MixNN: Combating Noisy Labels in Deep Learning by Mixing with Nearest Neighbours

***Abstract*—Noisy labels are ubiquitous in real-world datasets, especially in the ones from web sources. Training deep neural networks on noisy datasets is a challenging task, as the networks have been shown to overfit the noisy labels in training, resulting in performance degradation. When trained on noisy datasets, deep neural networks have been observed to fit the clean samples during an "early learning" phase, before eventually memorize the mislabeled samples. We further explore the representation distributions in the early learning stage and find that the representations of similar samples from the same classes congregate regardless of their labels. Inspired by these findings, we propose MixNN, a novel framework to mitigate the influence of noisy labels. In contrast with existing methods, which identify and eliminate the mislabeled samples, we modify the mislabeled samples by mixing them with their nearest neighbours through a weighted sum approach. The weights are calculated with a mixture model learning from the sample loss distribution. To enhance the performance with the presence of extreme label noise, we propose a strategy to estimate the soft targets by gradually correcting the noisy labels. We demonstrate that the estimated targets yield a more accurate approximation to ground truth labels and a better quality of the learned representations with more separated and clearly bounded clusters. Extensive experiments in two benchmarks and two challenging real-world datasets demonstrate that our approach outperforms the existing state-of-the-art methods.**

***Index Terms*—noisy labels, deep learning, weakly supervised learning, image classification**

I. Introduction

Deep neural networks (DNNs) have achieved remarkable success in many applications (e.g. classification, detection and semantic segmentation) as a result of the novel network architectures, robust optimization algorithms and the emergence of large-scale training data with human annotated labels. However, it is extremely expensive and time-consuming to attain high-quality labels in many real-world settings. The large-scale datasets may contain samples which are "weakly-labeled" through either proxy variables or web scraping [21, 27, 57]. Although some non-expert sources, such as Amazon's Mechanical Turk and the surrounding tags of collected data, have been widely used to mitigate the high labeling cost. However, human annotators, especially on crowdsourcing platforms, can also be prone to mislabeling by mistakes. Even the most celebrated and highly-curated datasets, like ImageNet [8], famously contain harmful examples.

Since unrealiable datasets contain a certain percentage of samples with wrong labels which may be corrupted from ground-truth labels, training DNNs on these datasets is known to be highly affected. The significant number of model parameters which render DNNs even overfit to noisy labels. Zhang et al. [60] have demonstrated that DNNs can easily fit an entire training dataset with any percentage of corrupted labels, and result in poor generalization capacity on a test set. Zhu et al. [64] have observed that the accuracy drop by label noise is more substantial than by other noises, such as feature noise. Therefore, it is of great significance to develop algorithms that achieve a robust generalization capability in the presence of noisy labels.

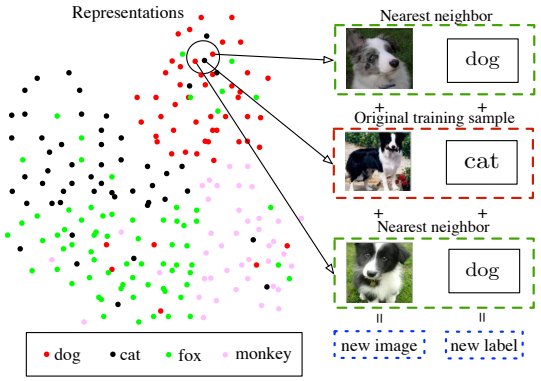


Fig. 1: An example for illustrating the way to generate a new training sample. Based on the learned representations, the new training sample is generated by mixing itself with its nearest neighbours.

Given a training dataset contains correctly labeled samples and mislabeled samples, a common approach to mitigate the negative influence of mislabeled samples is to identify and eliminate them in the first stage and train DNNs solely with the remaining correctly labeled samples in the second stage [19, 22, 48]. The filter mechanism to distinguish the mislabeled samples from the others in the first stage is decisive to the task performance of the second stage. If the filter mechanism removes few samples, the unfiltered mislabeled samples still influence the (supervised) loss and affect the task performance as in these previous works. If it eliminates too many samples, the remaining training data may not be rich enough to effectively generalize to held-out data in the second stage. Thus, is it possible to robustly train DNNs on noisy datasets without removing the informative training samples?

In this paper, we propose a novel approach to combat the negative influence of mislabeled samples by generating new samples to train the DNNs. As shown in Fig. 1, for each training sample, we search its nearest neighbours based on the learned feature embeddings (representations). Then we linearly

combine their images and labels to obtain a new sample. By using the new samples, it minimizes the noisy supervision from the original mislabeled samples and improves the task performance. Take the three samples in Fig. 1 as an example – assume we average their images and labels, then the new image is a dog image mixture and the new label is $2/3$ dog and $1/3$ cat. The new image is similar to the original image applied with cutout augmentation [9], while the new label is similar to the original label applied with label smoothing [33]. It is easy to see that training DNNs with this new sample is much more reasonable than with the original sample (i.e. a dog image with a wrong label cat).

Instead of simply averaging the above selected samples to get a new one, we assign a dynamic weight to each selected sample. The weight is calculated by fitting a two-component mixture model to the sample loss distribution, so that it indicates the 'probability' of a sample being clean or not. It is supported by the observation that DNNs learn from clean samples before memorizing the noisy ones during training. In particular, DNNs learn from correctly labeled samples at ease and receive inconsistent error supervison from the mislabeled samples before over-fitting to the entire dataset. Therefore, the networks prediction is likely to be consistent on correctly labeled samples and inconsistent on mislabeled samples, resulting in the seperation of their loss values.

To further enhance the robustness of our approach, we propose an exponential moving average strategy to estimate ground truth labels (targets) based on model predictions and given noisy labels. In other words, the noisy labels are gradually being corrected as the estimated soft targets. By using these targets, the label quality of new training samples is improved. Subsequently, the performance of dynamic weight estimation becomes more accurate, providing a more stable supervisory signal to DNNs.

In summary, our learning framework stabilizes the training process and improves the generalization capacity of DNNs. Our main contributions are summarized as follows:

- We provide insights to the learning procedure and representation distributions of training with noisy labels. Based on these findings, we propose to generate new training samples to robustly train DNNs, by mixing the orginal samples with their nearest neighbours.
- We fit a Gaussian Mixture Model to the sample loss distribution and calculate the dynamic weights that support the generation of new training samples.
- We propose an exponential moving average strategy to calculate the soft targets to estimate the ground truth labels, which enhances the task performance even in the presence of extreme label noise.
- We empirically demonstrate that the proposed approach outperforms the state-of-the-art methods on two standard benchmarks with simulated label noise and two real-world noisy datasets. We also provide empirical analysis (e.g. feature representations and noisy label correction) of our approach for better understanding.

Besides, our method does not require any prior knowledge of the type or the amount of noisy data. It does not require any tuning of hyperparameters based on prior knowledge, making our method applicable to real life.

The remainder of this paper is organized as follows: Section II introduces the related work on classification with noisy labels. Section III explores the early learning phenomenon and explains the failure of using cross entropy loss when learning with noisy labels. Section IV describes each part of our method with detailed explanations. Section V details the experiments including comparison with other state-of-art methods and provides empirical studies for deep understanding of our approach. Finally, we conclude the paper in Section VI.

II. Related work

Different approaches have been proposed to combating noisy labels in image classification task, and they can be divided into following categories:

Robust loss functions. These studies focus on developing noise-tolerant loss functions [5, 11, 34, 38, 52, 53, 55, 62]. For example, Ghosh et al. [11] use mean absolute error (MAE) as noise-tolerant loss function, while the accuracy degrades when adopted in deep neural networks [62]. Wang et al. [55] boost the cross entropy (CE) loss symmetrically by adding a reverse cross entropy loss to avoid overfitting to noisy labels. GCE [62] applies a Box-Cox transformation to probabilities which behaves like a generalized mixture of MAE and CE.

Loss correction and Label correction. These approaches either iteratively relabel the noisy labels with their own predictions [47, 54, 58] or estimate the noise transition matrix [13, 18, 41]. For example, Patrini et al. [41] estimate the noise transition matrix and equally treat all samples to correct the loss. Joint-optim [47] iteratively updates the labels with soft or hard pseudo-labels. PENCIL [58] refines the relabel procedure without using prior information about noisy labels. Reed et al. [43] propose a bootstrapping method which corrects the labels by predictions. D2L [35] improves the bootstrapping method by exploiting the dimensionality of feature subspaces.

Sample selection These methods [15, 20, 31, 36, 56, 59] select clean instances for effective training of DNNs. MentorNet [20] pre-trains a mentor network for selecting clean instances to guide the training of the student network. Co-teaching [15] and Co-teaching+ [59] symmetrically train two networks by selecting small-loss instances in a mini-batch to teach the other. In [29, 44], cleaner instances are assigned more weights for better update on the classifier. Kim et al. [22] use Negative Learning to select noise instances.

Semi-supervised learning and meta-learning. These methods either apply semi-supervised learning techniques after explicitly differentiating noisy samples from training data [10, 22, 25, 40] or use meta-learning [14, 26]. For example, Li et al. [25] divide the training data into clean and noisy ones, then train two networks with a semi-supervised algorithm MixMatch [4]. SELF [40] progressively filters out misabeled samples with a semi-supervised approach.

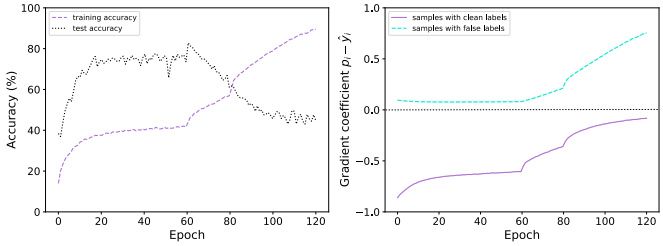


Fig. 2: We train ResNet34 [17] on CIFAR-10 dataset with 60% symmetric label noise using cross entropy loss. Left: The train and test accuracy vs. the number of training epochs. Right: The gradient coefficient $\boldsymbol{p}_i - \hat{\boldsymbol{y}}_i$ of clean and false labels vs. the number of training epochs.

Other approaches: Wang et al. [54] apply a Siamese network with a contrastive loss to iteratively pull noisy samples away from the clean ones. Huang et al. [19] use a heuristic way to adjust the learning rate to prevent DNNs from overfitting to label noise. Lu et al. [32] propose a confidence adatpive regularization to robust learning with noisy labels. Han et al. [16] use the clustering algorithm to find multiple prototypes for correcting noisy labels.

In contrast to the aforementioned literature, our method deals with noisy labels without: 1) consulting any clean subset; 2) eliminating training samples; 3) applying augmentation techniques from semi-supervised learning; 4) using any prior information. Specifically, we train DNNs with a new training set that generated by mixing the original samples with their nearest neighbours, preventing DNNs from overfitting to noisy labels in training.

III. Weakness of Cross Entropy

Our work aims to develop an algorithm to train a classifier which achieves robust performance on the test set when the provided training set contains noisy labels. Before we introduce our method, we first describe the preliminary of classification with noisy labels. Then we provide some insights to the learning procedure of deep neural networks and analyze the gradient to explain the weakness of using cross entropy loss when training DNNs with noisy labels.

A. Preliminary

Consider the C-class classification problem, we have a training set $\hat{D} = \{(\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i)\}_{i=1}^N$, where $\boldsymbol{x}_i$ is an input and $\hat{\boldsymbol{y}}_i \in \{0,1\}^C$ is the one-hot vector corresponding to $\boldsymbol{x}_i$. In the noisy label scenario, ground truth label $\boldsymbol{y}_i$ is unavailable, and the given noisy label $\hat{\boldsymbol{y}}_i$ is of certain probability to be incorrect. The classification model maps each input $\boldsymbol{x}_i$ to a C-dimensional logits using a deep neural network model $\mathcal{N}_\Theta$ and then feeds the logits into a softmax function to produce $\boldsymbol{p}_i$ of the conditional probability of each class.

$$\boldsymbol{p}_i = \texttt{softmax}(\mathcal{N}_\Theta(\boldsymbol{x}_i)) = \frac{e^{\mathcal{N}_\Theta(\boldsymbol{x}_i)}}{\sum_{c=1}^{C} e^{(\mathcal{N}_\Theta(\boldsymbol{x}_i))_c}}. \quad (1)$$

Θ denotes the parameters of the model and $(\mathcal{N}_\Theta(\boldsymbol{x}_i))_c$ denotes the c-th entry of logits $\mathcal{N}_\Theta(\boldsymbol{x}_i)$. Usually, the model $\mathcal{N}_\Theta$ is trained via the cross entropy (CE) loss to measure how well the model fits the training samples.

$$\mathcal{L}_{ce}(\Theta) = -\frac{1}{N}\sum_{i=1}^{N}\{\ell_{ce}\}^i = -\frac{1}{N}\sum_{i=1}^{N}\hat{\boldsymbol{y}}_i^T \log(\boldsymbol{p}_i). \quad (2)$$

B. Learning Procedure of CE under Noisy Labels

When trained with noisy labels, the overparameterized deep neural networks have been observed to first fit the training data with clean labels during an *early learning* stage, before eventually memorizing the examples with false labels [2, 60]. Reflected on training and test accuracy in Figure 2, the model achieves maximum test accuracy before achieving highest training accuracy. More specifically, the model starts by learning to predict the true labels for correctly labeled training samples. Thus it can predict correct labels for clean test data. However, with the increasing number of training epochs, the model begins making incorrect predictions as it memorizes the mislabeled samples.

C. Gradient Analysis of CE

To further explain the early learning phenomenon, we drive the gradient of cross entropy loss with respect to Θ as follows:

$$\nabla\mathcal{L}_{ce}(\Theta) = -\frac{1}{N}\sum_{i=1}^{N}\nabla\mathcal{N}_\Theta(\boldsymbol{x}_i)(\boldsymbol{p}_i - \hat{\boldsymbol{y}}_i), \quad (3)$$

where $\nabla\mathcal{N}_\Theta(\boldsymbol{x}_i)$ is the Jacobian matrix of the neural network logits for the i-th input with respect to Θ. In clean training data scenario, $\boldsymbol{p}_i - \boldsymbol{y}_i$ of true class entry will always be negative and the rest entries are positive. Therefore, performing stochastic gradient descent increases the probability of true class and reduces the residual probabilities at other classes, which promises the learning to continue on true class. However, in the noisy label scenario, if c is the true class, but c-th entry of noisy label $(\hat{\boldsymbol{y}}_i)_c = 0$, then the contribution of the i-th sample to $\nabla\mathcal{L}_{ce}(\Theta)$ is reversed (i.e. $(\boldsymbol{p}_i - \hat{\boldsymbol{y}}_i)_c$ should be negative but get positive instead). In the meanwhile, the entry corresponding to the impostor class c', is also reversed because $(\hat{\boldsymbol{y}}_i)_{c'} = 1$. Thus, for samples with clean labels, the cross entropy term $\boldsymbol{p}_i - \hat{\boldsymbol{y}}_i$ tends to vanish (i.e. closer to zero) after the early learning stage because $\boldsymbol{p}_i$ is close to $\boldsymbol{y}_i$. For samples with false labels, the cross entropy term $(\boldsymbol{p}_i - \hat{\boldsymbol{y}}_i)_c$ is positive, allowing them to dominate the gradient. The right plot in Figure 2 shows the change of gradient coefficient $(\boldsymbol{p}_i - \hat{\boldsymbol{y}}_i)_c$ in training. The gradients of correctly labeled samples dominate at the beginning, and then are gradually suppressed by the gradients of mislabeled samples. Therefore, performing stochastic gradient descent eventually results in the memorization of mislabeled samples.

IV. Methodology

As shown in Fig. 3, our approach contains three parts (boxed with different colors). In this section, we first explore the representation distributions in the early learning stage. Then

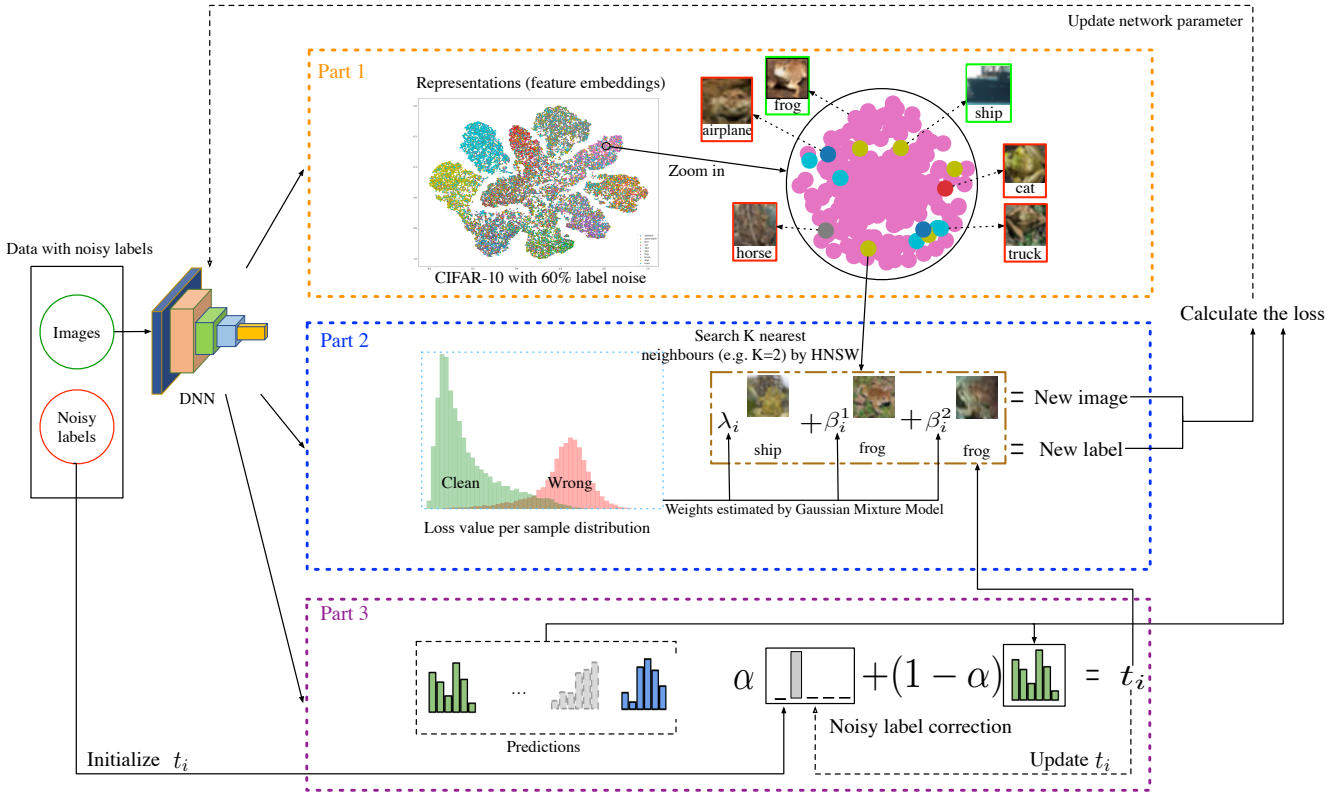


Fig. 3: The proposed method MixNN consists of three parts. Part 1: Based on the feature embeddings from the penultimate layer, we calculate each training sample's approximate K-nearest neighbours by using Hierarchical Navigable Small World (HNSW) graph. Part 2: We mix the original sample with its K-nearest neighbours by using the dynamic weights estimated from a Gaussian Mixture Model that learned on sample loss distribution. Part 3: We estimate the soft targets that gradually correct the noisy labels through an exponential moving average strategy.

we describe the details of each part in MixNN, including mixing with nearest neighbours, K-approximate nearest neighbour search, weight estimation, and noisy labels correction. Finally, we discuss the possible cases in mixing with nearest neighbours and explain one limitation of our approach.

A. Representation Distributions

In the noisy label scenario, the training data consist of correctly labeled samples and mislabeled samples. The goal of our approach is to prevent the model from memorizing mislabeled samples while learning from correctly labeled samples. We plot the t-SNE [51] of representations (feature embeddings) in the early learning stage in Fig. 3 Part 1. As we can see, the representations of majority correctly labeled samples congregate in their own classes, while the representations of mislabeled samples disperse in all classes. For instance, by zooming in a random region in frog (pink) class and display the samples in the right for clear illustration. We observe that most of the frog images have learned correct representations, only a few samples labeled by other classes have learned 'corrupted' representations. Besides, most of the mislabeled samples in frog class are also frog images but attached false labels from other classes. In our experiments, we find that 88.96% and 82.8% mislabeled samples' representations still congregate in their true class for CIFAR-10 with 40% and 60% label noise respectively.

This observation motivates us to think whether we can refer to the information from the nearest neighbours to achieve robustness to label noise. To achieve our goal, we generate a new training set where each sample is mixed with its nearest neighbours, so that the information of its correct feature is preserved while the negative impact of noisy label can be mitigated.

B. Dynamic Mixing with Nearest Neighbors

Our basic idea is referring to the knowledge from the nearest neighbours to mitigate the negative influence of noisy labels. For each training sample $\boldsymbol{x}_i$ in a mini-batch, we generate a new training sample $\tilde{\boldsymbol{x}}_i$ by linearly combining it with its K nearest neighbours. We denote K nearest neighbours of a training sample $(\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i)$ as $\Phi^K(\boldsymbol{x}_i) = \{(\bar{\boldsymbol{x}}_i^k, \bar{\boldsymbol{y}}_i^k)\}_{k=1}^K = \{(\bar{\boldsymbol{x}}_i^1, \bar{\boldsymbol{y}}_i^1), (\bar{\boldsymbol{x}}_i^2, \bar{\boldsymbol{y}}_i^2), \ldots, (\bar{\boldsymbol{x}}_i^K, \bar{\boldsymbol{y}}_i^K)\}$. Hence, the new training sample is

$$\tilde{\boldsymbol{x}}_i = \lambda_i \boldsymbol{x}_i + \sum_{k=1}^{K} \beta_i^k \bar{\boldsymbol{x}}_i^k, \quad \sum_{k=1}^{K} \beta_i^k = 1 - \lambda_i, \tag{4}$$

where λ_i is a dynamic scalar value denoting the weight of original training sample. β_i^k denotes the weight of k-th nearest neighbour. We have $\lambda_i + \sum_{k=1}^{K} \beta_i^k = 1$ to ensure the new training sample still follows the same distribution of original sample after normalization. Similarly, we calculate the new label $\tilde{\boldsymbol{y}}_i$ of the new sample $\tilde{\boldsymbol{x}}_i$ by

$$\tilde{\boldsymbol{y}}_i = \lambda_i \hat{\boldsymbol{y}}_i + \sum_{k=1}^{K} \beta_i^k \bar{\boldsymbol{y}}_i^k, \quad \sum_{k=1}^{K} \beta_i^k = 1 - \lambda_i. \tag{5}$$

We then train our model with the new training set $\tilde{D} = \{(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^{N}$. Based on $\tilde{D}$, we use the cross entropy loss as the measure of how well the model fits the $\tilde{D}$. We denote

$$\tilde{\boldsymbol{p}}_i = \texttt{softmax}(\mathcal{N}_\Theta(\tilde{\boldsymbol{x}}_i)) = \frac{e^{\mathcal{N}_\Theta(\tilde{\boldsymbol{x}}_i)}}{\sum_{c=1}^{C} e^{(\mathcal{N}_\Theta(\tilde{\boldsymbol{x}}_i))_c}}. \tag{6}$$

Thus, our total loss is

$$\begin{aligned} \mathcal{L}(\tilde{D}, \Theta) &= -\frac{1}{N} \sum_{i=1}^{N} \tilde{\boldsymbol{y}}_i^T \log(\tilde{\boldsymbol{p}}_i) \\ &= -\frac{1}{N} \sum_{i=1}^{N} (\lambda_i \hat{\boldsymbol{y}}_i + \sum_{k=1}^{K} \beta_i^k \bar{\boldsymbol{y}}_i^k)^T \log(\tilde{\boldsymbol{p}}_i) \end{aligned} \tag{7}$$

C. Approximate Nearest Neighbor Search

In our method, we need to search the K-Nearest Neighbors (KNN) for each training sample based on the feature representation. Assume the feature representation of a training sample is a query vector. A naive approach to performing exact KNN search is to directly compute the distances (e.g. Euclidean distance and cosine distance) between the query and every element in the training set and to select the elements with minimal distance. Hence, the complexity of the naive approach is $O(dN)$, where N is the size of training set and d is the dimension of representation.

Previous studies have shown that exact KNN search solutions may offer a substantial search speedup only in the case of relatively low dimentional data (e.g. $d < 20$) due to "curse of dimensionality". For instance, the complexity of KNN search in KD-tree [3] is $O(2^d \log(N))$, which is exponential to the dimension of representations. However, d can be very large in our case. For example, the dimension of representation in penultimate layer for ResNet34 [17] is 512. Therefore, it is inefficient to directly calcuate the exact KNN.

To overcome this problem, a concept of Approximate Nearest Neighbours Search (ANNS) was proposed, which relaxes the condition of the exact search by allowing a small number of errors. The quality of an inexact search is defined as the ratio between the number of found true nearest neighbours and K. In this paper, we adopt the Hierarchical Navigable Small World (HNSW) graph [37] as our ANNS index. It is a fully graph based incremental ANNS structure that can offer a superior logarithmic complexity scaling. The search index in HNSW is a multi-layered structure where each layer is a proximity graph. Each node in the graph corresponds to one of our feature representations. A nearest neighbour search in HNSW uses a "zooming-in" approach. It starts at an entry point node in the uppermost layer and recursively performs a greedy graph traversal in each layer until it reaches a local minimum in the bottommost one. The maximum number of connections per element in all layers can be made a constant, thus allowing a logarithmic complexity scaling of routing in a navigable small world graph. In this paper, we use Euclidean distance as the meature of similarity and the overall search complexity scaling is $O(\log(N))$.

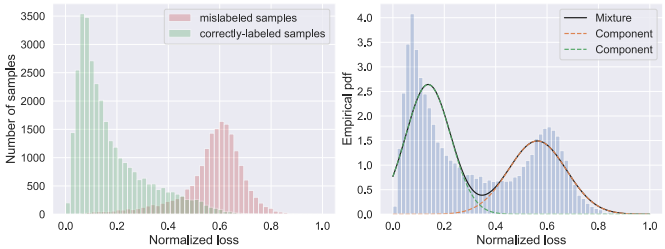


Fig. 4: Train on CIFAR-10 with 40% label noise after 10 epochs with cross entropy loss. Left: The ground truth normalized loss distribution. Right: The pdf of mixture model and two components after fitting a two-component GMM to loss distribution.

D. Weight Estimation

In Eq. (4), Eq. (5) and Eq. (7), we need the weight λ_i to indicate how confidently we can trust the original sample and $\beta_i^1, ..., \beta_i^K$ to indicate how much knowledge referred from the nearest neighbours. Intuitively, we desire to preserve more knowledge from the correctly labeled samples but to discard the knowledge from the mislabeled samples. In other words, the weights should be able to indicate the 'probability' of a training sample being correctly labeled or not.

Previous sample selection methods select the correctly-labeled samples by ranking their loss values. Due to the early learning phenomenon, the samples with small-loss values are more likely to be correctly labeled [15]. In this paper, we observe that the correctly-labeled samples can be distinguished from the loss distribution solely. As shown in Fig. 4 left plot, the normalized loss values of the correctly labeled samples are smaller than the mislabeled ones. To estimate the probability of a sample being correctly-labeled, we introduce a two-component Gaussian Mixture Model (GMM) [42] to fit the normalized loss distribution as shown in Fig. 4 right plot. The probability density function (pdf) of GMM with M components on the per sample loss value ℓ can be defined as

$$P(\ell) = \sum_{m=1}^{M} \pi_m \mathcal{G}(\ell \mid \mu_m, \sigma_m^2), \quad \sum_{m=1}^{M} \pi_m = 1, \tag{8}$$

where π_m are the mixing coefficient for the linear convex combination of each individual pdf $\mathcal{G}(\ell \mid \mu_m, \sigma_m^2)$. In our case, we use an Expectation-Maximization (EM) algorithm to estimate the π_m, μ_m and σ_m^2. Therefore, we can obtain the

probability of a sample being correctly-labeled or mislabeled through the posterior probability:

$$P(m \mid \ell) = \frac{P(m)P(\ell \mid m)}{P(\ell)} = \frac{\pi_m \mathcal{G}(\ell \mid \mu_m, \sigma_m^2)}{\sum_{m=1}^{M} \pi_m \mathcal{G}(\ell \mid \mu_m, \sigma_m^2)} \tag{9}$$

where $m = 0(1)$ indicate correct (wrong) labels. Note that we always calculate the cross entropy loss to estimate the clean probability for all samples after every epoch. But we use our loss defined in Eq. (7) for training the model which contains other loss terms to deal with label noise.

While mislabeled samples benefit from combining with correctly labeled ones, correctly labeled samples are contaminated by mislabeled ones, whose training objective is incorrectly modified. The goal of mixing strategy in Eq. (4) and Eq. (5) is to use the dynamic weights to reduce the contribution of mislabeled samples when they are combined with correctly labeled ones. We denote the per sample loss value of $\boldsymbol{x}_i$ as $\ell(\boldsymbol{x}_i)$. Thus the dynamic weights are calculated by

$$\lambda_i = \frac{P(m = 0 \mid \ell(\boldsymbol{x}_i))}{P(m = 0 \mid \ell(\boldsymbol{x}_i)) + \sum_{k=1}^{K} P(m = 0 \mid \ell(\bar{\boldsymbol{x}}_i^k))}, \tag{10}$$

$$\beta_i^k = \frac{P(m = 0 \mid \ell(\bar{\boldsymbol{x}}_i^k))}{P(m = 0 \mid \ell(\boldsymbol{x}_i)) + \sum_{k=1}^{K} P(m = 0 \mid \ell(\bar{\boldsymbol{x}}_i^k))}. \tag{11}$$

We then use the above weights to guide the generation of new training sample $(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i)$. Assume $K = 1$, we have four cases: clean-clean, clean-wrong, wrong-clean, and wrong-wrong. By using dynamic weights, it largely avoids generating the confusing input to the network in clean-wrong and wrong-clean cases, while retaining the strengths for clean-clean and wrong-wrong combinations.

E. Noisy Labels Correction

Despite that the new training samples set $\tilde{D} = \{(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^{N}$ is better than directly using given noisy training dataset $\hat{D} = \{(\boldsymbol{x}_i, \hat{\boldsymbol{y}}_i)\}_{i=1}^{N}$. Nevertheless, using the noisy labels $\hat{\boldsymbol{y}}_i$ in Eq. (5) and Eq. (7) may be less effective as $\hat{\boldsymbol{y}}_i$ is incorrect with a certain probability, especially when the noise rate is extremely high. Therefore, we need a better estimation of ground truth label $\boldsymbol{y}_i$. In Fig. 2, we oberve that most predictions in early learning stage are correct. Based on this observation, we propose an exponential moving average strategy to gradually estimate the soft target $\boldsymbol{t}_i$ by using the noisy label $\hat{\boldsymbol{y}}_i$ and model prediction $\boldsymbol{p}_i$. We update $\boldsymbol{t}_i$ in each epoch E by

$$\boldsymbol{t}_i = \begin{cases} \hat{\boldsymbol{y}}_i & \text{if } E < E_s \\ \alpha \boldsymbol{t}_i + (1-\alpha)\boldsymbol{p}_i & \text{if } E \geq E_s \end{cases} \tag{12}$$

where E_s is the epoch that starts performing label correction and $0 \leq \alpha < 1$ is the momentum. As E_s is not sensitive to performance, we fix $E_s = 60$ and $\alpha = 0.9$ by default. We then replace the noisy label $\hat{\boldsymbol{y}}_i$ in Eq. (5) and Eq. (7) with the estimated target $\boldsymbol{t}_i$. Consequently, using a better $\boldsymbol{t}_i$ facilitate the model memorize more correctly labeled samples. The correction accuracy will be discussed in Section V-C. Overall, put all parts together, our algorithm is described in Algorithm 1.

Algorithm 1: MixNN

Input: Networks $\mathcal{N}_\Theta$ with parameters Θ, training set $\hat{D}$, number of nearest neighbours K, batch size B, learning rate η, number of total training epochs $E_{\max}$, $E_s = 60$ momentum $\alpha = 0.9$;

$\Theta =$ Warmup$(\hat{D}, \Theta, \mathcal{L}_{ce})$; `// warmup with cross entropy loss.`

Initialize the target $\boldsymbol{t}_i = \hat{\boldsymbol{y}}_i$ for all samples in $\hat{D}$;

for $e = 1, 2, \ldots, E_{max}$ **do**

  $P(m = 0 \mid \ell_{ce}(\boldsymbol{x}_i)) =$ GMM $(\hat{D}, \ell_{ce}, \Theta)$; `// fit GMM to sample loss distibution.`

  Obtain K Approximate Nearest Neighbours $\Phi^K(\boldsymbol{x}_i) = \{(\bar{\boldsymbol{x}}_i^k, \bar{\boldsymbol{y}}_i^k)\}_{k=1}^{K}$ for each $\boldsymbol{x}_i$ by using HNSW;

  Shuffle $\hat{D}$ into $\frac{|\hat{D}|}{B}$ mini-batches ;

  for $n = 1, 2, \ldots, \frac{|\hat{D}|}{B}$ **do**

    Fetch n-th mini-batch $\hat{D}_n$ from $\hat{D}$;

    Obtain $\boldsymbol{p}_i$ for each sample in $\hat{D}_n$ by Eq. (1);

    if $e \geq E_s$ **then**

      Update $\boldsymbol{t}_i = \alpha \boldsymbol{t}_i + (1-\alpha)\boldsymbol{p}_i$ for all sample in $\hat{D}_n$;

    Estimate the weights $\lambda_i, \beta_i^1, ..., \beta_i^K$ for each sample in $\hat{D}_n$ by Eq. (10) and Eq. (11).;

    Generate $\tilde{D}_n = \{(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i)\}_{i=1}^{B}$ by Eq. (4), Eq. (5) and $\Phi^K(\boldsymbol{x}_i)$;

    Obtain $\tilde{\boldsymbol{p}}_i$ for each sample in $\tilde{D}_n$ by Eq. (6);

    Calculate the loss $\mathcal{L}(\tilde{D}_n, \Theta) = -\frac{1}{B}\sum_{i=1}^{B} \lambda_i \boldsymbol{t}_i^T \log(\tilde{\boldsymbol{p}}_i) - \frac{1}{B}\sum_{i=1}^{B}\sum_{k=1}^{K} \beta_i^k (\bar{\boldsymbol{y}}_i^k)^T \log(\tilde{\boldsymbol{p}}_i)$;

    Update $\Theta = \Theta - \eta \nabla \mathcal{L}(\tilde{D}_n, \Theta)$;

Output Θ.

F. Discussion and Limitation

To explain how MixNN works, we discuss all cases of a simplified version (when $K = 1$) for generating the new mixed sample $(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i)$ as follows.

Case 1: **Both $\boldsymbol{x}_i$ and its nearest neighbour $\bar{\boldsymbol{x}}_i^1$ are correctly labeled samples.** Due to the analysis of representation distributions in the early learning phase, both $\boldsymbol{x}_i$ and $\bar{\boldsymbol{x}}_i^1$ are most likely to be the samples from the same class. In this case, the new mixed input is a convex linear combination of two similar images, i.e., $\tilde{\boldsymbol{x}}_i = \lambda_i \boldsymbol{x}_i + \beta_i^1 \bar{\boldsymbol{x}}_i^1$. And its corresponding label is $\tilde{\boldsymbol{y}}_i = \lambda_i \hat{\boldsymbol{y}}_i + \beta_i^1 \bar{\boldsymbol{y}}_i^1$. Since $\hat{\boldsymbol{y}}_i = \bar{\boldsymbol{y}}_i^1$ and $\lambda_i + \beta_i^1 = 1$, we have $\tilde{\boldsymbol{y}}_i = \hat{\boldsymbol{y}}_i = \bar{\boldsymbol{y}}_i^1$. Therefore, the mixed sample $(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i)$ in this case is similar to data augmentation strategy [61] that encourages the model to behave linearly in-between training samples, resulting in reducing the amount of undesirable oscillations when predicting hard samples. For instance, assume $\lambda_i = 0.5$ and $\beta_i^1 = 0.5$, then $\tilde{\boldsymbol{x}}_i$ consists of half $\boldsymbol{x}_i$ and half $\bar{\boldsymbol{x}}_i^1$. The DNNs trained with such samples are significantly better calibrated [49], i.e., the prediction softmax scores are much better indicators of the actual likelihood of a correct prediction, improving the estimation of dynamic weights in Section IV-D.

Case 2: **$\boldsymbol{x}_i$ is correctly labeled sample and its nearest neighbour $\bar{\boldsymbol{x}}_i^1$ is mislabeled sample.** Similar to case 1, the new

TABLE I: Comparison of different label smearing methods. Here $\mathbf{I}$ denotes the identity and $\mathbf{J}$ the all-ones matrix. For backward correction, the theoretical optimal choice of $\alpha = \frac{C}{C-1} \cdot \varepsilon$, where ε is the level of label noise. For MixNN, $\mathbf{Y}$ is a $C \times C$ matrix generated by the one-hot labels of the original sample and its nearest neighbours. $\mathbf{B}$ is a $C \times 1$ vector where the entries are filled by dynamic weights $\lambda, \beta^1, \beta^2, ..., \beta^K$ with corresponding to their labels.

Method	Label after scaling
CE	$\mathbf{I} \cdot \hat{\boldsymbol{y}}$
Label smoothing [33]	$\left[(1-\gamma) \cdot \mathbf{I} + \frac{\gamma}{C} \cdot \mathbf{J}\right] \cdot \hat{\boldsymbol{y}}$
Backward correction [41]	$\left[\frac{1}{1-\gamma}\mathbf{I} - \frac{\gamma}{(1-\gamma)\cdot C} \cdot \mathbf{J}\right] \cdot \hat{\boldsymbol{y}}$
MixNN (Ours)	$\mathbf{Y} \cdot \mathbf{B} = \begin{bmatrix} \cdots \\ \hat{\boldsymbol{y}} \\ \cdots \\ \tilde{\boldsymbol{y}}^1 \\ \cdots \\ \tilde{\boldsymbol{y}}^K \\ \cdots \end{bmatrix} \cdot \begin{bmatrix} \cdots \\ \lambda \\ \cdots \\ \beta^1 \\ \cdots \\ \beta^K \\ \cdots \end{bmatrix}$

mixed input is most likely to be combined with two images from the same class, while their labels are inconsistent. Our method to generate the corresponding label is similar to label smoothing [33] which modifies the label $\hat{\boldsymbol{y}}_i$ to $(1-\gamma)\hat{\boldsymbol{y}}_i + \gamma/C$ (e.g. $\gamma = 0.5$). This corresponds to a scaling and translation of the original noisy label, but preserves the label with maximal probability, provided $\gamma < 1$. For clarity, we compare the label after scaled by different approaches in Table I. Unlike the label smoothing and Backward correction that use the fixed weights to scale the noisy label, our approach adopts the dynamic weights, where $\lambda, \beta^1, \ldots, \beta^K$ are learned from data, to adaptively adjust the smearing matrix for better calibration.

Case 3: $\boldsymbol{x}_i$ **is mislabeled sample and its nearest neighbour** $\bar{\boldsymbol{x}}_i^1$ **is correctly labeled sample.** This case is similar with case 2, so we do not further discuss it.

Case 4: **Both** $\boldsymbol{x}_i$ **and its nearest neighbour** $\bar{\boldsymbol{x}}_i^1$ **are mislabeled samples.** In this case, the new mixed input is combined with two images from the different classes, and their labels are also inconsistent. Thus the mixed sample is not promised to improve the generalization capacity. It is the main limitation of our approach as this case may degrade the performance. However, this case rarely occurs. In our experiments, we find that this case occurs in only 4.3% of the training data when trained on CIFAR-10 with 60% label noise.

V. Experiments

This section consists of three parts. We first test the efficacy of our method on datasets with simulated label noise and show its robustness by comparing with other existing approaches. Then we evaluate its performance on the real-world datasets which contain more complex label noise such as instance-dependent noise. Finally, we conduct empirical studies towards a deeper understanding of our approach, including feature representations and label correction accuracy. All experiments are implemented in Pytorch and run on a single Nvidia A100 GPU.

A. Effectiveness on Datasets with Simulated Label Noise

We conduct the experiments with simulated label noise on the following two datasets.

- **CIFAR-10** dataset [23] consists of natural colour images, each of size 32 × 32 pixels. Each image is classified into 1 of 10 classes, such as dog, cat, automobile, or ship. The training set contains 50,000 images, while the test set contains 10,000 images.
- **CIFAR-100** dataset [23] is similar with CIFAR-10, except it has 100 classes contains 600 images each. There are 500 training images and 100 testing images per class. The 100 classes are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

Label Noise Simulation: Since CIFAR-10 and CIFAR-100 are initially clean, we follow the way in [41, 43] to corrupt these datasets manually by label transition matrix Q, where $Q_{ij} = Pr[\hat{y} = j \mid y = i]$ given that noisy label $\hat{y}$ is flipped from clean label y. Generally, the matrix Q has two representative label noise models: (1) Symmetric noise [52] is generated by uniformly flipping labels in each class to one of the other class labels with probability ε. (2) Asymmetric noise [41] is a simulation of fine-grained classification with noisy labels in the real world, where the labelers are more likely to make mistakes only within very similar classes. Fig. 5 shows an example of noise transition matrix Q for above two label noise models. Specifically, the asymmetric noisy labels are generated by flipping *truck* → *automobile*, *bird* → *airplane*, *deer* → *horse* and *cat* ↔ *dog* for CIFAR-10. For CIFAR-100, the noise flips each class into the next, circularly within super-classes.

Data Preprocessing: We apply normalization and regular data augmentation (i.e. random crop and horizontal flip) on training sets. The crop size for CIFAR-10 and CIFAR-100 is 32.

Network, Optimizer and Parameter: We use ResNet34 [17] for both CIFAR-10 and CIFAR-100, and train them using SGD with a momentum of 0.9, a weight decay of 0.001, and a batch size of 128. The networks are trained for 300 epochs. We use the cosine annealing learning rate [30] where the maximum number of epoch for each period is 10, the maximum and minimum learning rate is set to 0.02 and 0.001 respectively. We warm up our model 10 epochs for CIFAR-10 and 30 epochs for CIFAR-100 with traditional cross entropy loss. For efficiency, we set $K = 1$ as default since we find that $K = 1, 2, 3$ achieves similar performance.

Baselines: We compare our method to the following baselines from different categories: (1) CE: directly uses the standard cross entropy loss to train the deep neural network on noisy

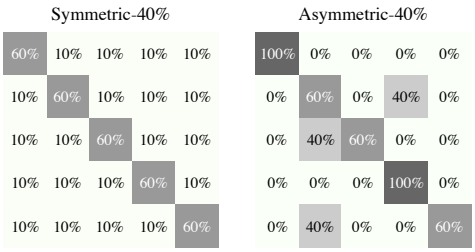


Fig. 5: Example of noise transition matrix Q (taking 5 classes and noise rate $\varepsilon = 0.4$ as an example).

training data. (2) F-correction [41] and Bootstrap [43] belong to loss correction category. (3) GCE [62], SCE [55], NFL+MAE [34], NCE+RCE[34] belong to robust loss function category. (4) Joint Optim [47], PENCIL [58], RoG+D2L [24], M-correction [1], SEAL [7] and LRT [63] belong to label correction category. (4) Decoupling [36], Co-teaching [15], MentorNet [20] and Iterative-CV [6] belong to sample selection with two networks category. (4) O2U-net [19], NLNL [22], DAC [48], Crust [39] and ODD [45] belong to noisy samples pruning category. (5) SELF [40] belongs to semi-supervised learning category.

Results on CIFAR-10 and CIFAR-100: Table II shows the classification test accuracies of our approach on CIFAR-10 and CIFAR-100 with different levels of symmetric and asymmetric label noise. As we can see, MixNN achieves the highest accuracy in most cases, especially in the challenging ones. For example, on CIFAR-10 with 80% symmetric label noise, MixNN outperforms the best state-of-the-art method (74.84% of DAC) by more than 11%. In the hardest case that CIFAR-100 with 80% symmetric label noise, we observe that most existing methods achieve relatively low test accuracies and PENCIL even fails to converge. However, MixNN still achieves the best accuracy up to 48.81%. Note that on CIFAR-10/CIFAR-100 with 20% symmetric label noise, NLNL and DAC obtain superior performance and even outperform our approach. The reason is that these methods perform multiple discrete training stages for different purposes. For example, NLNL performs three training stages including: a) Division of training data into either clean or noisy data with a DNN model. b) Training initialized DNN with clean data from a) and then updating noisy datas label following the output of DNN trained with clean data. c) Clean data and label-updated noisy data are both used for training initialized DNN in the final stage. In contrast, our approach conducts an end-to-end learning manner which is much simpler than NLNL. In summary, our approach shows a consistently strong performance across all datasets with different types and ratios of simulated label noise.

B. Effectiveness on Real-world Datasets with Noisy Labels

We use the Clothing1M dataset [57] and Webvision [27] to evaluate the performance of our approach in the real-world noisy labels settings.

- **Clothing1M**: The Clothing1M dataset contains 1 million images of clothing obtained from online shopping websites with 14 classes: T-shirt, Shirt, Knitwear, Chiffon, Sweater, Hoodie, Windbreaker, Jacket, Down Coat, Suit, Shawl, Dress, Vest, and Underwear. The labels are generated by using the surrounding texts of the images that are provided by the sellers, and thus contain many wrong labels. The overall accuracy of the labels is around 61.54%, with some pairs of classes frequently confused with each other (e.g. Knitwear and Sweater). The Clothing1M dataset also contains $50k$, $14k$, and $10k$ of clean data for training, validation, and testing, respectively. Note that we do not use the $50k$ clean training data. We report the classification accuracy on the test set when the performance on the validation set is optimal.
- **Webvision**: It is a large web images dataset that contains more than 2.4 millions of images crawled from the Flickr and Google Images search. The label noise level of Webvision is estimated at 20%. Following [6], we compare the baseline methods on the first 50 classes of Google image subset. We test the trained model on the human-annotated WebVision validation set and the ILSVRC12 validation set.

Data Preprocessing: We apply normalization and regular data augmentation (i.e. random crop and horizontal flip) on the training sets of the above datasets. Since real-world images are of different sizes, we perform the cropping consistent with the existing work [6]. Specifically, 224 $\times$ 224 for Clothing1M (after resizing to 256 $\times$ 256), and 227 $\times$ 227 for Webvision.
Network, Optimizer and Parameter: We use the ResNet-50 [17] pretrained on ImageNet for Clothing1M. We train the model with batch size 64. The optimization is done using SGD with a momentum of 0.9, and weight decay of 0.001. We use the same cosine annealing learning rate as CIFAR-10 except the minimum learning rate is set to 0.0001 and the total epoch is 200. For each epoch, we sample 1000 mini-batches from the training data ensuring that the classes of the noisy labels are balanced. For Webvision, we use an InceptionResNetV2 [46] as the backbone architecture. All other optimization details are the same as for CIFAR-10, except for the weight decay (0.0005), the batch size (32) and maximum learning rate (0.01). We also set $K = 1$ for both datasets.
Results on Clothing1M and Webvision: Table III shows the results on Clothing1M dataset. MixNN consistently outperforms other baselines, slightly superior to Joint-Optim. Table IV compares MixNN to state-of-the-art methods trained on the Webvision and evaluated on both the WebVision and ImageNet ILSVRC12 validation sets. MixNN produces superior results in terms of both top 1 and top 5 accuracy, which implies the proposed MixNN is reliable on datasets containing real-world noisy labels.

C. Efficiency of Noisy Label Correction

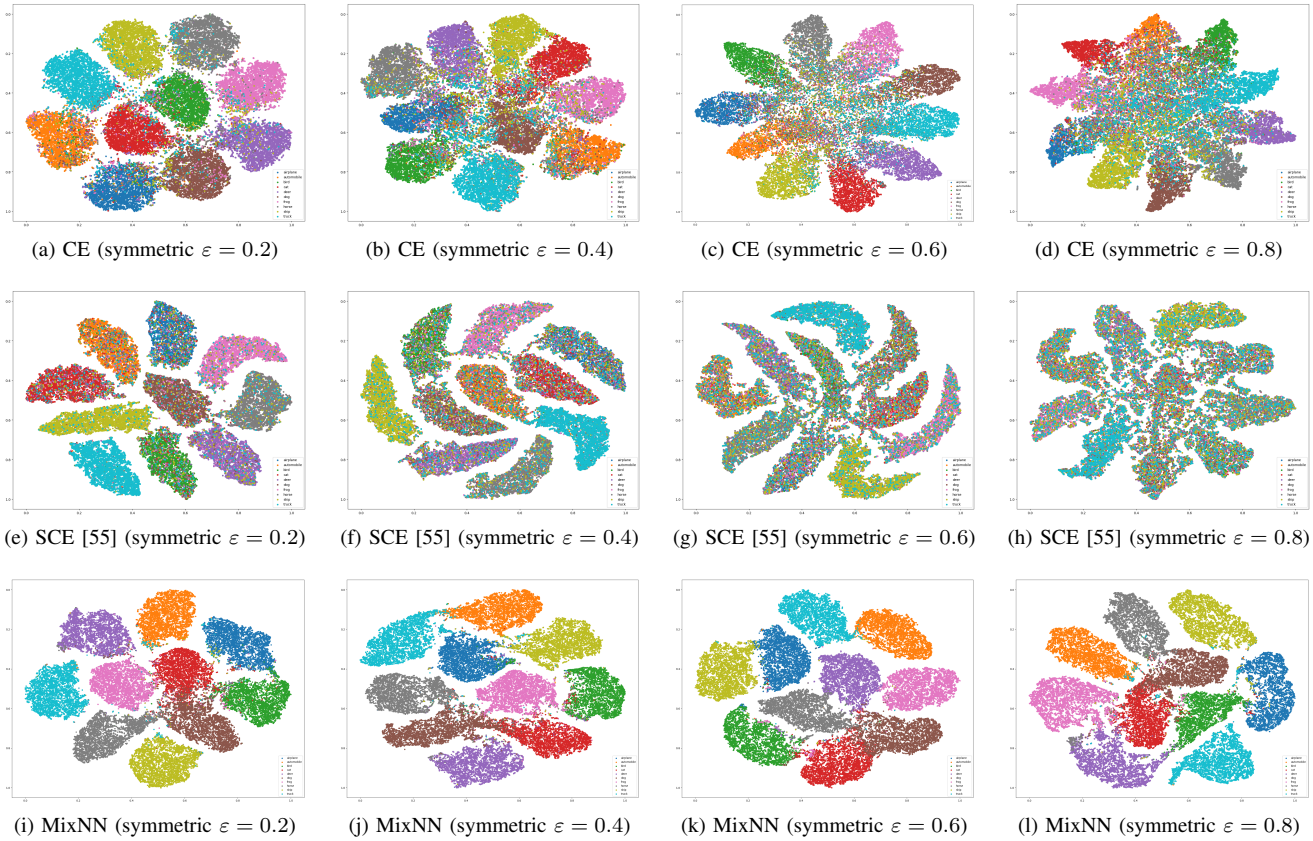
Recall that we perform noisy label correction in Section IV-E. Since the estimated target $\boldsymbol{t}_i$ is calculated by an exponential moving average between the given noisy labels and

TABLE II: Test Accuracy (%) on CIFAR-10 and CIFAR-100 with different ratios of symmetric and asymmetric label noise. We compare with previous works under the same backbone ResNet34 [17]. The average accuracy and standard deviation of 3 random runs are reported. **Bold** indicates the best results.

Model	Dataset	CIFAR-10					CIFAR-100				
	Noise type	symm				asymm	symm				asymm
	Method/Noise ratio	20%	40%	60%	80%	40%	20%	40%	60%	80%	40%
ResNet34	CE	86.98 ± 0.12	81.88 ± 0.29	74.14 ± 0.56	53.82 ± 1.04	80.11 ± 1.44	58.72 ± 0.26	48.20 ± 0.65	37.41 ± 0.94	18.10 ± 0.82	42.74 ± 0.61
	F-correction [41]	87.99 ± 0.36	83.25 ± 0.38	74.96 ± 0.65	54.64 ± 0.44	83.55 ± 0.58	39.19 ± 2.61	31.05 ± 1.44	19.12 ± 1.95	8.99 ± 0.58	34.44 ± 1.93
	Bootstrap [43]	86.23 ± 0.23	82.23 ± 0.37	75.12 ± 0.56	54.12 ± 1.32	81.21 ± 1.47	58.27 ± 0.21	47.66 ± 0.55	34.68 ± 1.10	21.64 ± 0.97	45.12 ± 0.57
	GCE [62]	89.83 ± 0.20	87.13 ± 0.22	82.54 ± 0.23	64.07 ± 1.38	76.74 ± 0.61	66.81 ± 0.42	61.77 ± 0.24	53.16 ± 0.78	29.16 ± 0.74	47.22 ± 1.15
	SCE [55]	89.83 ± 0.32	87.13 ± 0.26	82.81 ± 0.61	68.12 ± 0.81	82.51 ± 0.45	70.38 ± 0.13	62.27 ± 0.22	54.82 ± 0.57	25.91 ± 0.44	49.32 ± 0.87
	NFL+MAE [34]	-	83.81 ± 0.06	76.36 ± 0.31	45.23 ± 0.52	77.16 ± 0.10	-	58.18 ± 0.08	46.10 ± 0.50	24.78 ± 0.82	43.51 ± 0.42
	NCE+RCE [34]	-	86.02 ± 0.09	79.78 ± 0.50	52.71 ± 1.90	79.59 ± 0.40	-	59.48 ± 0.56	47.12 ± 0.62	25.80 ± 1.12	46.79 ± 0.96
	Joint Optim [47]	92.25	90.79	86.87	69.16	-	58.15	54.81	47.94	17.18	-
	PENCIL [58]	-	-	-	-	**91.01**	-	69.12 ± 0.62	57.79 ± 3.86	fail	63.61 ± 0.23
	RoG+D2L [24]	-	87.00	78.00	-	-	-	64.90	40.60	-	-
	M-correction [1]	-	92.30	86.10	74.10	-	-	70.10	59.50	39.50	-
	MentorNet [20]	92.00	91.20	74.20	60.00	-	73.50	68.50	61.20	35.50	-
	O2U-net [19]	-	90.30	-	43.40	-	-	69.20	-	39.40	-
	NLNL [22]	**94.23**	92.43	88.32	-	89.86	71.52	66.39	56.51	-	45.70
	DAC [48]	92.91	90.71	86.30	74.84	-	73.55	66.92	57.17	32.16	-
	SELF [40]	-	91.13	-	63.59	-	-	66.71	-	35.56	-
	MixNN (Ours)	93.27 ± 0.04	**92.89 ± 0.02**	**91.66 ± 0.07**	**86.08 ± 1.01**	**90.25 ± 0.76**	**73.90 ± 0.26**	**72.97 ± 0.14**	**67.56 ± 0.17**	**48.81 ± 0.06**	**68.18 ± 0.11**

TABLE III: Comparison with state-of-the-art methods trained on Clothing1M. Results of other methods are taken from original papers. All methods use a ResNet-50 architecture pretrained on ImageNet.

CE	F-correction [41]	GCE [62]	Co-teaching [15]	SEAL [7]	SCE [55]	LRT [63]	Joint-Optim [47]	MixNN
69.21	69.84	69.75	70.15	70.63	71.02	71.74	72.16	**72.39**



(a) CE (symmetric $\varepsilon = 0.2$) (b) CE (symmetric $\varepsilon = 0.4$) (c) CE (symmetric $\varepsilon = 0.6$) (d) CE (symmetric $\varepsilon = 0.8$)

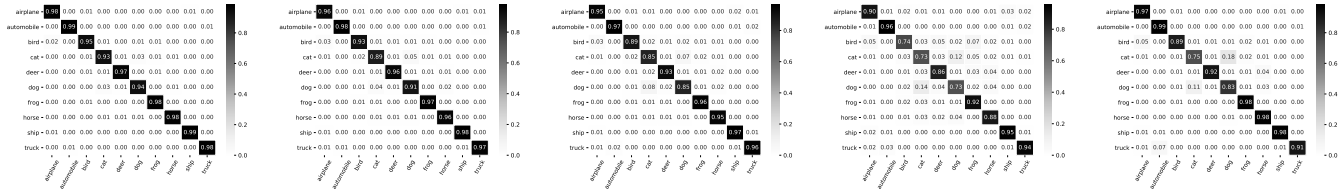
(e) SCE [55] (symmetric $\varepsilon = 0.2$) (f) SCE [55] (symmetric $\varepsilon = 0.4$) (g) SCE [55] (symmetric $\varepsilon = 0.6$) (h) SCE [55] (symmetric $\varepsilon = 0.8$)

(i) MixNN (symmetric $\varepsilon = 0.2$) (j) MixNN (symmetric $\varepsilon = 0.4$) (k) MixNN (symmetric $\varepsilon = 0.6$) (l) MixNN (symmetric $\varepsilon = 0.8$)

Fig. 6: t-SNE plots of feature representations learned by CE, SCE, and our proposed method MixNN on CIFAR-10 with different rates of label noise. Different colors represent the different classes in CIFAR-10.

TABLE IV: Comparison with state-of-the-art methods trained on (mini) WebVision. Numbers denote top-1 (top-5) accuracy (%) on the WebVision validation set and the ImageNet ILSVRC12 validation set. Results of other baseline methods are taken from original papers. All methods use an InceptionResNetV2 architecture.

		F-correction [41]	Decoupling [36]	D2L [35]	MentorNet [20]	Co-teaching [15]	Iterative-CV [6]	Crust [39]	ODD [45]	MixNN
WebVision	top1	61.12	62.54	62.68	63.00	63.58	65.24	72.40	74.60	**75.39**
	top5	82.68	84.74	84.00	81.40	85.20	85.34	89.56	90.60	**90.71**
ILSVRC12	top1	57.36	58.26	57.80	57.80	61.48	61.60	67.36	66.70	**70.54**
	top5	82.36	82.26	81.36	79.92	84.70	84.98	87.84	86.30	**90.31**



(a) 20% symmetric noise (b) 40% symmetric noise (c) 60% symmetric noise (d) 80% symmetric noise (e) 40% asymmetric noise

Fig. 7: Confusion matrix of corrected labels w.r.t clean labels on CIFAR-10 dataset with different label noise.

TABLE V: Correction accuracy (%) on CIFAR-10 and CIFAR-100 with various levels of label noise injected to training set.

Dataset	CIFAR-10					CIFAR-100				
Noise type	symm				asymm	symm				asymm
Noise ratio	20%	40%	60%	80%	40%	20%	40%	60%	80%	40%
Correction accuracy (%)	96.78	95.25	92.89	86.77	91.81	90.71	84.86	74.93	52.45	72.40

model predictions, our method is able to gradually correct the wrong labels. The correction accuracy can be calculated by $\frac{1}{N}\sum_i^N \mathbb{1}\{\text{argmax } \boldsymbol{y}_i = \text{argmax } \boldsymbol{t}_i\}$, where $\boldsymbol{y}_i$ is the ground truth label of training sample $\boldsymbol{x}_i$. We evaluate the correction accuracy on CIFAR-10 and CIFAR-100 with different levels of label noise. As we can see the results in Table V, our method successfully corrects a huge amount of wrong labels and obtains high correction accuracy in all cases. We also plot the confusion matrix of corrected labels w.r.t the clean labels on CIFAR-10 with different levels of label noise in Figure 7. Our approach corrects the wrong labels impressively well for all classes under different level of label noise. We also observe that class dog and class cat are the most similar classes in CIFAR-10, which greatly increases the difficulty in the label correction process.

D. Feature Representations

We further investigate the representations learned by our approach compared to that learned by traditional cross entropy loss and SCE [55]. We extract the high-dimensional representations at the penultimate layer and project them to a 2D embedding by using t-SNE [51]. The projected representations are illustrated in Fig. 6 for 20%, 40%, 60% and 80% symmetric label noise respectively. Under all settings, the feature representations learned by our approach are of significantly better quality than that of CE and SCE with more separated and clearly bounded clusters. We find that SCE always keeps the feature representations of mislabeled samples in their true classes, the same as what CE does in the early learning stage, which prevents the model from memorizing them. However, the boundary formed by the SCE becomes increasingly blurred as the noise ratio rises (see $\varepsilon = 0.8$ case). In contrast, our approach gradually corrects the noisy labels to clean labels, resulting in most of the feature representations in different classes are corrected.

VI. Conclusion

In this paper, we explore the representation distribution in the early learning phase and propose MixNN for learning with noisy labels. Our approach mitigates the negative influence of noisy labels by training with the new samples that are obtained by mixing the original training samples with the nearest neighbours. The mixing procedure is dynamically adjusted by the learned mixture model on sample loss distribution. We also propose a strategy that gradually corrects the noisy labels by using an exponential moving average on the given labels and model predictions. Through extensive experiments across multiple datasets with simulated and real-world label noise, we demonstrate that MixNN consistently exhibits substantial performance improvements compared to state-of-the-art methods. Importantly, the proposed approach works with any classifier out-of-the-box without any changes to architecture or training procedure. We are interested in adapting MixNN to other domains such as natural language process (NLP) and object detection, and believe MixNN is a promising algorithm for training robust DNNs against noisy labels. We hope that our work will trigger interest in the design of new approaches that provide robustness to label noise in big data applications.

REFERENCES

[1] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. Mcguinness. Unsupervised label noise modeling and loss correction. In *Proceedings of the 36th International Conference on Machine Learning*, pages 312–321, 2019.

[2] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org, 2017.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

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

[5] J. P. Brooks. Support vector machines with the ramp loss and the hard margin loss. *Operations research*, 59(2):467–479, 2011.

[6] P. Chen, B. B. Liao, G. Chen, and S. Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, pages 1062–1070, 2019.

[7] P. Chen, J. Ye, G. Chen, J. Zhao, and P.-A. Heng. Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11442–11450, 2021.

[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*, pages 248–255. Ieee, 2009.

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

[10] Y. Ding, L. Wang, D. Fan, and B. Gong. A semi-supervised two-stage approach to learning from noisy labels. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1215–1224. IEEE, 2018.

[11] A. Ghosh, H. Kumar, and P. Sastry. Robust loss functions under label noise for deep neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[12] A. Ghosh, N. Manwani, and P. Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

[13] J. Goldberger and E. Ben-Reuven. Training deep neural-networks using a noise adaptation layer. 2016.

[14] B. Han, G. Niu, J. Yao, X. Yu, M. Xu, I. Tsang, and M. Sugiyama. Pumpout: A meta approach for robustly training deep neural networks with noisy labels. 2018.

[15] 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*, pages 8527–8537, 2018.

[16] J. Han, P. Luo, and X. Wang. Deep self-learning from noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5138–5147, 2019.

[17] 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*, pages 770–778, 2016.

[18] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*, 2018.

[19] J. Huang, L. Qu, R. Jia, and B. Zhao. O2u-net: A simple noisy label detection approach for deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3326–3334, 2019.

[20] 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.

[21] A. Joulin, L. Van Der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision*, pages 67–84. Springer, 2016.

[22] Y. Kim, J. Yim, J. Yun, and J. Kim. Nlnl: Negative learning for noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 101–110, 2019.

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

[24] K. Lee, S. Yun, K. Lee, H. Lee, B. Li, and J. Shin. Robust inference via generative classifiers for handling noisy labels. In *International Conference on Machine Learning*, pages 3763–3772. PMLR, 2019.

[25] J. Li, R. Socher, and S. C. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.

[26] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli. Learning to learn from noisy labeled data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5051–5059, 2019.

[27] W. Li, L. Wang, W. Li, E. Agustsson, and L. V. Gool. Webvision database: Visual learning and understanding from web data. *CoRR*, 2017.

[28] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems*, 33, 2020.

[29] T. Liu and D. Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.

[30] I. Loshchilov and F. Hutter. Sgdr: Stochastic gra-

dient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[31] Y. Lu, Y. Bo, and W. He. Co-matching: Combating noisy labels by augmentation anchoring. *arXiv preprint arXiv:2103.12814*, 2021.

[32] Y. Lu, Y. Bo, and W. He. Confidence adaptive regularization for deep learning with noisy labels. *arXiv preprint arXiv:2108.08212*, 2021.

[33] M. Lukasik, S. Bhojanapalli, A. Menon, and S. Kumar. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, pages 6448–6458. PMLR, 2020.

[34] X. Ma, H. Huang, Y. Wang, S. Romano, S. Erfani, and J. Bailey. Normalized loss functions for deep learning with noisy labels. In *International Conference on Machine Learning*, pages 6543–6553. PMLR, 2020.

[35] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey. Dimensionality-driven learning with noisy labels. *arXiv preprint arXiv:1806.02612*, 2018.

[36] E. Malach and S. Shalev-Shwartz. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pages 960–970, 2017.

[37] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

[38] H. Masnadi-Shirazi and N. Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in neural information processing systems*, pages 1049–1056, 2009.

[39] B. Mirzasoleiman, K. Cao, and J. Leskovec. Coresets for robust training of neural networks against noisy labels. *Neural Information Processing Systems (NeurIPS)*, 2020.

[40] T. Nguyen, C. Mummadi, T. Ngo, L. Beggel, and T. Brox. Self: learning to filter noisy labels with self-ensembling. In *International Conference on Learning Representations (ICLR)*, 2020.

[41] 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*, pages 1944–1952, 2017.

[42] H. Permuter, J. Francos, and I. Jermyn. A study of gaussian mixture models of color and texture features for image classification and segmentation. *Pattern recognition*, 39(4):695–706, 2006.

[43] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[44] M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343. PMLR, 2018.

[45] J. Song, Y. Dauphin, M. Auli, and T. Ma. Robust and on-the-fly dataset denoising for image classification. In *European Conference on Computer Vision*, pages 556–572. Springer, 2020.

[46] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[47] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

[48] S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, and J. Mohd-Yusof. Combating label noise in deep learning using abstention. In *International Conference on Machine Learning*, pages 6234–6243. PMLR, 2019.

[49] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[50] A. Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605, 2017.

[51] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[52] B. Van Rooyen, A. Menon, and R. C. Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems*, pages 10–18, 2015.

[53] X. Wang, Y. Hua, E. Kodirov, and N. M. Robertson. Imae for noise-robust learning: Mean absolute error does not treat examples equally and gradient magnitude's variance matters. *arXiv preprint arXiv:1903.12141*, 2019.

[54] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia. Iterative learning with open-set noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8688–8696, 2018.

[55] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey. Symmetric cross entropy for robust learning with noisy labels. *arXiv preprint arXiv:1908.06112*, 2019.

[56] H. Wei, L. Feng, X. Chen, and B. An. Combating noisy labels by agreement: A joint training method with co-regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13726–13735, 2020.

[57] 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*, pages 2691–2699, 2015.

[58] K. Yi and J. Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7017–7025, 2019.

[59] X. Yu, B. Han, J. Yao, G. Niu, I. W. Tsang, and M. Sugiyama. How does disagreement help generalization against label corruption? *arXiv preprint arXiv:1901.04215*, 2019.

[60] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization, 2018.

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

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

[63] S. Zheng, P. Wu, A. Goswami, M. Goswami, D. Metaxas, and C. Chen. Error-bounded correction of noisy labels. In *International Conference on Machine Learning*, pages 11447–11457. PMLR, 2020.

[64] X. Zhu and X. Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210, 2004.