Learning with Noisy Labels via Self-Reweighting from Class Centroids

Fan Ma, Yu Wu, Xin Yu, Yi Yang

***Abstract*—Although deep neural networks have been proved effective in many applications, they are data hungry and training deep models often requires laboriously labeled data. However, when labeled data contain erroneous labels, they often lead to model performance degradation. A common solution is to assign each sample with a dynamic weight during optimization, and the weight is adjusted in accordance with the loss. However, those weights are usually unreliable since they are measured by the losses of corrupted labels. Thus, this scheme might impede the discriminative ability of neural networks trained on noisy data. To address this issue, we propose a novel reweighting method, dubbed self-reweighting from class centroids (SRCC), by assigning sample weights based on the similarities between the samples and our online learned class centroids. Since we exploit statistical class centers in the image feature space to reweight data samples in learning, our method is robust to noise caused by corrupted labels. In addition, even after reweighting the noisy data, the decision boundaries might still suffer distortions. Thus, we leverage mixed inputs that are generated by linearly interpolating two random images and their labels to further regularize the boundaries. We employ the learned class centroids to evaluate the confidence of our generated mixed data via measuring feature similarities. During the network optimization, the class centroids are updated as more discriminative feature representations of original images are learned. In doing so, SRCC will generate more robust weighting coefficients for noisy and mixed data, and facilitates our feature representation learning in return. Extensive experiments on both the synthetic and real image recognition tasks demonstrate that our method SRCC outperforms the state-of-the-art on learning with noisy data.**

***Index Terms*—Self-Reweighting, Centroids, Noisy Labels**

I. Introduction

Convolutional Neural Networks (CNN) have achieved great success in many fields, such as computer vision [1] and reinforcement learning [2]. However, training deep neural networks often requires laboriously labeled data in order to achieve promising performance. However, human annotations inevitably involve erroneous labels. This would significantly degrade model performance if many noisy labels exhibit in a training set [3].

Noisy labels are commonly encountered in practical computer vision and machine learning tasks. Existing datasets collected by search engines [4], [5], [6] or annotated by crowdsourcing systems [7] usually contain a large number of noisy labels. In addition, there are also many erroneous labels even in manually annotated datasets as annotators may label data by mistake [8], [9], [10], [11]. Noisy labels in general handicap the performance of deep networks in two aspects: First, the increasing number of incorrectly annotated samples may lead to sampling effective samples insufficiently for training networks. Second, these noisy samples will harm the model optimization process by providing incorrect supervision signals. Therefore, learning with noisy data is a critical and challenging task [12], [13], [14], [15].

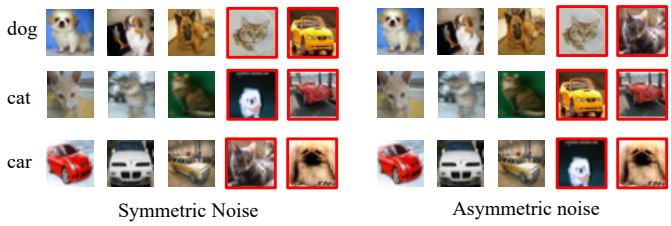


Fig. 1: Training samples of different noise types. Noise samples are marked with red boxes. Falsely annotated labels with symmetric noise could belong to any other classes in the training set. In asymmetric noise, noise samples are only from a certain class.

A common solution is to assign a dynamic weight to each sample when calculating the overall training loss. Impacts of noisy labeled data will be potentially reduced in training when they are weighted with smaller weights [7]. For instance, assigning zero weights to falsely annotated samples in Fig. 1 prevents a model learning from fallacious supervision signals. Current methods generate sample weights solely based on the losses of training samples [3], [13], [16]. Specifically, a large training loss may imply that a sample is incorrectly annotated and thus a small weight will be assigned to the sample [13]. However, a model often fits the data well to achieve small training losses. The model will assign large weights to noise samples with small losses. In this case, the assigned weights become unreliable as small weights should be produced.

We propose a novel reweighting method, namely self-reweighting from class centroids (SRCC), to ameliorate the weight assignment for noisy data. For each sample, we generate the weight by exploiting all training samples. To be specific, we first calculate the centroid of each class in the feature space. Then, the similarities between samples and class centroids are calculated to produce sample weights. Furthermore, the decision boundaries might still suffer distortions even after reweighting the noisy data. We thus leverage mixed inputs that are generated by linearly interpolating two

Manuscript received May 06, 2021; revised Sep 03, 2020 and Mar 23, 2021; accepted Mar 31, 2021.

The authors are with the ReLER Lab, University of Technology Sydney, Sydney, NSW 2007, Australia, and also with the Australian Artificial Intelligence Institute, University of Technology Sydney, NSW 2007, Australia (e-mail: fan.ma@student.uts.edu.au; yu.wu-3@student.uts.edu.au; xin.yu@uts.edu.au; yi.yang@uts.edu.au).

random images and their labels to regularize the boundaries.. Unlike the setting in [17], our data labels are noisy, and it is detrimental to train a model with directly interpolated labels. We leverage our learned robust class centroids to evaluate the confidence of the generated mixed data. The confidence of a mixed input is determined by the feature similarities between the mixed input and class centroids. In this fashion, assigning the sample weights of mixed inputs also takes all the data into account rather than two input labels that might be noisy. Our SRCC thus improves the reliability of sample weights and alleviates erroneous supervision signals caused by corrupted mixed inputs in training.

As the model optimization proceeds, sample features as well as the centroids of all classes will be updated. However, using all training samples to update class centroids at every training iteration requires a tremendous computational cost. In this paper, we propose a momentum based scheme to update class centroids in an online fashion, where only the features of training samples in a batch are used to update the centroids. During the optimization, we update the class centroids and the model parameters alternatingly. The effectiveness of our proposed method is analyzed to show the superiority of our algorithm. We have also conducted extensive experiments to validate the robustness of the proposed method. Experiments on both the synthetic and real image recognition tasks demonstrate that our SRCC outperforms the state-of-the-art methods.

Above all, our contributions are summarized in the following three-fold aspects:

- We propose a simple yet effective self-reweighting from class centroids method (SRCC) to address samples with erroneous labels in deep network optimization. To reduce the impact of corrupted labels, we generate a robust sample weight for each sample based on its feature similarity to the class centroids.
- Our SRCC assigns the mixed data with weights based on their confidences belonging to different classes and thus mitigates the problem of noisy mixed labels. To the best of our knowledge, our work is the first attempt to exploit mixed data with noisy labels to enhance the generalization of deep networks..
- Extensive experimental results on the CIFAR10, CIFAR100, Tiny-ImageNet and Clothing1M datasets demonstrate that our method achieves promising classification performance as well as a plausible network generalization ability on the test set.

II. Related Work

Conventional Statistical Learning. Statistical learning has provide a theoretically sound foundation to the problem of learning from noisy labels [12]. It addresses the issue from three different perspectives: surrogate losses, noise rate estimation, and probabilistic modeling. Manwani *et al*. [18] showed that the risk minimization under the 0-1 loss function has impressive noise-tolerance properties from the surrogate loss perspective,. Liu *et al*. [19] and Menon *et al*. [20] proposed class-probability estimators using order statistics on prediction scores in terms of noise rate estimation. From the probabilistic modeling perspective, Raykar *et al*. [12] introduced a two-coin model to handle noisy labels. However, these traditional methods mainly focus on exploring the distribution of noisy labels, but neglect the representation learning of data. In our proposed method, both supervision signals and feature representation learning are taken into account.

Robust Losses in Deep Learning. Robust losses have been proved useful in learning with noisy labels [21]. Lv *et al*. [22] proposed a curriculum loss to adaptively select samples in training. Zhang *et al*. [23] proposed a generalized cross-entropy loss (GCE) combining MAE (Mean Absolute Error) and CE losses to deal with noisy labels. Wang *et al*. [24] proposed a Reverse Cross-Entropy (RCE) to facilitate robust learning. Amid *et al*. [25] presented a Bi-Tempered loss, where two tunable temperatures are contained in the softmax layer and CE loss. Xu *et al*. [26] provided a novel information-theoretically robust loss function different from the distance-based loss as aforementioned. Although the robust loss encourages deep models to learn feature representations from noisy data, these approaches are effective only for certain types of noisy data. In contrast, our proposed method does not need to know noise types in advance and thus can be applied in practical scenarios.

Sample Reweighting. Sample reweighting has attracted increasing interest in recent years and achieved appealing performance on many practical problems [3], [13], [27]. The main idea is to introduce sample weights for the loss calculation, and iteratively update these weights during training [13], [28], [29], [30], [31]. Meng *et al*. [32] found that a monotonically decreasing weighting function makes the reweighting learning process equivalent to optimizing an implicit robust loss function. Inspired by meta-learning [33], more complicated sample reweighting schemes have been recently proposed [3], [13] by employing extra validation samples. Nevertheless, all these reweighting methods generate sample weights based on individual training losses, but neglect the fact that losses of incorrect labels may provide the misguided weights once a model overfits the training data. As class centroids are more robust to noisy data, we propose a weight assignment scheme by measuring the similarities between samples and class centroids, and thus achieve more reliable sample weights.

Data Augmentation Data augmentation methods have been designed to improve generalization performance in many applications [34], [35], [36]. Earlier works [37] use the random flipping and cropping to increase the variations of training images. Random occlusion techniques, such as random erasing, are introduced in [34], [38] to improve image recognition accuracy. Rather than operating on a single image, MixUp [17] interpolated paired inputs and their targets. To prevent manifold intrusion in MixUp, AdaMixUp learned a mixing policy [39], while Manifold MixUp [40] produced mixed representations in hidden layers. All these mixed methods assume that the mixed label is a linear combination of input labels. However, when noisy labels exist in the data, the mixed samples generated by the above methods would be assigned with wrong labels, thus degrading the performance.

III. PRELIMINARIES

In this section, we first review the empirical risk minimization (ERM) and the sample reweighting framework. Then, we present the objective function of MixUp.

A. Empirical Risk Minimization

Suppose training samples, $f(x_1, y_1), \ldots, (x_N, y_N)g \ 2 \ D$ are drawn i.i.d. from an unknown training distribution P, where x_i and y_i represent the i^{th} input image and the label, respectively. N indicates the number of the training samples. Let $F(\theta)$ be a prediction function with parameters θ, mapping the input $x_i \ 2 \ R^d$ into the output label $F(x_i;\theta) \ 2 \ R^K$. The objective of the risk minimization (RM) is:

$$\min \quad \mathop{\mathbb{E}}_{(x_i;y_i) \quad P} \ell(F(x_i;\theta), y_i), \tag{1}$$

where $\ell(\ )$ is the loss function. Eqn. (1) is empirically approximated by the training data D:

$$\min \quad \frac{1}{N}\sum_{i=1}^{N} \ell(F(x_i;\theta), y_i). \tag{2}$$

As indicated in Eqn. (2), the empirical risk minimization (ERM) assigns the same weight to all training data during optimization. However, the noise labels will handicap the model optimization severely if they are treated equally as the clean data. To alleviate the impact of corrupted data, a sample reweighting scheme is introduced to the ERM optimization,

$$\min \quad \sum_{i=1}^{N} v_i \ell(F(\boldsymbol{x}_i;\theta), \boldsymbol{y}_i), \qquad s.t. \quad \sum_{i=1}^{N} v_i = 1, \tag{3}$$

where v_i is a weight ranging from 0 to 1 for sample x_i. It represents the confidence that sample x_i is correctly labeled. By assigning large weights to correctly labeled samples and small weights to noise data, we can reduce the impacts of inaccurate training losses caused by corrupted labels. Note that, the sample weights in previous studies are either determined by manually defined weight functions [41], [16], [42] or learned from extra clean data [13], [3]. To be specific, the weight of each sample is first calculated based on the sample loss and then normalized to ensure the sum of all the sample weights to be 1. However, since the labels are noisy and weights are solely computed based on sample losses, the generated sample weights might be unreliable and fail to tackle noisy data.

B. MixUp

In ERM, deep models are prone to overfit training examples. When noisy labels exist in the training data, overfitting will worsen the generalization performance of deep networks. As the number of clean samples decreases, the performance and generalization ability of deep models will degrade. To improve the model discriminative capacity, MixUp [17] feed mixed inputs that are linearly interpolated from two random images to our model. By doing so, the model is able to regularize the decision boundaries and thus boost the model generalization and classification performance. The objective function for the mixed data input is written as,

$$\begin{aligned} \min \quad & \mathop{\mathbb{E}}_{(x_i;y_i) \quad P} \ \mathop{\mathbb{E}}_{(x_j;y_j) \quad P} \ \mathop{\mathbb{E}}_{\text{Beta}(\ ;\ )} \\ & \ell(F(g_{mix}(x_i, x_j, \lambda); \theta), g_{mix}(y_i, y_j, \lambda)), \end{aligned} \tag{4}$$

where $g_{mix}(a, b, \lambda) = \lambda \ \ a + (1 \quad \lambda) \ \ b$ is a mix function. Similar to [17], the coefficient λ follows the distribution Beta(α, α). The hyper-parameter α controls the interpolation weight between an image pair. When α is 0, we have the ERM principle. The objective in Eqn. (4) is empirically estimated by minimizing the following mixed loss function:

$$\begin{aligned} \min \quad L_{mix}(D;\theta) &= \frac{1}{M}\sum_{i=1}^{M} \ell(F(\boldsymbol{x}_i;\theta), \boldsymbol{y}_i), \\ \boldsymbol{y}_i &= g_{mix}(y_p, y_q, \lambda), \\ \boldsymbol{x}_i &= g_{mix}(x_p, x_q, \lambda), \end{aligned} \tag{5}$$

where (x_p, y_p) and (x_q, y_q) are vectors drawn from the N training samples randomly, and $\lambda \ 2 \ [0, 1]$. M indicates the number of mixed samples generated from the original samples.

IV. SELF-REWEIGHTING FROM CLASS CENTROIDS

A. Overview of SRCC

We design a self-reweighting strategy from class centroids for our training images. Notably, we intend to leverage more reliable information to generate a sample weight. Compared to individual samples, the class centers are statistically more stable to noise labels. Using the class centroids to generate the sample weight and confidence score, we are able to explore the relationship between the given sample and all the other training data rather than treating it as a single data point. The framework of our self-reweighting from class centroids is shown in Figure 2.

Although MixUp performs well on many tasks, the erroneous supervision caused by noise labels will limit the effectiveness. For example, if two samples (x_1, y_1) and (x_2, y_2) come from the same distribution containing false annotations, the ground-truth label for the mixed input $\boldsymbol{x}_1 = g_{mix}(x_1, x_2, \lambda)$ does not correspond to $g_{mix}(y_1, y_2, \lambda)$. When the ground-truth labels of interpolated samples are inconsistent with the mixed ones, known as the manifold intrusion, a model will be trained with incorrect supervision signals. As shown in Figure 3, a mixed data point generated by two samples from two diagonal classes has a high probability of lying outside the original diagonal classes. These mixed inputs will degrade the model performance when a model is trained with MixUp. To solve this issue, we also assign a sample weight to every mixed data during training. The self-reweighting objective is thus formulated as,

$$\begin{aligned} \min \quad & L_{sr}(D;\theta) = \sum_{i=1}^{M} v(\boldsymbol{x}_i)\ell(F(\boldsymbol{x}_i;\theta), \boldsymbol{y}_i), \\ & s.t. \quad \sum_{i=1}^{M} v(\boldsymbol{x}_i) = 1, \end{aligned} \tag{6}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ indicate the mixed data and label as described in Eqn. (5), and $v(\boldsymbol{x}_i)$ denotes the sample weight of $\boldsymbol{x}_i$. The

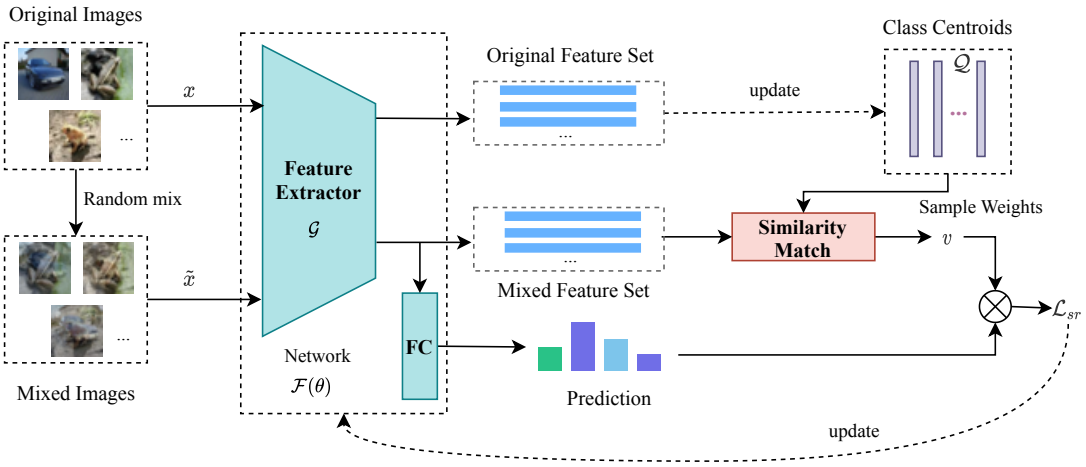


Fig. 2: Framework of our self-reweighting from class centroids (SRCC). We use solid and dash lines to denote the forward and update operations. At each training step, we first extract features and calculate the class centroids for input images (the upper part of the figure). Then we randomly mix two images by linearly interpolating two original images. The weight of the mixed data is evaluated by the similarity between its feature and all class centroids. The reweighted losses are used to update the network. The class centroids and the network are iteratively updated to learn feature representations and classify images.

higher $v(\boldsymbol{x}_i)$ means that the mixed label is more reliable and closer to the ground-truth one. Otherwise, the mixed inputs are deemed as noise samples.

Since interpolating mixed labels are often inaccurate as seen in Figure 3, our SRCC measures the quality of mixed inputs to avoid assigning high weights to mixed ones with incorrect labels. Note that mixed data are used to train the network parameters and original images are exploited to produce mixed data as well as class centroids.

B. Sample Weight Generation

To obtain the sample weight for each mixed data, we first calculate the feature centroid of each class and then compute the confidence score for each mixed data. Here, we use ResNet architecture [37] as the classifier network $F(\theta)$. The features from the penultimate layer (*i.e.*, the last fully-connected layer before the classification layer) are used as our deep features, denoted by $G(x_i)$. Therefore, the relationship between $F(x_i;\theta)$ and $G(x_i)$ is $F(x_i;\theta) = f(G(x_i))$, where f is a fully-connected layer followed by a softmax operation for classification. For each mixed example $\boldsymbol{x}_i$, its feature is denoted as $G(\boldsymbol{x}_i)$. We use Q_c to represent the feature of the center of c^{th} class. The similarity between the mixed input and the c^{th} class centroid is defined as:

$$S_c(\boldsymbol{x}_i) = \frac{e^{G(\boldsymbol{x}_i)^T Q_c}}{\sum_{k=1}^{K} e^{G(\boldsymbol{x}_i)^T Q_k}}, \tag{7}$$

where $S_c(\boldsymbol{x}_i)$ denotes the similarity between the mixed data $\boldsymbol{x}_i$ and the class centroid Q_c, and K is the total class number.

Then, we use normalized similarity scores with respect to all classes to measure mixed input confidence. As a mixed example is generated from examples from any two classes, we use $q(\boldsymbol{x}_i) = g_{mix}(S_{y_1}(\boldsymbol{x}_i), S_{y_2}(\boldsymbol{x}_i), \lambda)$ to denote the confidence of a mixed input. To ensure that the sum of all the sample weights is equal to one, we normalize sample confidence as our sample weight,

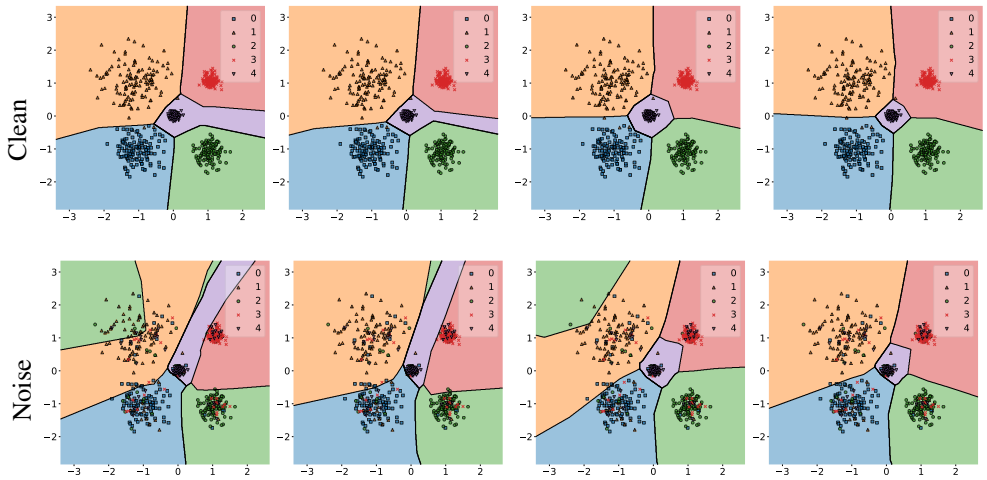
$$v(\boldsymbol{x}_i) = \frac{q(\boldsymbol{x}_i)}{\sum_{m=1}^{M}(q(\boldsymbol{x}_m))}. \tag{8}$$

To further reduce the impact of corrupted mixed inputs, we set the weight of the most unreliable sample to zero. This is achieved by using the normalization as follows:

$$v^{\theta}(\boldsymbol{x}_i) = \frac{v(\boldsymbol{x}_i) \quad v_{min}}{\sum_{m=1}^{M}(v(\boldsymbol{x}_m) \quad v_{min})}, \tag{9}$$

where $v_{min} = \min_m \; v(\boldsymbol{x}_m)$ is the minimum confidence score among all the training examples. By doing this, we alleviate the impact of the most unreliable mixed data and enlarge the range of sample weights. In other words, the reliable samples are assigned with higher weights.

Compared to the individual sample based weight generation methods [13], [3], our confidence is more robust since the similarities between the sampled data and all the class centroids provide a more comprehensive manner to measure the position of the sample in the feature space. As illustrated in the second row of Figure 3, a training loss might be small for a mixed sample interpolated from two data points in the categories 1 (orange) and 2 (green) when their data labels are mislabeled as category 3 (red). In this case, the network would classify this sample into the category 3 (red zone) and produces a small loss. Previous methods will assign a large sample weight for the mixed data point. Thus, the noisy samples would deviate the optimization process and degrade classification performance. In contrast, our method correlates the sample weight of one mixed example with all training samples via the class centroids. Although the loss for a single data might be small, the calculated distance between the mixed sample (interpolated from category 1 and 2) and the class centroid of category 3 will be larger than the distance between



(a) CE (b) Reweight (c) MixUp (d) SRCC

Fig. 3: A toy experiment on synthetic data illustrates the effectiveness of our SRCC on regularizing the decision boundaries. The clean data (the first row) are generated from five Gaussian distributions with different means and standard deviations. The noisy data (the second row) are generated by randomly changing labels of examples in the clean data. The decision boundaries are displayed by Mlxtend [43].

the sample and the class centroid of category 2 (green) or the category 4 (purple). Thus, the computed confidence of the mixed data will be small. We thus assign a small weight to the mixed data to avoid the distraction in optimization.

C. Class Centroid Update

The weight of the mixed input is generated by similarities between the mixed feature and all class centroids. If the learned centroid of one class is close to its ground-truth feature center, the similarity measurement is of high confidence to reflect the label correctness of the given samples. For the original sample x_i, we first extract the feature representation $G(x_i)$. Suppose we have a model with parameters D and training samples N. In a fully-connected network, it takes at least DN arithmetic operations to update the class centroids once. As it requires about BD (B denotes the batch size, $B << N$) arithmetic operations to update parameters one time, updating class centroids using all training samples will consume massive computational resources in each iteration.

We instead propose a momentum based scheme to update class centroids through batch samples. Specifically, at the t^{th} iteration, we first calculate the sample weight $v^t(x_i)$ for the original data x_i according to Eqn. (7). We then update the class centroids as follows:

$$Q_c^t = (1 \quad \xi)Q_c^{t\ 1} + \xi \sum_{i=1}^{B} v^t(x_i) \quad G^t(x_i) \quad I_c(y_i), \qquad (10)$$

where t indicates the iteration step and ξ is set to the learning rate to control the momentum. I_c is an indicator function. It outputs 1 when the c^{th} position in the one-hot encoding label y_i is also 1. Otherwise, the indicator function outputs 0. It only takes BD arithmetic operations to update the class centroids in one iteration, which is the same as the computational cost in a model forward process.

Algorithm 1 Self-Reweighting from Class Centroids

Input: Dataset D, Initiated parameters θ^0, Initialted centers Q^0, mixed parameter α.
for $t = 1$: num_iterations **do**
  Sample $(x_i, y_i)_{i=1}^{B}$ from D.
  Extract features $fG^t(x_i)g^B$ from original samples.
  Update class centroids using Eqn. (10).
  Sample another B examples $(x_i, y_i)_{i=1}^{B}$ from D.
  Calculate mixed samples $f(x_i, y_i)g_{i=1}^{B}$.
  Extract features $fG^t(x_i)g_{i=1}^{B}$ for mixed inputs.
  Calculate weights $fv^t(x_i)g_{i=1}^{B}$ for mixed samples.
  Update model parameters using Eqn. (6).
end for

After obtaining the updated class centroids Q^t, we sample another batch of the original data to generate mixed images. Since the network may overly trusts the mixed inputs if they are produced from the same data used for updating the centroids, we resample another batch data to avoid this phenomenon. Then, we extract the features of the mixed samples $G(x_i)$ as well as calculate the sample weights $v^t(x_i)$ to measure the training loss. Finally, the loss is backpropagated to update our model parameters.

D. Analysis of SRCC

In this section, we analyze the effectiveness of our proposed method. For illustration, we consider a binary classification case and use the features $fG(x_i)g_{i=1}^{N}$ from the last layer. We use $y_i = f0, 1g$ to denote the label of x_i. Let Q_0 and Q_1 be the centroids for negative and positive sets, respectively. All samples are split into the positive set $P = fx_i|y_i = 1g$ and the negative set $N = fx_i|y_i = 0g$ based on the noisy labels. The samples in the noise set S are falsely annotated. Let $\mathbf{w}^t$ and

$\mathbf{w}$ be the model parameters in the t^{th} step and the optimal parameter. For each sample in the noise set, we have

$$jy_i \quad \sigma(\mathbf{w}^T G(x_i))j \quad 1, 8x_i \; 2 \; S, \tag{11}$$

where $\sigma(\,)$ is the sigmoid function. Let $\hat{y}_i^t = \sigma(\mathbf{w}^{tT} G(x_i))$ be the predicted label in the t^{th} iteration. Here, the model with the optimal parameter $\mathbf{w}$ is able to output the clean labels for falsely annotated samples.

For simplicity, we use ℓ_i to denote the loss of sample x_i. The loss based sample reweighting methods fail to learn optimal parameter in the following theorem.

Theorem 1: Suppose that $v_i = 1 \quad \epsilon$ *holds when* $\ell_i < \epsilon$ *(*$\epsilon > 0$ *and* $\epsilon^2 \quad 0$*) in the loss sample reweighting algorithms. At the* t^{th} *iteration, if* $jy_i \quad \hat{y}_i^t j = \epsilon_i$ *and* $\frac{i}{1 \quad i} < \epsilon$ *for every* $\hat{y}_i^t$*, and* $\sum^P \epsilon_i G(x_i) \quad \sum^N \epsilon_j G(x_j) = \vec{0}$*, the model parameter* $\mathbf{w}^t$ *will not converge to* $\mathbf{w}$ *after iterations.*

It indicates that losses of noise samples will not be rectified if a model overfits the training samples. In contrast, our proposed method can still update $\mathbf{w}^t$ to approach $\mathbf{w}$ in the following theorem.

Theorem 2: If $9S^{\theta} \quad S$*, the condition* $Q_{y_i}^T G(x_i) < Q_1^T \; _{y_i} G(x_i)$ *satisfies for every* x_i *in* S^{θ}*. For the rest of* x_i*,* $(Q_1 \quad Q_0 \quad \mathbf{w}^t)^T G(x_i) = 0$*. The model parameter* $\mathbf{w}^t$ *will converge to* $\mathbf{w}$ *after iterations.*

The proofs of the theorems are provided in the Appendix.

E. Training Details

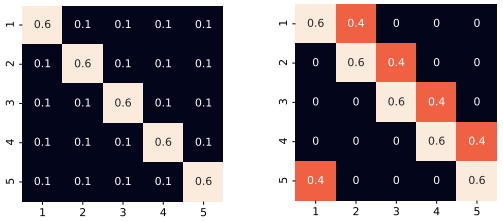
Algorithm 1 illustrates that our model parameters θ and class centroids Q are updated alternatingly in each iteration. We first sample B original examples from the dataset D and extract their features to update the class centroids. In the first few iterations, we assign all original samples with the same weight when updating class centroids since the initial network is not discriminative enough at first. Afterwards, we update class centroids by adopting our proposed reweighting mechanism and momentum based update strategy as indicated in Eqn. (10). We then use the updated class centroids to obtain sample weights of mixed inputs. The mixed inputs are generated from the original images with a mixing coefficient λ which is randomly sampled from a beta distribution parameterized by α. The weights of mixed inputs are determined and normalized by Eqn. (7) and Eqn. (8). We update the model parameters θ by minimizing the objective in Eqn. (6).

V. Experiments

In this section, we first describe the datasets and implementation details in the experiment setting. We then evaluate our model on the image recognition datasets and compare with the state-of-the-art algorithms.

A. Experimental Setup

1) Datasets: We testify the effectiveness of our proposed model on two benchmark datasets: CIFAR-10 and CIFAR-100 [44], consisting of color images of 32 32 pixels arranged in 10 and 100 classes, respectively. There are 50,000 training and 10,000 test images in both datasets. We then evaluate our method on a larger dataset Tiny-ImagNet [45] which contains a training set of 100,000 images and a validation set of 10,000 images. These images are collected from 200 different classes of objects in ImageNet [1], and images are downsampled from the original resolution 256x256 pixels to 64x64 pixels. Instead of selecting a few clean examples as metadata to guide the learning process [3], [13], we use all the training images without using any priors of clean data. We also conduct experiments on Clothing1M [46], which is a large-scale dataset with real-world noisy labels and consists of 1M training images collected from online shopping websites.



(a) Symmetric (b) Asymetric

Fig. 4: Transition matrices of different noise types at 40% noise rate (using 5 classes as an example).

2) Noise Setting: We test two types of label noise following the setting in [47]. Symmetric noisy labels (Figure 4a) are produced by replacing a certain proportion of the labels of one class with other class labels uniformly [47]. In addition, we follow the setting in [3] to generate asymmetric noisy data (Figure 4b), where labels are changed to another class in a pre-defined portion. As shown in the Fig. 4, we adopt different transitional matrices to produce noisy data of different noise types. We also set the noise rate to different levels following [24], [48] to measure the model robustness.

3) Implementation Details: We adopt various neural networks as the base classifiers for CIFAR-10 and CIFAR-100 datasets. ResNet32 [37], Preact-ResNet18 [49], MobileNet [50] and WRN28-10 [51] are selected as our backbone networks in our experiments. We train all the models by stochastic gradient descent (SGD) with the batch size of 128 and set the initial learning rate to 0.1 with momentum 0.9 following [17]. The learning rate decreases by 10 at the 50 epoch and 60 epoch, and models are trained for 70 epochs. For the Clothing1M dataset, we follow the previous work [3] and use ResNet-50 with ImageNet pre-trained weights. Note that we fix the number of mixed training samples M in every training epoch. The model obtains poor performance if M is set to a small value. When M is large, it requires a longer time to train a model for one epoch. In our experiments, we set the number of mixed samples M to the number of original training samples for the sake of implementation efficiency and model performance. The default mixing parameter α is set to 1.0. We also analyze the effect of α in Sec. VI C. We run all the experiments on the Nvidia RTX-2080Ti card. For all the compared methods, we use the optimal hyperparameters reported in their original papers.

TABLE I: Comparisons with different state-of-the-art methods on CIFAR10 and CIFAR100. Mean and standard deviation of Top-1 Accuracy are reported. The relative degradation between the noise and clean cases is also reported in the parentheses. The best and second best results are marked in red and blue respectively.

Dataset	Methods	Clean	Symmetric Noise			Asymmetric Noise	
			Noise Rate			Noise Rate	
			0.2	0.4	0.6	0.2	0.4
CIFAR10	CE	92.89 0.32	76.83(16.06) 2.30	70.77(22.12) 2.31	63.21(29.68) 4.22	79.24(13.65) 1.33	69.92(22.97) 1.97
	Forward [45]	91.85 0.15	87.83(4.12) 0.32	84.19(7.66) 0.21	78.92(12.93) 0.29	89.29(2.56) 0.50	82.32(9.53) 0.70
	Coteach [46]	92.19 0.12	90.69(1.50) 0.12	85.30(6.89) 0.29	78.21(13.98) 2.58	82.22(9.97) 0.93	79.00(13.19) 1.27
	Meta-WeightNet [4]	92.04 0.15	89.19(2.85) 0.57	86.10(5.94) 0.18	81.31(10.73) 0.37	90.33(1.71) 0.61	87.57(4.47) 0.23
	GCE [27]	90.03 0.30	88.51(1.52) 0.37	85.48(4.55) 0.16	81.29(8.74) 0.23	88.55(1.48) 0.22	83.31(6.72) 0.14
	SL [28]	89.31 0.29	88.38(0.93) 0.29	86.00(3.31) 0.23	81.19(8.12) 0.40	88.41(0.90) 0.28	82.87(6.44) 0.65
	Bi-Tempered [29]	90.11 0.23	88.51(1.60) 0.31	84.93(5.18) 0.67	77.82(12.29) 0.79	88.23(1.88) 0.23	82.43(7.68) 0.23
	SRCC	92.41 0.17	90.52(1.89) 0.13	87.43(4.98) 0.42	81.59(10.82) 0.41	91.09(1.32) 0.25	87.89(4.52) 0.52
CIFAR100	CE	70.50 0.12	50.8(19.64)6 0.27	43.01(27.49) 1.16	34.43(36.07) 0.94	52.36(18.14) 0.17	41.23(29.27) 1.26
	Forward [45]	68.52 0.36	61.27(7.24) 0.40	55.69(12.83) 0.32	45.15(23.37) 1.88	65.04(3.48) 7.71	46.77(21.75) 0.51
	Coteach [46]	65.09 0.19	62.48(2.61) 0.28	53.88(11.21) 0.29	40.94(24.15) 0.87	57.55(7.54) 0.25	49.23(15.86) 0.78
	Meta-WeightNet [4]	69.13 0.33	64.22(5.09) 0.28	58.64(10.67) 0.47	47.43(21.58) 0.76	64.22(5.09) 0.28	55.25(14.06) 0.47
	GCE [27]	67.39 0.12	63.97(3.42) 0.43	58.33(9.06) 0.35	41.73(25.66) 0.36	62.07(5.32) 0.41	53.29(14.10) 0.09
	SL [28]	62.82 1.44	60.74(2.08) 1.27	58.04(4.78) 1.79	46.77(16.05) 2.43	61.49(1.33) 0.85	51.21(11.61) 0.49
	Bi-Tempered [29]	67.90 0.27	64.95(2.95) 0.22	59.83(8.07) 0.46	50.73(17.17) 0.50	61.25(6.65) 0.33	46.26(21.64) 0.36
	SRCC	69.31 1.16	65.76(3.55) 0.36	60.62(8.69) 0.68	49.23(20.08) 1.49	65.98(3.33) 0.50	54.86(14.45) 0.64

TABLE II: Comparisons with different state-of-the-art methods in terms of test accuracy (%) on Tiny-ImageNet.

Methods	Clean	Symmetric			Asymmetric	
		Noise Rate			Noise Rate	
		0.2	0.4	0.6	0.2	0.4
CE	58.59	44.39	37.14	31.19	47.01	34.06
Forward [52]	58.52	46.51	37.16	29.72	48.16	34.51
Coteach [47]	55.23	46.93	42.17	21.53	50.71	39.06
Meta-WeightNet [3]	57.55	51.33	46.68	39.91	51.23	37.72
GCE [23]	57.13	49.16	46.02	40.73	47.93	39.43
SL [24]	56.45	53.09	49.68	41.24	53.16	36.67
Bi-Tempered [25]	58.04	53.49	46.44	35.11	49.36	39.25
SRCC	59.77	54.24	50.64	41.56	54.45	40.91

B. Comparisons with the State-of-the-art

We compare our methods with different state-of-art methods. For fair comparisons on CIFAR10 and CIFAR100, ResNet32 is adopted as the base classifier by all the methods. For the Tiny-ImagnetNet dataset, we use Preact-ResNet18 as the base classifier for all the methods. The CE denotes the model utilizes the cross-entropy loss to train the networks on noisy data. Forward [52] corrects the prediction by a label transition matrix. Coteach [47] adopts two models and an exchange loss for robust training. Meta-WeightNet [3] uses a simple network to learn a weighting function in a data-driven fashion, representing the state-of-the-art sample weighting methods. GCE [23] introduces a generalized cross-entropy loss for training deep neural networks with noisy labels. SL [24] employs an asymmetric cross-entropy loss for robust learning with noisy labels. Bi-Tempered [25] introduces a robust bi-tempered logistic loss for training models with noisy labels.

As shown in Table I, our proposed SRCC in general achieves the highest test accuracy. Bi-Tempered performs well for symmetric noisy datasets but fails to handle asymmetric noise. Although the results of Meta-WeigthNet for different noisy types are stable, it requires extra clean data to calibrate the sample weights during training. As Meta-WeigthNet requires to feed forward both training and validation data, and propagate gradients backwards three times, it takes at least several times computational resources of baseline to update the network. In our SRCC, we only need extra forward operations to update the class centroids. We also summarize the average running time of a training epoch on CIFAR10. For one epoch, training the baseline, our SRCC and Meta-WeightNet on average cost 10.72s, 17.73s, and 81.27s, respectively. Therefore, Meta-WeightNet performs much slower than our approach during training.

For all compared methods, they either adopt different loss functions (such as Bi-temp [29]) or introduce sample weights (such as Meta-WeightNet [4]). The supervision signals of these methods are thus different from the standard CE loss. In practice, noise levels of training data are unknown to the compared algorithms. Therefore, we do not assume that the clean data are known in advance. Even though all samples are clean, the compared algorithms may also recognize them as noise ones, thus leading to the performance degeneration. Following the work [29], we also report top-1 accuracy of different methods in the noise-free case. As suggested, we report the relative degradation in Table I. Our method is comparable to the baseline in the noise-free case but our model is agnostic against the noise degrees of data. This implies that our proposed model identifies the clean samples accurately.

Furthermore, we also report the classification performance of different methods on Tiny-ImageNet in Table II. Our proposed SRCC not only achieves the best classification accuracy when there is no noise in the training data, but also outperforms all the other methods when the training data contain different types of noisy examples.

To verify the effectiveness of the proposed method on real-world data, we further conduct experiments on Clothing1M. The results of other methods are reported from original papers. Table III shows the test accuracy of different methods. As expected, our proposed method achieves the highest test accuracy on this dataset.

C. Model Evaluation

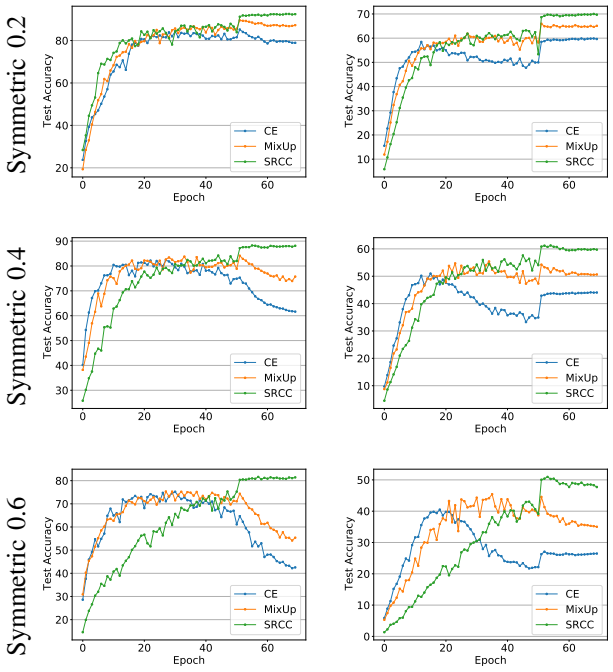
1) Regularization on Decision Boundaries: We conduct experiments on synthetic noise samples to demonstrate the

TABLE III: Comparisons with state-of-the-art methods in terms of test accuracy (%) on Clothing1M.

Method	CE	Forward [52]	SL [24]	Meta-WeightNet [3]	SRCC
Accuracy	69.21	69.84	71.02	73.72	**73.99**

TABLE IV: Classification accuracy on synthetic noisy data. Instances are sampled from the same distribution for five seeds.

Methods	seed 1	seed 2	seed 3	seed 4	seed 5
CE	89.67	91.83	89.33	89.67	91.50
Reweight	95.33	97.00	95.67	96.67	97.17
MixUp	97.33	97.33	98.33	98.00	97.33
SRCC	98.33	99.00	98.66	98.83	99.00

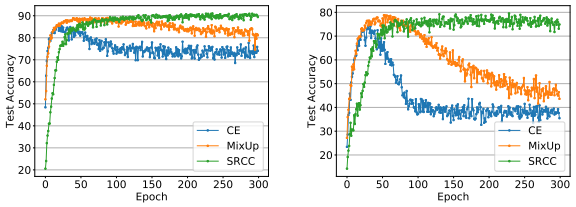


(a) CIFAR10 (b) CIFAR100

Fig. 5: Test accuracy vs. the number of epochs for SRCC and the compared methods.

effectiveness of our proposed SRCC on regularizing the decision boundaries. The training samples are generated from five Gaussian distributions centred at five 2D points as shown in Figure 3. There are in total 600 samples in the training set. We random flip labels of 40 examples to other classes to generate noisy data. Test samples are generated from the same distribution. We use a network with 2 hidden layers as the base classifier for all the methods. All the methods are trained with the batch size 64 for 200 epochs. In Figure 3, the decision boundaries of the compared methods are visualized by the Mlxtend tool [43]. As illustrated in Figure 3, our proposed SRCC better regularizes the decision boundaries of the network on both clean and noisy data. As a consequence, our SRCC outperforms the other methods on the classification accuracy, as indicated in Table IV.

2) Robustness wrt. Noise Rates and Architectures: To demonstrate the robustness of our proposed method, we investigate the performance of our method with respect to different



(a) Symmetric 0.2 (b) Symmetric 0.6

Fig. 6: Test accuracy vs. the number of epochs for SRCC and the compared methods trained with the fixed learning rate.

backbones and noise rates. We conduct experiments on the CIFAR10, and each experiment is repeated five times using different random seeds. The mean and standard deviation of the top-1 error rate are reported. In particular, we compare our SRCC with the MixUp method [17] in the following settings: (1) The noisy data are generated with symmetric noise, where the noise rates are set to 0.2, 0.4 and 0.6 respectively. (2) Different classifier architectures, *i.e.*, MobileNet [50], Preact-ResNet18 [49] and Wide ResNet [51], are adopted.

Table V indicates that different base classifiers armed with our SRCC algorithm achieve the lowest classification errors. Additionally, the error rates of Wide-ResNet28 are lower than those of Preact-ResNet18 and MobileNet. This indicates that a stronger base classifier also improves the model performance. All the classifiers trained with our SRCC achieve lower error rates than those trained with MixUp on both datasets in different noise rates. This manifests that the models trained with SRCC obtain better robustness against noisy data, demonstrating the superiority of our SRCC.

3) Generalization Ability Analysis: To analyze the test accuracy behaviours of different losses during training, we plot the test accuracy in every iteration in Figure 5. Preact-ResNet18 is adopted as the base classifier for different methods. For the models trained with the CE loss, the performance decreases dramatically after reaching the highest test accuracy. This phenomenon indicates that the noise samples provide erroneous supervision in training, leading to inferior predictions for the test samples. Compared with the MixUp, our model achieves better performance on both CIFAR10 and CIFAR100.

To further investigate the causes of performance degradation, we conduct extra experiments on CIFAR10 with different symmetric noise rates. Instead of decaying the learning rate, we train all models for 300 epochs with the same learning rate 0.1. As shown in Figure 6, the performance of compared methods (*e.g.*, CE and MixUP) still decreases even if the learning rate is not decayed. This indicates that the performance degradation is caused by the fact that the noise examples provide false supervision and the compared methods cannot suppress these erroneous signals during training. The performance degradation becomes more severe when compared models are trained on data containing more noise samples. This shows that the performance of CE and MixUp is easily affected by noise samples, especially when there are many noise samples in the training data. Compared to these methods, the performance of our SRCC is stable, demonstrating the

TABLE V: Classification errors on CIFAR10 and CIFAR100 in different noise rates. Mean and standard deviation are reported.

Model	Methods	CIFAR10 Symmetric Noise Rate 0.2		CIFAR10 Symmetric Noise Rate 0.4		CIFAR10 Symmetric Noise Rate 0.6		CIFAR100 Symmetric Noise Rate 0.2		CIFAR100 Symmetric Noise Rate 0.4		CIFAR100 Symmetric Noise Rate 0.6	
MobileNet	CE	13.53	0.41	18.03	0.28	25.69	0.67	47.95	0.90	55.10	0.22	66.66	0.93
	MixUp	12.90	0.16	16.40	0.57	23.24	0.41	38.30	0.69	47.14	0.77	60.84	0.80
	SRCC	**12.02**	0.12	**16.11**	0.54	**22.44**	0.69	**35.14**	0.39	**40.76**	0.29	**54.35**	0.49
Preact-ResNet18	CE	16.60	0.40	18.56	2.05	25.16	0.55	40.28	0.53	48.85	0.96	58.97	1.14
	MixUp	9.86	0.36	15.54	0.40	22.81	0.71	35.15	0.80	44.86	0.78	54.50	0.68
	SRCC	**8.50**	0.92	**12.02**	0.45	**19.26**	1.20	**30.48**	0.72	**38.29**	0.45	**49.02**	1.60
Wide-ResNet28	CE	14.53	0.42	17.40	0.37	25.38	0.35	35.00	0.58	45.80	0.93	56.73	0.97
	MixUp	8.83	0.46	13.69	0.48	18.98	1.61	28.80	0.55	37.45	0.81	45.65	0.17
	SRCC	**7.37**	0.36	**11.55**	0.97	**17.81**	0.79	**27.35**	0.21	**33.64**	0.39	**41.77**	0.81

TABLE VI: Classification errors on CIFAR10 for different hyper-parameter values α.

Methods	Noise Rate 0.2		Noise Rate 0.4	
MixUp ($\alpha = 0.2$)	11.13	0.38	16.43	0.72
MixUp ($\alpha = 0.5$)	11.82	4.23	15.85	0.82
MixUp ($\alpha = 1.0$)	10.22	0.62	15.25	0.41
SRCC ($\alpha = 0.2$)	9.35	0.32	13.38	0.34
SRCC ($\alpha = 0.5$)	**8.64**	0.49	11.81	0.23
SRCC ($\alpha = 1.0$)	8.70	0.78	**11.52**	0.27

TABLE VII: Effects of reweighting strategies on CIFAR10 and CIFAR100.

Dataset	Methods	Noise Rate 0.2		Noise Rate 0.4		Noise Rate 0.6	
CIFAR10	CE	83.40	0.40	81.44	2.05	74.84	0.55
	Reweight	88.41	0.56	85.79	1.19	75.41	2.58
	MixUp	90.14	0.36	84.46	0.40	77.19	0.71
	SRCC (L_c)	91.22	1.09	87.71	0.62	80.55	1.15
	SRCC ($L_{rc} + v_1$)	91.28	1.28	87.59	0.61	**80.81**	0.67
	SRCC ($L_{rc} + v_2$)	**91.50**	0.92	**87.98**	0.45	80.74	1.20
CIFAR100	CE	59.72	0.53	51.15	0.96	41.03	1.14
	Reweight	59.55	0.55	53.11	0.21	42.51	0.58
	MixUp	64.85	0.80	55.14	0.78	45.50	0.68
	SRCC (L_c)	67.80	0.14	58.72	0.86	48.65	0.61
	SRCC ($L_{rc} + v_1$)	67.88	0.33	59.12	0.84	50.38	0.84
	SRCC ($L_{rc} + v_2$)	**69.52**	0.72	**61.71**	0.45	**50.98**	1.60

robustness and superiority of our proposed algorithm.

4) Sensitivity of Hyper-parameter α: We evaluate the sensitivity of MixUp and SRCC with respect to different mixing coefficients controlled by α. The test error rates on CIFAR10 are reported in Table VI. Each model is run three times in this experiment. It can be seen that our SRCC is less sensitive to the hyper-parameter than MixUp. When α is set to 0.5, the performance variance of MixUp is much higher than ours. It implies that our proposed method is more robust to different parameters than MixUP.

5) Effectiveness of Class Centroids: To study the effects of reweighting strategies for updating class centroids, we carry out experiments on CIFAR10 and CIFAR100 in different noise rates. We report test accuracy in Table VII. L_{rc} and L_c denote the class centroids with or without reweigthing original images. v_1 and v_2 denote that the weights of mixed inputs are normalized by Eqn. (8) and Eqn. (9) respectively. It shows that the models with reweighting outperform those with MixUp on both CIFAR10 and CIFAR100. Furthermore, we observe that a model updating class centroids with weighted features outperforms the one updating class centroids with the mean of sample features. This also demonstrates that our SRCC is able to recognize reliable data when updating the class centroids.

TABLE VIII: Overall test accuracy of models with different centroids update strategies on CIFAR10.

Methods	Noise Rate 0.2		Noise Rate 0.4		Noise Rate 0.6	
Identical sampling	91.02	1.71	87.35	0.67	79.04	2.52
Offline	**92.75**	0.46	**88.70**	0.89	79.62	2.64
SRCC	91.50	0.92	87.98	0.45	**80.74**	1.20

6) Effects of Class Centroid Update Schema: We further investigate the effects of different centroids update strategies on the model performance. As shown in Table VIII, we adopt three ways to update class centroids. The "Identical sampling" denotes the model using same batch samples to produce mixed inputs and update class centroids. The "Offline" represents the model updating class centroids using all training samples after a training epoch. It is observed that the model that samples another batch of images for generating mixed inputs (*i.e.*, our SRCC) indeed outperforms the "Identical Sampling" model.

VI. Conclusion

In this paper, we proposed a novel reweighting method, dubbed self-reweighting from class centroids (SRCC), for learning with noisy labels. Our method exploits the class centers to measure the reliability of data labels in computing the objective function, thus being more robust to corrupted labels. Furthermore, we also reweight class centroids to remove the noisy data in an online fashion. By doing so, we significantly reduce the computational cost while maintaining the effectiveness of the training process. The effectiveness of our proposed method is analyzed to show the advantage of our proposed method. Extensive experiments demonstrate that our SRCC achieves superior performance compared to the state-of-the-art on noisy data.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[3] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," in *Advances in Neural Information Processing Systems*, 2019, pp. 1917–1928.

[4] J. Liang, L. Jiang, D. Meng, and A. G. Hauptmann, "Learning to detect concepts from webly-labeled video data." in *IJCAI*, 2016, pp. 1746–1752.

[5] B. Zhuang, L. Liu, Y. Li, C. Shen, and I. Reid, "Attend in groups: a weakly-supervised deep learning framework for learning from web data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1878–1887.

[6] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1459–1469.

[7] W. Bi, L. Wang, J. T. Kwok, and Z. Tu, "Learning to predict from crowdsourced data." in *UAI*, 2014, pp. 82–91.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[9] F. Ma, L. Zhu, Y. Yang, S. Zha, G. Kundu, M. Feiszli, and Z. Shou, "Sf-net: Single-frame supervision for temporal action localization," in *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, pp. 420–437.

[10] R. Wang, T. Liu, and D. Tao, "Multiclass learning with partially corrupted labels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2568–2580, 2018.

[11] B. Han, I. W. Tsang, L. Chen, J. T. Zhou, and C. P. Yu, "Beyond majority voting: A coarse-to-fine label filtration for heavily noisy labels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 12, pp. 3774–3787, 2019.

[12] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1297–1322, 2010.

[13] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," *arXiv preprint arXiv:1803.09050*, 2018.

[14] M. Fang, T. Zhou, J. Yin, Y. Wang, and D. Tao, "Data subset selection with imperfect multiple labels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 7, pp. 2212–2221, 2019.

[15] B. Han, I. W. Tsang, L. Chen, C. P. Yu, and S. Fung, "Progressive stochastic learning for noisy labels," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 5136–5148, 2018.

[16] F. Ma, D. Meng, Q. Xie, Z. Li, and X. Dong, "Self-paced co-training," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2275–2284.

[17] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[18] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 1146–1151, 2013.

[19] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 38, no. 3, pp. 447–461, 2015.

[20] A. Menon, B. Van Rooyen, C. S. Ong, and B. Williamson, "Learning from corrupted binary labels via class-probability estimation," in *International Conference on Machine Learning*, 2015, pp. 125–134.

[21] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in neural information processing systems*, 2013, pp. 1196–1204.

[22] Y. Lyu and I. W. Tsang, "Curriculum loss: Robust learning and generalization against label corruption," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rkgt0REKwS

[23] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Advances in neural information processing systems*, 2018, pp. 8778–8788.

[24] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proceedings of IEEE International Conference on Computer Vision*, 2019, pp. 322–330.

[25] E. Amid, M. K. Warmuth, R. Anil, and T. Koren, "Robust bi-tempered logistic loss based on bregman divergences," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 987–14 996.

[26] Y. Xu, P. Cao, Y. Kong, and Y. Wang, "L_dmi: An information-theoretic noise-robust loss function," *arXiv preprint arXiv:1909.03388*, 2019.

[27] X. Yu, Y. Tian, F. Porikli, R. Hartley, H. Li, H. Heijnen, and V. Balntas, "Unsupervised extraction of local image descriptors via relative distance ranking loss," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[28] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann, "Easy samples first: Self-paced reranking for zero-example multimedia search," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 547–556.

[29] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," *arXiv preprint arXiv:1712.05055*, 2017.

[30] Y. Wu, Y. Lin, X. Dong, Y. Yan, W. Bian, and Y. Yang, "Progressive learning for person re-identification with one example," *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 2872–2881, June 2019.

[31] G. Kennedy, Z. Zhuang, X. Yu, and R. Mahony, "Iterative optimisation with an innovation cnn for pose refinement," *arXiv preprint arXiv:2101.08895*, 2021.

[32] D. Meng, Q. Zhao, and L. Jiang, "A theoretical understanding of self-paced learning," *Information Sciences*, vol. 414, pp. 319–328, 2017.

[33] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 1126–1135.

[34] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.

[35] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 113–123.

[36] Y. Wu and Y. Yang, "Exploring heterogeneous clues for weakly-supervised audio-visual video parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[39] H. Guo, Y. Mao, and R. Zhang, "Mixup as locally linear out-of-manifold regularization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3714–3722.

[40] V. Verma, A. Lamb, C. Beckham, A. Courville, I. Mitliagkis, and Y. Bengio, "Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer," *stat*, vol. 1050, p. 13, 2018.

[41] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems*, 2010, pp. 1189–1197.

[42] F. Ma, D. Meng, X. Dong, and Y. Yang, "Self-paced multi-view co-training," *Journal of Machine Learning Research*, vol. 21, no. 57, pp. 1–38, 2020. [Online]. Available: http://jmlr.org/papers/v21/18-794.html

[43] S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack," *The Journal of Open Source Software*, vol. 3, no. 24, Apr. 2018. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00638

[44] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[45] L. Yao and J. Miller, "Tiny imagenet classification with convolutional neural networks," *CS 231N*, vol. 2, no. 5, p. 8, 2015.

[46] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2691–2699.

[47] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Advances in neural information processing systems*, 2018, pp. 8527–8537.

[48] J. Shu, Q. Zhao, K. Chen, Z. Xu, and D. Meng, "Learning adaptive loss for robust learning with noisy labels," *arXiv preprint arXiv:2002.06482*, 2020.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[50] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[51] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[52] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1944–1952.