i
Double Augmentation
v_i^1
v_i^2
LID-guided Label Generator (LID-gen) f_{GE}
$\hat{y}_i^{1,G}$
$\hat{y}_i^{2,G}$
LID-based Label Update
$\tilde{y}_i$
Dataset: $\{x_i, \tilde{y}_i\}$
$w_c * L_{\text{clean}}$
$+$
$w_n * L_{\text{noisy}}$
$+$
$w_h * L_{\text{hard}}$
$\tilde{y}_i$
Random Different Label Assignment
$\tilde{y}_i^*$
LID-based Noisy Label Discriminator (LID-dis) f_{LD}
Backbone g_{LD}
Label embedding g_{LE}
Classifier h_{LD}
LID-based Loss Weight Assignment

Figure 2: The overall framework of CoLafier.

$\tilde{Y}_B\}$. Consequently, we can define four input pair sets: $\tilde{D}_B^k = \{(v_i^k, \tilde{y}_i) | v_i^k \in V_B^k, \tilde{y}_i \in \tilde{Y}_B\}$, $\tilde{D}_B^{k*} = \{(v_i^k, \tilde{y}_i^*) | v_i^k \in V_B^k, \tilde{y}_i^* \in \tilde{Y}_B^*\}$, where $k \in \{1,2\}$. The LID-gen subnet f_{GE} processes V_B^1 and V_B^2, while the LID-dis subnet f_{LD} handles $\tilde{D}_B^1, \tilde{D}_B^2, \tilde{D}_B^{1*}$, and $\tilde{D}_B^{2*}$.

3.3.2 Prediction and LID Calculation. The subnet f_{GE} takes V_B^1 and V_B^2 as inputs to predict: $\hat{Y}_B^{k,G} = \{\hat{y}_i^{k,G} | \hat{y}_i^{k,G} = f_{\mathrm{GE}}(v_i^k), v_i^k \in V_B^k\}$, where $k \in \{1,2\}$. The subnet f_{LD} processes $\tilde{D}_B^1, \tilde{D}_B^2, \tilde{D}_B^{1*}$, and $\tilde{D}_B^{2*}$ to predict: $\hat{Y}_B^{k,D} = \{\hat{y}_i^{k,D} | \hat{y}_i^{k,D} = f_{\mathrm{LD}}(v_i^k, \tilde{y}_i), (v_i^k, \tilde{y}_i) \in \tilde{D}_B^k\}$, $\hat{Y}_B^{k*,D} = \{\hat{y}_i^{k*,D} | \hat{y}_i^{k*,D} = f_{\mathrm{LD}}(v_i^k, \tilde{y}_i^*), (v_i^k, \tilde{y}_i^*) \in \tilde{D}_B^{k*}\}$, where $k \in \{1,2\}$. In f_{LD}, each input pair result in an enhanced representations, we use Equation 3.2 to calculate LID scores for instances in $\tilde{D}_B^1$ and $\tilde{D}_B^2$:

$$\widehat{\mathrm{LID}}^W(v_i^k, \tilde{y}_i) = \widehat{\mathrm{LID}}((v_i^k, \tilde{y}_i), \tilde{D}_B^k), \tag{3.3}$$

$$\widehat{\mathrm{LID}}^W(\tilde{D}_B^k) = \{\widehat{\mathrm{LID}}^W(v_i^k, \tilde{y}_i) \mid (v_i^k, \tilde{y}_i) \in \tilde{D}_B^k\}, \tag{3.4}$$

where $k \in \{1,2\}$. These LID scores are for the weight assignment use only. After we obtain prediction $\hat{Y}_B^{1,G}$ and $\hat{Y}_B^{2,G}$ from f_{GE}, we create another two input pair sets: $\hat{D}_B^k = \{(v_i^k, \hat{y}_i^{k,G}) | v_i^k \in V_B^k, \hat{y}_i^{k,G} \in \hat{Y}_B^{k,G}\}$, where $k \in \{1,2\}$. Both $\hat{D}_B^1$ and $\hat{D}_B^2$ are fed into the f_{LD} to obtain enhanced representations. Because we want to compare the LID scores from current noisy label and f_{GE}'s prediction to determine if we want to update the label, we create two union sets, then calculate LID scores within the two sets as follows:

$$U_B^k = \tilde{D}_B^k \cup \hat{D}_B^k, \tag{3.5}$$

$$\widehat{\mathrm{LID}}^U(v_i^k, \tilde{y}_i^k) = \widehat{\mathrm{LID}}((v_i^k, \tilde{y}_i^k), U_B^k), \tag{3.6}$$

$$\widehat{\mathrm{LID}}^U(v_i^k, \hat{y}_i^{k,G}) = \widehat{\mathrm{LID}}((v_i^k, \hat{y}_i^{k,G}), U_B^k), \tag{3.7}$$

$$\begin{aligned}\widehat{\mathrm{LID}}^U(U_B^k) = \{&\widehat{\mathrm{LID}}^U(v_i^k, \tilde{y}_i^k) | (v_i^k, \tilde{y}_i^k) \in \tilde{D}_B^k\} \cup \\ &\{\widehat{\mathrm{LID}}^U(v_i^k, \hat{y}_i^{k,G}) | (v_i^k, \hat{y}_i^{k,G}) \in \hat{D}_B^k\},\end{aligned} \tag{3.8}$$

where $k \in \{1,2\}$. We also collect the output from f_{LD}: $\hat{Y}_B^{k,G,D} = \{\hat{y}_i^{k,G,D} | \hat{y}_i^{k,G,D} = f_{\mathrm{LD}}(v_i^k, \hat{y}_i^{k,G}), (v_i^k, \hat{y}_i^{k,G}) \in \hat{D}_B^k\}$, where $k \in \{1,2\}$. Note that $\hat{Y}_B^{1,G,D}$ and $\hat{Y}_B^{2,G,D}$ are only used in label update step and do not participate in loss calculation.

3.3.3 Loss Weight Assignment. After estimating the LID scores, we compute weights for each instance. We introduce three types of weights: clean, hard, and noisy. Each type of weight is associated with specific designed loss function. A higher clean weight indicates that the instance is more likely to be a true-labeled instance, while a higher noisy weight suggests the opposite. A high hard weight indicates uncertainty in labeling. As observed in Section 3.2, instances with smaller LID scores tend to be correctly labeled. Thus, we assign higher clean weights to instances with lower LID scores, higher noisy weights to instances with higher LID scores, and higher hard weights to instances with significant discrepancies in LID scores from two views. To mitigate potential biases in weight assignment, we prefer using $\widehat{\mathrm{LID}}^W$ over $\widehat{\mathrm{LID}}^U$. This preference is due to the observation that the prediction from f_{GE} that are identical to the label could lower the label's $\widehat{\mathrm{LID}}^U$ score. Such a decrease does not necessarily indicate the correctness of a label and hence could skew the weight assignment. The weights are defined as:

$$q_{\text{low}}^{k,W} = \text{quantile}(\widehat{\mathrm{LID}}^W(\tilde{D}_B^k), \epsilon_{\text{low}}^W), \tag{3.9}$$

$$q_{\text{high}}^{k,W} = \text{quantile}(\widehat{\mathrm{LID}}^W(\tilde{D}_B^k), \epsilon_{\text{high}}^W), \tag{3.10}$$

$$q_i^{k,W} = \frac{q_{\text{high}}^{k,W} - \widehat{\mathrm{LID}}^W(v_i^k, \tilde{y}_i)}{q_{\text{high}}^{k,W} - q_{\text{low}}^{k,W}}, \tag{3.11}$$

$$w_{i,k} = \min(\max(q_i^{k,W}, 0), 1), \tag{3.12}$$

$$w_{i,c} = \min(w_{i,1}, w_{i,2}), \tag{3.13}$$

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

$$w_{i,h} = |w_{i,1} - w_{i,2}|, \tag{3.14}$$

$$w_{i,n} = \min\left(1 - w_{i,1}, 1 - w_{i,2}\right), \tag{3.15}$$

where $k \in \{1, 2\}$. Here, $w_{i,c}$, $w_{i,h}$, and $w_{i,n}$ represent clean, hard, and noisy weights, respectively. It's ensured that the sum of $w_{i,c}$, $w_{i,h}$, and $w_{i,n}$ equals 1, which is proved in the supplementary materials. The thresholds ϵ^W_{low} and ϵ^W_{high} are predefined, satisfying $0 \leq \epsilon^W_{\text{low}} \leq \epsilon^W_{\text{high}} \leq 1$. Initially, the value of ϵ^W_{high} is set low and is then linearly increased over τ epochs. This approach ensures that the model does not prematurely allocate a large number of high clean weights, given that the majority of labels have not been refined in the early stages. Only instances with low LID scores are predominantly correctly labeled. For instances with a high clean weight, we employ the cross-entropy loss for optimization. The clean loss is defined as:

$$\mathcal{L}_{\text{clean,GE}} = w_{i,c} \sum_{k=1}^{2} \mathcal{L}_{\text{CE}}\left(\tilde{y}_i, \hat{y}_i^{k,G}\right), \tag{3.16}$$

$$\begin{aligned} \mathcal{L}_{\text{clean,LD}} = w_{i,c} \sum_{k=1}^{2} \Big(&\mathcal{L}_{\text{CE}}\left(\tilde{y}_i, \hat{y}_i^{k,D}\right) \\ &+ \lambda^* \mathcal{L}_{\text{CE}}\left(\tilde{y}_i, \hat{y}_i^{k*,D}\right)\Big). \end{aligned} \tag{3.17}$$

Instances with a high $w_{i,h}$ indicate a significant discrepancy between $\widehat{\text{LID}}(v_i^1, \tilde{y}_i)$ and $\widehat{\text{LID}}(v_i^2, \tilde{y}_i)$. This suggests that these instances might be near the decision boundary. While we aim to utilize these instances, the cross-entropy loss is sensitive to label noise. Therefore, we adopt a more robust loss function, the generalized cross entropy (GCE) [44], defined as:

$$\mathcal{L}_{\text{GCE}}\left(\tilde{y}_i, \hat{y}_i\right) = \sum_{j=1}^{N_C} \tilde{y}_{i,j}\left(1 - (\hat{y}_{i,j})^q\right)/q, \tag{3.18}$$

where $q \in (0, 1]$. As shown in [44], this loss function approaches the cross-entropy loss as $q \to 0$ and becomes the MAE loss when $q = 1$. We set $q = 0.7$ as recommended by [44]. The hard loss is then:

$$\mathcal{L}_{\text{hard,GE}} = w_{i,h} \sum_{k=1}^{2} \mathcal{L}_{\text{GCE}}\left(\tilde{y}_i, \hat{y}_i^{k,G}\right), \tag{3.19}$$

$$\begin{aligned} \mathcal{L}_{\text{hard,LD}} = w_{i,h} \sum_{k=1}^{2} \Big(&\mathcal{L}_{\text{GCE}}\left(\tilde{y}_i, \hat{y}_i^{k,D}\right) \\ &+ \lambda^* \mathcal{L}_{\text{GCE}}\left(\tilde{y}_i, \hat{y}_i^{k*,D}\right)\Big). \end{aligned} \tag{3.20}$$

Instances with a high $w_{i,n}$ are likely to be mislabeled. To leverage these instances without being influenced by label noise, we adopt the CutMix augmentation strategy [37]. In essence, CutMix combines two training samples by cutting out a rectangle from one and placing it onto the other [1]. For a detailed explanation and methodology of CutMix, readers are referred to [37]. We apply CutMix twice for each sample within $\tilde{D}_B^1$ and $\tilde{D}_B^2$. The augmented views are defined as:

$$\breve{v}_i^k = \mathbf{M}^k v_i^k + (1 - \mathbf{M}^k) v_{r_k(i)}^k, \quad k \in \{1, 2\}. \tag{3.21}$$

$$\breve{y}_i^k = \lambda_k \tilde{y}_i + (1 - \lambda_k)\tilde{y}_{r_k(i)} \quad k \in \{1, 2\}. \tag{3.22}$$

Here, $r_1(i)$ and $r_2(i)$ are random indices for the two views, and $\mathbf{M}^k$ is a binary mask indicating the regions to combine. The factors λ_1 and λ_2 are sampled from the beta distribution $Beta(\alpha, \alpha)$, with $\alpha = 1$ as suggested by [37]. The proportion of the combination is determined by the λ term. Specifically, λ represents the ratio of the original view retained, while $1 - \lambda$ denotes the proportion of the other view that's patched in. The CutMix views $\breve{v}_i^1$ and $\breve{v}_i^2$ are then fed into f_{GE} to obtain predictions $\hat{y}_i^{\breve{1},G}$ and $\hat{y}_i^{\breve{2},G}$. Similarly, $(\breve{v}_i^1, \breve{y}_i^1)$ and $(\breve{v}_i^2, \breve{y}_i^2)$ are processed by f_{LD} to get $\hat{y}_i^{\breve{1},D}$ and $\hat{y}_i^{\breve{2},D}$. The loss for CutMix instances is defined as:

$$\begin{aligned} \mathcal{L}'_{\text{noisy,GE}} = \sum_{k=1}^{2} \Big[&\lambda_k \mathcal{L}_{\text{CE}}\left(\tilde{y}_i, \hat{y}_i^{\breve{k},G}\right) \\ &+ (1 - \lambda_k)\, \mathcal{L}_{\text{CE}}\left(\tilde{y}_{r_k(i)}, \hat{y}^{\breve{k},G}\right)\Big], \end{aligned} \tag{3.23}$$

$$\begin{aligned} \mathcal{L}'_{\text{noisy,LD}} = \sum_{k=1}^{2} \Big[&\lambda_k \mathcal{L}_{\text{CE}}\left(\tilde{y}_i, \hat{y}_i^{\breve{k},D}\right) \\ &+ (1 - \lambda_k)\, \mathcal{L}_{\text{CE}}\left(\tilde{y}_{r_k(i)}, \hat{y}^{\breve{k},D}\right)\Big]. \end{aligned} \tag{3.24}$$

To enhance the learning from instances with high noise weights in f_{LD}, we also employ a consistency loss. This loss, based on cosine similarity, ensures consistent predictions between $\tilde{y}_i$ and $\tilde{y}_i^*$ without relying on label guidance. The consistency loss for f_{LD} is:

$$\mathcal{L}_{\text{cons,LD}} = \sum_{k=1}^{2} \left(1 - \cos\left(\hat{y}_i^{k,D}, \hat{y}_i^{k*,D}\right)\right). \tag{3.25}$$

We combine the consistency loss and CutMix loss to get the noisy loss as:

$$\mathcal{L}_{\text{noisy,GE}} = w_{i,n} \mathcal{L}'_{\text{noisy,GE}}, \tag{3.26}$$

$$\mathcal{L}_{\text{noisy,LD}} = w_{i,n}(\mathcal{L}'_{\text{noisy,LD}} + \lambda_{\text{cons}} \mathcal{L}_{\text{cons,LD}}). \tag{3.27}$$

The overall training objectives for f_{GE} and f_{LD} combine the clean, hard, and noisy losses:

$$\mathcal{L}_{\text{GE}} = \mathcal{L}_{\text{clean,GE}} + \mathcal{L}_{\text{hard,GE}} + \mathcal{L}_{\text{noisy,GE}}, \tag{3.28}$$

$$\mathcal{L}_{\text{LD}} = \mathcal{L}_{\text{clean,LD}} + \mathcal{L}_{\text{hard,LD}} + \mathcal{L}_{\text{noisy,LD}}. \tag{3.29}$$

Both f_{GE} and f_{LD} are optimized separately using their respective loss functions.

[1] In this work, we use images as input. While CutMix was designed for images, it hasn't been widely applied to other types of input. For non-image data, other augmentation methods like Mixup [43] can be considered.

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

3.3.4 Label Update.

For determining whether to update the label based on the prediction from f_{GE}, we consider both the LID scores and the prediction difference between f_{GE} and f_{LD}. As discussed in Section 3.2, instances with smaller LID scores are more likely to be correctly labeled. If the LID scores associated with f_{GE}'s prediction are smaller than the current label's scores, then the prediction is more likely to be accurate. The prediction difference is defined as:

$$\Delta\tilde{y}_i^k = \textstyle\sum_{j=1}^{N_c} |\hat{y}_{i,j}^{k,G} - \hat{y}_{i,j}^{k,D}|, \quad k \in \{1,2\}. \tag{3.30}$$

$$\Delta\hat{y}_i^k = \textstyle\sum_{j=1}^{N_c} |\hat{y}_{i,j}^{k,G} - \hat{y}_{i,j}^{k,G,D}|, \quad k \in \{1,2\}. \tag{3.31}$$

The principle of agreement maximization suggests that different models are less likely to agree on incorrect labels [28]. The Δ value measures the level of disagreement between f_{GE} and f_{LD}. A larger Δ value indicates that the corresponding prediction or label is less likely to be correct. Generally, if a prediction has a smaller LID score and a smaller Δ compared to the current label, it's a candidate for label replacement. Using the LID scores computed in Section 3.3.2, CoLafier make decision on label updating as follows:

$$q_{\text{low}}^{k,U} = \text{quantile}(\widehat{\text{LID}}^U(U_B^k), \epsilon_{\text{low}}^U), \tag{3.32}$$

$$q_{\text{high}}^{k,U} = \text{quantile}(\widehat{\text{LID}}^U(U_B^k), \epsilon_{\text{high}}^U), \tag{3.33}$$

$$\tilde{q}_i^{k,U} = (q_{\text{high}}^{k,U} - \widehat{\text{LID}}^U(v_i^k, \tilde{y}_i))/(q_{\text{high}}^{k,U} - q_{\text{low}}^{k,U}), \tag{3.34}$$

$$\hat{q}_i^{k,U} = (q_{\text{high}}^{k,U} - \widehat{\text{LID}}^U(v_i^k, \hat{y}_i^{k,G}))/(q_{\text{high}}^{k,U} - q_{\text{low}}^{k,U}), \tag{3.35}$$

$$\tilde{t}_i^k = \min(1, \max(0, \tilde{q}_i^{k,U} * (2 - \Delta\tilde{y}_i^k)/2)), \tag{3.36}$$

$$\hat{t}_i^k = \min(1, \max(0, \hat{q}_i^{k,U} * (2 - \Delta\hat{y}_i^k)/2)), \tag{3.37}$$

where $k \in \{1,2\}$, ϵ_{low}^U and ϵ_{high}^U are thresholds. Mirroring the approach of ϵ_{high}^W, ϵ_{high}^U starts low and linearly rises over τ epochs, enabling the model to judiciously assess the reliability of labels and predictions. The Δ values are normalized to the [0,1] range using $(2-\Delta)/2$, which is elaborated in the supplementary materials. In these equations, predictions or labels with smaller LID values and smaller cross-subnet differences have larger t values, and vice versa. The process of converting both $\hat{y}_i^{1,G}$ and $\hat{y}_i^{2,G}$ to one-hot label vectors is as follows:

$$\acute{y}_i = [\acute{y}_{i,0}, \acute{y}_{i,1}, \ldots \acute{y}_{i,N_c}] \tag{3.38}$$

$$\acute{y}_{i,l} = \begin{cases} 1, & \text{if } l = \underset{j}{\text{argmax}}(\hat{y}_{i,j}) \\ 0, & \text{otherwise} \end{cases}$$

We determine whether to update the label as follows:

$$\begin{aligned} \text{cond} = {} & \hat{t}_i^1 > \tilde{t}_i^1 \;\&\; \hat{t}_i^2 > \tilde{t}_i^2 \\ & \&\; \hat{t}_i^1 > \epsilon_k \;\&\; \hat{t}_i^2 > \epsilon_k \;\&\; \acute{y}_i^{1,G} = \acute{y}_i^{2,G} \end{aligned} \tag{3.39}$$

$$\tilde{y}_i = \begin{cases} \acute{y}_i^{1,G}, & \text{if cond} \\ \tilde{y}_i, & \text{otherwise} \end{cases} \tag{3.40}$$

In this decision-making process, a label is only updated when the prediction's t value from both views surpasses a predefined threshold ϵ_k and is higher than higher than the t value of the corresponding label. Both predictions $\acute{y}_i^{1,G}$ and $\acute{y}_i^{2,G}$ must also be equal, minimizing the chance of assigning an incorrect label to instance x_i. Note that the new $\tilde{y}_i$ is used for the next epoch, ensuring that in current epoch, the loss calculation in Section 3.3.3 is not affected by the label update.

4 Experiments

In this section, we assess the performance of CoLafier across various noise settings. We also present ablation studies to validate the contribution of each component.

Experiment Setup. CoLafier is evaluated on CIFAR-10 [11] with three type of noise: symmetric (sym.), asymmetric (asym.) and instance-dependent (inst.) noise. Sym. noise involves uniformly flipping labels at random, while asym. noise flips labels between neighboring classes at a fixed probability, following methods in [5, 12]. Inst. noise is generated per instance using a truncated Gaussian distribution as per [13, 32]. Noise ratios are set at {20%, 50%, 80%} for sym. noise, 40% for asym. noise, and {20%, 40%, 60%} for inst. noise, aligning with settings in [45, 13]. Additionally, experiments are conducted on CIFAR-10N [29], a real-world noisy dataset with re-annotated CIFAR-10 images. CIFAR-10N provides three submitted labels (i.e., Random 1, 2, 3) per image, aggregated to create an Aggregate and a Worst label. The Aggregate, Random 1, and Worst label are used in experiment. ResNet-18 [6] serves as the backbone for CIFAR-10 with sym. and asym. noise, while ResNet-34 is used for CIFAR-10 with inst. noise and CIFAR-10N. Additional details are provided in the supplementary materials.

4.1 Experiment Results.

Table 1a shows that CoLafier consistently ranks among the top three in prediction accuracy on the CIFAR-10 dataset under both sym. and asym. noise conditions, achieving the highest average accuracy across four scenarios. Its robustness to various noise ratios and types stands out. In Table 1c, CoLafier achieves the highest average accuracy under inst. noise, notably excelling at an 80% noise ratio. Table 2 further demonstrates CoLafier's superior performance and robustness under real-world noise settings, particularly under high noise conditions (Worst, 40% noise). These findings underscore the robustness and superior generalization capability of CoLafier.

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

Table 1: Comparative performance on CIFAR-10 under different noise settings. Our implementations are marked with *; others are from [13]. **Bold** scores are the highest, and underlined scores the second highest in each setting.

(a) CIFAR-10, sym. and asym. noise

Methods	Sym. 20%	Sym. 50%	Sym. 80%	Asym. 40%	Average
Cross Entropy	83.31 ± 0.09	56.41 ± 0.32	18.52 ± 0.16	77.06 ± 0.26	58.83
Mixup [43]	90.17 ± 0.12	70.94 ± 0.26	47.15 ± 0.37	82.68 ± 0.38	72.74
Decoupling [20]	85.40 ± 0.12	68.57 ± 0.34	41.08 ± 0.24	78.67 ± 0.81	68.43
Co-teaching [5]	87.95 ± 0.07	48.60 ± 0.19	17.48 ± 0.11	71.14 ± 0.32	56.29
JointOptim [26]	91.34 ± 0.40	89.28 ± 0.74	59.67 ± 0.27	90.63 ± 0.39	82.73
Co-teaching+[36]	87.20 ± 0.08	54.24 ± 0.23	22.26 ± 0.55	79.91 ± 0.46	60.90
GCE [44]	90.05 ± 0.10	79.40 ± 0.20	20.67 ± 0.11	74.73 ± 0.39	66.21
PENCIL [35]	88.02 ± 0.90	70.44 ± 1.09	23.20 ± 0.81	76.91 ± 0.26	64.64
JoCoR [28]	89.46 ± 0.04	54.33 ± 0.12	18.31 ± 0.11	70.98 ± 0.21	58.27
DivideMix* [12]	92.87 ± 0.46	94.75 ± 0.14	81.25 ± 0.26	91.88 ± 0.12	90.19
ELR [15]	90.35 ± 0.04	87.40 ± 3.86	55.69 ± 1.00	89.77 ± 0.12	80.80
ELR+ [15]	95.27 ± 0.11	94.41 ± 0.11	81.86 ± 0.23	91.38 ± 0.50	90.73
Co-learning [25]	92.14 ± 0.09	77.99 ± 0.65	43.80 ± 0.76	82.70 ± 0.40	74.16
GJS* [4]	83.57 ± 1.24	50.26 ± 2.54	15.49 ± 0.18	85.64 ± 1.37	58.74
DISC* [13]	**95.99 ± 0.15**	**95.03 ± 0.12**	81.84 ± 0.21	94.20 ± 0.07	91.69
CoLafier (ours)	95.32 ± 0.08	93.64 ± 0.11	**84.42 ± 0.20**	**94.67 ± 0.11**	**92.01**

(b) CIFAR-10, inst. noise

Methods	Inst. 20%	Inst. 40%	Inst. 60%	Average
Cross Entropy	83.93 ± 0.15	67.64 ± 0.26	43.83 ± 0.33	65.13
Forward T [21]	87.22 ± 1.60	79.37 ± 2.72	66.56 ± 4.90	77.72
DMI [34]	88.57 ± 0.60	82.82 ± 1.49	69.94 ± 1.34	80.44
Mixup [43]	87.71 ± 0.66	82.65 ± 0.38	58.59 ± 0.58	76.32
GCE [44]	89.80 ± 0.12	78.95 ± 0.15	60.76 ± 3.08	76.50
Co-teaching [5]	88.87 ± 0.24	73.00 ± 1.24	62.51 ± 1.98	74.79
Co-teaching+ [36]	89.80 ± 0.28	73.78 ± 1.39	59.22 ± 6.34	74.27
JoCoR [28]	88.78 ± 0.15	71.64 ± 3.09	63.46 ± 1.58	74.63
Reweight-R [33]	90.04 ± 0.46	84.11 ± 2.47	72.18 ± 2.47	82.11
Peer Loss [16]	89.12 ± 0.76	83.26 ± 0.42	74.53 ± 1.22	82.30
DivideMix* [12]	92.95 ± 0.29	94.99 ± 0.14	89.30 ± 1.32	92.41
CORSES^2 [2]	91.14 ± 0.46	83.67 ± 1.29	77.68 ± 2.24	84.16
CAL [45]	92.01 ± 0.12	84.96 ± 1.25	79.82 ± 2.56	85.60
DISC* [13]	**96.34 ± 0.13**	**95.27 ± 0.21**	91.15 ± 2.20	94.25
CoLafier (ours)	95.73 ± 0.10	94.66 ± 0.11	**92.45 ± 1.25**	**94.28**

Table 2: Performance comparison on CIFAR-10N. Our implementations are marked with *; others are from [14].

Methods	Aggregate	Random	Worst	Average
Cross Entropy	87.77 ± 0.38	85.02 ± 0.65	77.69 ± 1.55	83.49
Forward T [21]	88.24 ± 0.22	86.88 ± 0.50	79.79 ± 0.46	84.97
Co-teaching [5]	91.20 ± 0.13	90.33 ± 0.13	83.83 ± 0.13	88.45
ELR+ [15]	94.83 ± 0.10	94.43 ± 0.41	91.09 ± 1.60	93.45
CORES^2 [2]	95.25 ± 0.09	94.45 ± 0.14	91.66 ± 0.09	93.79
DISC* [13]	**95.96 ± 0.04**	**95.33 ± 0.12**	90.20 ± 0.24	93.83
CoLafier (ours)	95.74 ± 0.14	95.21 ± 0.27	**92.65 ± 0.10**	**94.53**

Table 3: Ablation study for two views and loss types.

Variations	CIFAR-10, 40% Inst. Noise
CoLafier w/o two views	84.31 ± 0.59
CoLafier w/o noise and hard loss	91.06 ± 0.22
CoLafier w/o noise loss	91.36 ± 0.34
CoLafier w/o hard loss	93.56 ± 0.25
CoLafier	**94.66 ± 0.11**

4.2 Ablation Study. In our ablation study, we examine four CoLafier variants to assess the impact of the dual-view design and the three loss types. Table 3 presents these variants: the first uses only the original features for weight assignment and label updates; the second to fourth exclude both noise & hard loss, noise loss only, or, hard loss only, respectively. All these variants still use LID scores to assign weight and determine label updates. Results show that the complete CoLafier model outperforms its variants. The performance gap between the single-view variant and CoLafier highlights the dual-view approach's role in reducing errors and enhancing model robustness. The lesser performance of the latter three variants compared to CoLafier confirms that combining all three loss types effectively utilizes information from both correctly and incorrectly labeled instances.

5 Conclusion

In this study, we present CoLafier, a novel framework designed for learning with noisy labels. It is composed of two key subnets: LID-based noisy label discriminator (LID-dis) and LID-guided label generator (LID-gen). Both two subnets leverage two augmented views of features for each instance. The LID-dis assimilates features and labels of training samples to create enhanced representations. CoLafier employs LID scores from enhanced representations to weight the loss function for both subnets. LID-gen suggests pseudo-labels, and LID-dis process pseudo-labels along with two views to derive LID scores. These LID scores and the discrepancies in prediction from the two subnets inform the label correction decisions. This dual-view and dual-subnet approach significantly reduces the risk of errors and enhances the overall effectiveness of the framework. After training, LID-gen is ready to be deployed as the classifier. Extensive evaluations demonstrate CoLafier's superiority over existing state of the arts in various noise settings, notably improving prediction accuracy.

Acknowledgments

This work was supported by the AFRI award no. 2020-67021-32459 from USDA NIFA. We also thank the WPI DAISY group for their input on this research.

References

[1] D. Arpit et al., *A closer look at memorization in deep networks*, in ICML, 2017.

[2] H. Cheng, Z. Zhu, X. Li, Y. Gong, X. Sun, and Y. Liu, *Learning with instance-dependent label noise: A sample sieve approach*, in ICLR, 2021.

[3] S. Coles, *An introduction to statistical modeling of extreme values*, Springer London eBooks,Springer London eBooks, (2001).

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

[4] E. Englesson and H. Azizpour, *Generalized jensen-shannon divergence loss for learning with noisy labels*, in NeurIPS, 2021.

[5] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, et al., *Co-teaching: Robust training of deep neural networks with extremely noisy labels*, in NeurIPS, 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, *Identity mappings in deep residual networks*, in ECCV, 2016.

[7] D. Hofmann, P. VanNostrand, H. Zhang, Y. Yan, L. Cao, S. Madden, and E. Rundensteiner, *A demonstration of autood: a self-tuning anomaly detection system*, in VLDB, 2022.

[8] M. E. Houle, *Local intrinsic dimensionality i: an extreme-value-theoretic foundation for similarity applications*, in SISAP, 2017.

[9] R. Hu, D. Zhang, D. Tao, T. Hartvigsen, H. Feng, and E. Rundensteiner, *TWEET-FID: An annotated dataset for multiple foodborne illness detection tasks*, in LREC, 2022.

[10] R. Hu, D. Zhang, D. Tao, H. Zhang, H. Feng, and E. Rundensteiner, *Uce-fid: Using large unlabeled, medium crowdsourced-labeled, and small expert-labeled tweets for foodborne illness detection*, in IEEE Big Data, 2023.

[11] A. Krizhevsky, *Learning multiple layers of features from tiny images*, University of Toronto, (2009).

[12] J. Li et al., *Dividemix: Learning with noisy labels as semi-supervised learning*, in ICLR, 2019.

[13] Y. Li, H. Han, S. Shan, and X. Chen, *Disc: Learning from noisy labels via dynamic instance-specific selection and correction*, in CVPR, 2023.

[14] S. Liu et al., *Robust training under label noise by over-parameterization*, in ICML, 2022.

[15] S. Liu, J. Niles-Weed, et al., *Early-learning regularization prevents memorization of noisy labels*, in NeurIPS, 2020.

[16] Y. Liu and H. Guo, *Peer loss functions: Learning from noisy labels without knowing noise rates*, in ICML, 2020.

[17] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, arXiv:1711.05101, (2017).

[18] X. Ma et al., *Dimensionality-driven learning with noisy labels*, in ICML, 2018.

[19] X. Ma, B. Li, Y. Wang, S. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. Houle, and J. Bailey, *Characterizing adversarial subspaces using local intrinsic dimensionality*, in ICLR, 2017.

[20] E. Malach et al., *Decoupling "when to update" from "how to update"*, in NeurIPS, 2017.

[21] G. Patrini, A. Rozza, A. Krishna Menon, et al., *Making deep neural networks robust to label noise: A loss correction approach*, in CVPR, 2017.

[22] M. Ren, W. Zeng, et al., *Learning to reweight examples for robust deep learning*, in ICML, 2018.

[23] H. Song et al., *Selfie: Refurbishing unclean samples for robust deep learning*, in ICML, 2019.

[24] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, *Learning from noisy labels with deep neural networks: A survey*, IEEE TNNLS, 34 (2022), pp. 8135–8153.

[25] C. Tan, J. Xia, et al., *Co-learning: Learning from noisy labels with self-supervision*, in MM, 2021.

[26] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, *Joint optimization framework for learning with noisy labels*, in CVPR, 2018.

[27] Y. Tu, B. Zhang, Y. Li, L. Liu, J. Li, et al., *Learning from noisy labels with decoupled meta label purifier*, in CVPR, 2023.

[28] H. Wei, L. Feng, X. Chen, and B. An, *Combating noisy labels by agreement: A joint training method with co-regularization*, in CVPR, 2020.

[29] J. Wei, Z. Zhu, et al., *Learning with noisy labels revisited: A study using real-world human annotations*, in ICLR, 2021.

[30] Y. Wu, J. Shu, Q. Xie, Q. Zhao, and D. Meng, *Learning to purify noisy labels via meta soft label corrector*, in AAAI, 2021.

[31] X. Xia, T. Liu, et al., *Robust early-learning: Hindering the memorization of noisy labels*, in ICLR, 2021.

[32] X. Xia, T. Liu, B. Han, N. Wang, et al., *Part-dependent label noise: Towards instance-dependent label noise*, in NeurIPS, 2020.

[33] X. Xia, T. Liu, N. Wang, B. Han, C. Gong, G. Niu, and M. Sugiyama, *Are anchor points really indispensable in label-noise learning?*, in NeurIPS, 2019.

[34] Y. Xu, P. Cao, Y. Kong, and Y. Wang, *L_dmi: A novel information-theoretic loss function for training deep nets robust to label noise*, in NeurIPS, 2019.

[35] K. Yi et al., *Probabilistic end-to-end noise correction for learning with noisy labels*, in CVPR, 2019.

[36] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama, *How does disagreement help generalization against label corruption?*, in ICML, 2019.

[37] S. Yun, D. Han, et al., *Cutmix: Regularization strategy to train strong classifiers with localizable features*, in CVPR, 2019.

[38] D. Zhang, C. Sen, J. Thadajarassiri, T. Hartvigsen, X. Kong, and E. Rundensteiner, *Human-like explanation for text classification with limited attention supervision*, in IEEE Big Data, 2021.

[39] D. Zhang, J. Thadajarassiri, C. Sen, and E. Rundensteiner, *Time-aware transformer-based network for clinical notes series prediction*, in MLHC, 2020.

[40] D. Zhang, L. Wang, X. Dai, S. Jain, J. Wang, Y. Fan, C.-C. M. Yeh, Y. Zheng, Z. Zhuang, and W. Zhang, *Fata-trans: Field and time-aware transformer for sequential tabular data*, in CIKM, 2023.

[41] H. Zhang, L. Cao, S. Madden, and E. Rundensteiner, *Lancet: labeling complex data at scale*, in VLDB, 2021.

[42] H. Zhang, L. Cao, P. VanNostrand, S. Madden, and E. A. Rundensteiner, *Elite: Robust deep anomaly detection with meta gradient*, in SIGKDD, 2021.

[43] H. Zhang, M. Cisse, et al., *mixup: Beyond empirical risk minimization*, in ICLR, 2018.

[44] Z. Zhang and M. Sabuncu, *Generalized cross entropy*

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

loss for training deep neural networks with noisy labels, in NeurIPS, 2018.

[45] Z. Zhu, T. Liu, and Y. Liu, *A second-order approach to learning with instance-dependent label noise*, in CVPR, 2021.

A Local Intrinsic Dimensionality (LID)

Local Intrinsic Dimensionality (LID) serves as an expansion-centric metric, capturing the intrinsic dimensionality of a data's underlying subspace or submanifold [8]. Within intrinsic dimensionality theory, expansion models quantify the growth rate of in the number of data objects encountered as the distance from a reference sample expands [19]. To provide an intuitive perspective, consider a Euclidean space where the volume of an m-dimensional ball scales in proportion to r^m as its size is adjusted by a factor of r. Given this relationship between volume growth and distance, dimension m can be inferred using:

$$\frac{V_2}{V_1} = (\frac{r_2}{r_1})^m \Rightarrow m = \frac{\ln(V_2/V_1)}{\ln(r_2/r_1)} \tag{A.1}$$

By interpreting the probability distribution as a volume surrogate, traditional expansion models offer a local perspective on data's dimensional structure, as their estimates are confined to the vicinity of the sample of interest. Adapting the expansion dimension concept to the statistical realm of continuous distance distributions results in LID's formal definition [8]:

Definition 1 (Local Intrinsic Dimensionality). *For a data sample $x \in X$, let $r > 0$ represent the distance from x to its neighboring data samples. If the cumulative distribution function $F(r)$ is both positive and continuously differentiable at a distance $r > 0$, then the LID of x at distance r is expressed as:*

$$\mathrm{LID}_F(r) \triangleq \lim_{\epsilon \to 0} \frac{\ln(F((1+\epsilon)r)/F(r))}{\ln(1+\epsilon)} = \frac{rF'(r)}{F(r)} \tag{A.2}$$

whenever the limit exists. The LID at x is subsequently defined as the limit as radius $r \to 0$:

$$\mathrm{LID}_F = \lim_{r \to 0} \mathrm{LID}_F(r) \tag{A.3}$$

Estimation of LID. Consider a reference sample point $x \sim \mathcal{X}$, where $\mathcal{X}$ denotes a global data distribution. Each sample $x_* \sim \mathcal{X}$ being associated with the distance value $d(x, x_*)$ relative to x. When examining a dataset X derived from $\mathcal{X}$, the smallest k nearest neighbor distances from x can be interpreted as extreme events tied to the lower end of the induced distance distribution[18]. Delving into the statistical theory of extreme values, it becomes evident that the tails of continuous distance distributions tend to align with the Generalized Pareto Distribution (GPD), a type of power-law distribution[3]. In this work, we adopt the methodology from [18], and employ the Maximum Likelihood Estimator, represented as:

$$\widehat{\mathrm{LID}}(x) = -\left(\frac{1}{k}\sum_{i=1}^{k} \log \frac{r_i(x)}{r_{\max}(x)}\right)^{-1} \tag{A.4}$$

Here, $r_i(x)$ signifies the distance between x and its i-th nearest neighbor, while $r_{\max}(x)$ represents the maximum of these neighbor distances. It's crucial to understand that the LID defined in (A.3) is a *distributional* quantity, and the $\widehat{\mathrm{LID}}$ defined in (A.4) serves as its *estimate*.

However, in practice, computing neighborhoods with respect to the entire feature set X can be prohibitively expensive, we will estimate LID of a training example x from its k-nearest neighbor set within a batch randomly selected from X. Consider a L-layer neural network $h : \mathcal{X} \to \mathbb{R}^c$, where h_j is the transformation at the j-th layer, and given a batch $X_B \subset X$ and a reference point x, the LID score of x is estimated as[18]:

$$\widehat{\mathrm{LID}}(x, X_B) = -\left(\frac{1}{k}\sum_{i=1}^{k} \log \frac{r_i(h_j(x), h_j(X_B))}{r_{\max}(h_j(x), h_j(X_B))}\right)^{-1} \tag{A.5}$$

In this equation, $h_j(x)$ is the output from the j-th layer of the network. The term $r_i(h_j(x), h_j(X_B))$ represents the distance of $h_j(x)$ to its i-th nearest neighbor in the transformed set $h_j(X_B)$, and $r_{\max}$ is the neighborhood's radius. The value $\widehat{\mathrm{LID}}(x, X_B)$ indicates the dimensional complexity of the local subspace surrounding x after the transformation by h_j. If the batch is adequately large, ensuring the k-nearest neighbor sets remain in the vicinity of $h_j(x)$, the estimate of LID at $h_j(x)$ within the batch serves as an approximation to the value that would have been computed within the full dataset $h_j(X)$.

B The Pseudo Code of CoLafier

The pseudo-code for CoLafier is presented in Algorithm 1. Initially, CoLafier undergoes a warm-up phase for T_0 epochs. Subsequent epochs involve loss weight assignment and label update influenced by LID scores. To counteract error accumulation, CoLafier integrates two augmented views for each sample, using their respective LID scores to guide weight calculation and label update.

C The Design of of Equation 3.13-3.15

The design of $w_{i,c}$, $w_{i,h}$, $w_{i,n}$ in equations aims to ensure that the sum of $w_{i,c}$, $w_{i,h}$, and $w_{i,n}$ equals 1.

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

Algorithm 1 CoLafier algorithm.

```
Input: noisy dataset $\tilde{D} = \{(x_i, \tilde{y}_i)\}$, start epoch $T_0$, total epochs $T_{\max}$, total number of batches $B_{\max}$, LID-dis $f_{\text{LD}}(\Theta_{\text{LD}})$, LID-gen $f_{\text{GE}}(\Theta_{\text{GE}})$, $\lambda^*, \lambda_{cons}, \epsilon^W_{\text{low}}, \epsilon^W_{\text{high}}, \epsilon^U_{\text{low}}, \epsilon^U_{\text{high}}, \tau, \epsilon_k$.
Output: LID-gen $f_{\text{GE}}(\Theta_{\text{GE}})$
for $T = 1, ..., T_{\max}$ do
  for $B = 1, ..., B_{\max}$ do
    obtain a mini-batch $\tilde{D}_B = \{(x_i, \tilde{y}_i)\}_{i=1}^{N_B}$
    obtain view sets $V_B^1$ and $V_B^2$, and input pair sets $\tilde{D}_B^1, \tilde{D}_B^2, \tilde{D}_B^{1*}, \tilde{D}_B^{2*}$
    obtain prediction sets: $\hat{Y}_B^{k,G}$ from $f_{\text{GE}}$, and $\hat{Y}_B^{k,D}, \hat{Y}_B^{k*,D}$ from $f_{\text{LD}}$, where $k \in \{1,2\}$
    if $T \le T_0$ then
      obtain $\mathcal{L}_{\text{GE}} = \sum_{k=1}^2 \left(\mathcal{L}_{\text{CE}}(\tilde{y}_i, \hat{y}_i^{k,G})\right)$, $\mathcal{L}_{\text{GE}} = \sum_{k=1}^2 \left(\mathcal{L}_{\text{CE}}(\tilde{y}_i, \hat{y}_i^{k,D}) + \lambda^* \mathcal{L}_{\text{CE}}(\tilde{y}_i, \hat{y}_i^{k*,D})\right)$ {Warm-up}
    else
      obtain $\widehat{\text{LID}}^W(\tilde{D}_B^1)$, and $\widehat{\text{LID}}^W(\tilde{D}_B^2)$ {Using Equation 3.3-3.4 to get LID scores for weight assignment}
      obtain $\hat{D}_B^1, \hat{D}_B^2$ {Input pairs for predictions from $f_{\text{GE}}$}
      obtain $\hat{U}_B^k = \tilde{D}_B^k \cup \hat{D}_B^k$, and $\widehat{\text{LID}}^U(\hat{U}_B^k)$ {Using Equation 3.5-3.8 to get LID scores for label update}
      obtain $\{w_{i,c}\}, \{w_{i,h}\}$, and $\{w_{i,n}\}$ {Using Equation 3.9 - 3.15 to get weights for each loss term}
      obtain $\mathcal{L}_{\text{clean,GE}}, \mathcal{L}_{\text{hard,GE}}, \mathcal{L}_{\text{noisy,GE}}, \mathcal{L}_{\text{clean,LD}}, \mathcal{L}_{\text{hard,LD}}, \mathcal{L}_{\text{noisy,LD}}$ {Using Equation 3.16 - 3.27 to calculate weighted clean, hard, and noisy loss}
      obtain $\mathcal{L}_{\text{GE}} = \mathcal{L}_{\text{clean,GE}} + \mathcal{L}_{\text{hard,GE}} + \mathcal{L}_{\text{noisy,GE}}, \mathcal{L}_{\text{LD}} = \mathcal{L}_{\text{clean,LD}} + \mathcal{L}_{\text{hard,LD}} + \mathcal{L}_{\text{noisy,LD}}$
    end if
    $\Theta_{\text{GE}}^{B+1} = \text{AdamW}(\mathcal{L}_{\text{GE}}, \Theta_{\text{GE}}^B)$, and $\Theta_{\text{LD}}^{B+1} = \text{AdamW}(\mathcal{L}_{\text{LD}}, \Theta_{\text{LD}}^B)$
    if $T > T_0$ then
      for $i = 1, ..., N_B$ do
        obtain $\Delta\tilde{y}_i^k, \Delta\hat{y}_i^k$, where $k \in 1,2$ {Using Equation 3.30 and 3.31 to calculate prediction difference}
        obtain $\tilde{t}_i^k, \hat{t}_i^k, \acute{y}_i^{k,G}$, where $k \in 1,2$, then determine whether to update label $\tilde{y}_i$ with $\acute{y}_i^{k,G}$ or not {Using Equation 3.32-3.40 to make decision on label update}
      end for
    end if
  end for
end for
```

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

Proof. Without loss of generality, assume that $w_{i,1} > w_{i,2}$. Then:

$$w_{i,c} = w_{i,2},$$
$$w_{i,h} = w_{i,1} - w_{i,2},$$
$$w_{i,n} = 1 - w_{i,2}.$$

Hence $w_{i,c} + w_{i,h} + w_{i,n} = 1$. □

D The Design of Equation 3.36 and 3.37

The design of $(2 - \Delta\hat{y}_i^k)/2$ and $(2 - \Delta\tilde{y}_i^k)/2$ terms in equations aims to map both $\Delta\hat{y}^{k_i}$ and $\Delta\tilde{y}^{k_i}$ into the interval $[0, 1]$. This is based on the fact that the range of Δ is $[0, 2]$.

Proof. 1. Consider vectors $y = [y_1, y_2, ..., y_n]$ and $u = [u_1, u_2, ..., u_n]$, where $y_i \in [0, 1], u_i \in [0, 1]$. $\sum_i^n y_i = 1$, and $\sum_i^n u_i = 1$. Define Δ as $\Delta = \sum_{i=1}^n |y_i - u_i|$.

2. Without loss of generality, assume that $y_i \geq u_i$ for $i \in [1, 2, ..., m]$ and $y_j < u_j$ for $j \in [m + 1, m + 2, ..., n]$. Then:

$$\Delta = \sum_{i=1}^{m} (y_i - u_i) + \sum_{j=m+1}^{n} (u_j - y_j).$$

3. Rearranging and utilizing the fact that the summation of each vector is 1, we deduce:

$$\Delta = \sum_{i=1}^{m} y_i - \sum_{i=1}^{m} u_i + \sum_{j=m+1}^{n} u_j - \sum_{j=m+1}^{n} y_j,$$

$$\sum_{i=1}^{m} y_i + \sum_{j=m+1}^{n} y_j = \sum_{i=1}^{m} u_i + \sum_{j=m+1}^{n} u_j,$$
$$\sum_{i=1}^{m} y_i - \sum_{i=1}^{m} u_i = \sum_{j=m+1}^{n} u_j - \sum_{j=m+1}^{n} y_j,$$
$$\Delta = 2(\sum_{i=1}^{m} y_i - \sum_{i=1}^{m} u_i).$$

4. Since $\sum_{i=1}^m y_i \in [0, 1], \sum_{i=1}^m u_i \in [0, 1]$ and $\sum_{i=1}^m y_i \geq \sum_{i=1}^m u_i \geq 0$, it is implied that $(\sum_{i=1}^m y_i - \sum_{i=1}^m u_i) \in [0, 1]$.

5. Hence, $\Delta \in [0, 2]$ and consequently, $\frac{2-\Delta}{2} \in [0, 1]$. □

E Experiment Setup

All experiments are executed using A100 GPUs and PyTorch 1.13.1. We use an AdamW [17] optimizer with a learning rate of 0.001 and a weight decay of 0.001. The training epochs are 200 and the batch size is 128. CoLafier first warms up for 15 epochs, during the warm-up stage, CoLafier is optimized with cross entropy loss from two views, without weight assignment or label update.

Inspired by [13], for CoLafier, we employ two separate augmentation strategies to produce two views. The first approach involves random cropping combined with horizontal flipping, and the second incorporates random cropping, horizontal flipping, and RandAugment [25]. The value of ϵ_{low}^W and ϵ_{low}^U are both 0.001. The value of ϵ_{high}^W and ϵ_{high}^U start at values of 0.05 and 0.5 respectively, linearly increase to 1.0 in 30 epochs. The value of ϵ_k is fixed at 0.1. The values of λ^* and λ_{cons} are 0.5 and 10 respectively.

In real-world applications, the noise ratio and pattern are often unknown. Earlier studies [5, 36, 12] assumed the availability of prior knowledge about the noise ratio or pattern, and they based their hyper-parameter settings on these assumptions. Recent works [13] contend that such information is typically inaccessible in practice. Even though these works claim not to rely explicitly on noise information, their hyper-parameters still change as the noise ratio and type shift. For the sake of a fair comparison, we re-evaluated certain methods (indicated by a * symbol after the method name) using their open-sourced code. However, if these methods originally assumed unknown noise ratios or types, we kept their hyper-parameters consistent. The hyper-parameter settings we adopted were based on their medium noise ratio configurations for each noise type (50% symmetric noise, 40% asymmetric noise, 40% instance dependent noise). We executed each method five times and recorded the average of the top three accuracy scores obtained during the training process. We employ a distinct backbone model for instance-dependent noise and real-world noise to ensure our results are comparable with those presented in [14, 13].

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited