DualGraph: A graph-based method for reasoning about label noise

HaiYang Zhang, XiMing Xing, Liang Liu
Beijing University of Posts and Telecommunications
School of Computer Science
{zhhy, ximingxing, liangliu}@bupt.edu.cn

Abstract

Unreliable labels derived from large-scale dataset prevent neural networks from fully exploring the data. Existing methods of learning with noisy labels primarily take noise-cleaning-based and sample-selection-based methods. However, for numerous studies on account of the above two views, selected samples cannot take full advantage of all data points and cannot represent actual distribution of categories, in particular if label annotation is corrupted. In this paper, we start from a different perspective and propose a robust learning algorithm called DualGraph, which aims to capture structural relations among labels at two different levels with graph neural networks including instance-level and distribution-level relations. Specifically, the instance-level relation utilizes instance similarity characterize sample category, while the distribution-level relation describes instance similarity distribution from each sample to all other samples. Since the distribution-level relation is robust to label noise, our network propagates it as supervised signals to refine instance-level similarity. Combining two level relations, we design an end-to-end training paradigm to counteract noisy labels while generating reliable predictions. We conduct extensive experiments on the noisy CIFAR-10 *dataset,* CIFAR-100 *dataset, and the* Clothing1M *dataset. The results demonstrate the advantageous performance of the proposed method in comparison to state-of-the-art baselines.*

1. Introduction

Deep learning has turned out to be excellent performance at discovering intricate structures in high-dimensional data [14], particularly deep convolutional nets have brought about breakthroughs in processing image classification [8], semantic segmentation [16], and object detection [23]. Most of these tasks require reliable and clean large-scale datasets to train Deep Neural Networks (DNNs), but it is time-consuming and expensive to collect such high-quality datasets like ImageNet [3]. To alleviate this problem, alternatives such as crowdsourcing [34, 37] and web-crawlers [5] are available to improve annotation efficiency. However, those low-cost approaches introduce low-quality annotations, and these labels are unreliable due to various types of noise. Meanwhile, DNNs easily fit a random labeling of the training data [38]. As noisy labels severely degrade the generalization performance [2, 30] of DNNs, learning from noisy labels has become a significant task.

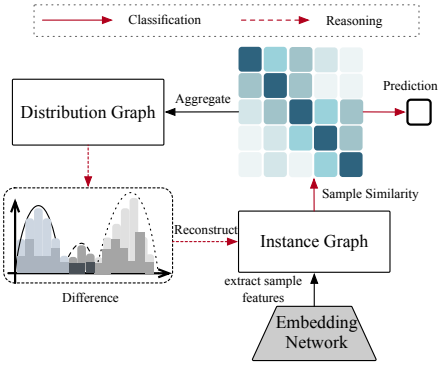

Figure 1. *The concept of iterate optimization*. Distribution graph forward propagation instance similarity distribution difference to refine structural relationships among labels. In the classification phase, we utilize the reconstructed instance graph to generate reliable predictions. The two training processes are executed alternatively.

Existing methods of learning with noisy labels primarily take noise-cleaning-based and sample-selection-based methods [1, 27]. *Noise-cleaning-based* methods mainly remove samples with suspicious labels or correct their noisy labels to corresponding true class [6, 29, 35]. [6] obtains correct label by building the prototype and comparing its similarity with the training data. However, the proposed heuristics algorithms [29, 35] have been criticized for removing too many instances or keeping mislabeled instances. *Sample-selection-based* methods aim to identify true-labeled samples from noisy training data [7, 17, 36]. Co-teaching [7] and Co-teaching+ [36] train models on small-loss instances. Decoupling [17] and Co-teaching+ [36] introduce the "Disagreement" strategy, where "when to update" depends on a disagreement between two dif-

ferent networks. However, there are only a part of training examples that can be selected by the "Disagreement" strategy, and these examples cannot be guaranteed to have ground-truth labels. As described, studies based on these two ideas generally choose trusted examples to avoid the phenomenon of over-fitting. Also, these methods are limited by memorization effects [2], and examples with noisy labels are among the most forgotten examples [30]. Furthermore, previous methods lack of a global perspective to explore patterns in the relationship between samples, which naturally motivates us to improve them in our research.

In this paper, we start from a different perspective and propose a robust learning algorithm called DualGraph, which aims to capture structural relationships among labels at two different levels. Relevant references reveal that both noise and hard examples are the causes of classifier forgetting [2, 30, 38], but the methods of probability selection and small loss selection tend to confuse noise and hard examples. To this end, we design an end-to-end training paradigm called *iterate optimization mechanism* as Figure 1 illustrates. It consists of two alternate phases, *i.e.*, reasoning and classification. In the reasoning phase, we generate the similarity distribution for each sample by calculating the similarity from one sample to all other samples. In the classification phase, the distribution features are applied to reconstruct the instance graph nodes and generate reliable predictions via calculating node similarity. Such a cyclic operation is executed several times until convergence. Specifically, we train two graph neural networks with a joint loss, including the example classifier loss and the distribution loss. Furthermore, we utilize the joint loss to weighted edge loss to enhance positive examples (clean examples) and weaken negative examples (noise and hard examples) in instance graph.

In summary, the main contributions of this paper are:

- To the best of our knowledge, we are the first to exploit the graph neural network that captures structural relations among labels at two different levels. Our approach refines instance-level relations via instance similarity distribution obtained from distribution-level relations to establish the robust label relations.
- We propose an iterate optimization mechanism that contains two phases to train two graph neural networks simultaneously. In the reasoning phase, our approach obtains the distribution feature for each sample and propagates distribution information in a graph neural network to correct corrupted labels. In the classification phase, our approach utilizes the reconstructed instance-level graph to generate reliable predictions.
- We experimentally show that our approach significantly advances state-of-the-art results on multiple benchmarks with different types and levels of label noise. Especially on the Clothing1M dataset, our approach outperforms existing methods by $6\% \sim 8\%$.

2. Related work

A comprehensive label noise overview is given by [1, 27]. In this section, we briefly review existing studies on learning with noisy labeled datasets.

2.1. Learning with Noisy Labels

Learning with noisy labels is a longstanding problem and has been studied extensively. The early methods focus on estimating the label transition matrix [18, 19, 21, 31]. For example, F-correction [21] uses a two-step solution to heuristically estimate the noise transition matrix. An additional softmax layer is introduced to model the noise transition matrix [4].

To avoid any false corrections, many recent studies [7, 11, 17, 36] have adopted sample selection that involves selecting true-labeled samples from a noisy training dataset [27]. Recently, a promising method of handling noisy labels is to train models on small-loss instances [7, 36]. In particular, the widely used *small-loss criterion* is based on a concept, that DNNs tend to learn simple patterns first, then gradually memorize all samples [2]. For instance, Co-teaching [7] and Co-teaching+ [36] maintain two DNNs, but each DNN selects a certain number of small-loss examples and feeds them to its peer DNN for further training. Compared with Co-teaching[7], Co-teaching+[36] further employs the disagreement strategy of decoupling [17]. Intuitively, different classifiers can generate different decision boundaries and then have different abilities to learn. Thus, when training on noisy labels, the authors also expect that their methods have different abilities to filter out the label noise. However, there are only a part of training examples that can be selected by the disagreement strategy, and these examples cannot be guaranteed to have ground-truth labels.

A simple methodology to deal with noisy labels is to correct their noisy label [6, 29, 35] to corresponding true class. This can be done in preprocessing stage of the training data, however such methods usually tackle the difficulty of distinguishing informative hard samples from those with noisy labels [1]. For instance, [6] constructs prototypes that are able to represent deep feature distribution of the corresponding class. Then corrected label is found by checking similarity among the data sample and prototypes. Joint optimization framework for both training classifier and propagating noisy labels to cleaner labels is presented in [29]. PENCIL [35] adopts label probability distributions to supervise network learning and to update these distributions through back-propagation end-to-end in each epoch.

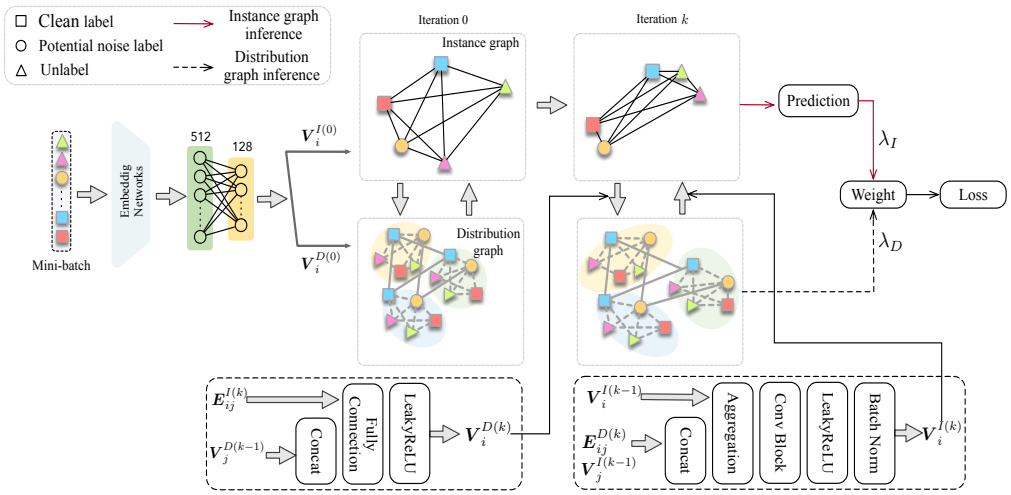

Figure 2. *DualGraph architecture* and the iteration from the 0 to the k forward propagation. The features from the embedding network are compressed into m-dimensional tensors and fed into two graph modules to initialize the instance and distribution graph.

2.2. Graph Neural Network

Globally modeling and reasoning over relations between regions can be beneficial for many computer vision tasks on both images and videos. Graph neural networks are proposed to collectively aggregate information from graph structure [41, 25], thus they can model input and/or output consisting of elements and their dependency. Recent approaches [10, 39, 40] are proposed to exploit graph neural network in the field of label noise task. [40] formulates video anomaly detection as a classification with label noise problem and trains a Graph Convolutional Neural (GCN) label noise cleaning network depending on features and temporal consistency of video snippets. [10] learns a classifier on a large-scale weakly-labeled collection jointly with only a few clean labeled examples and applies GCN to clean noisy data. FaceGraph [39] uses two cascaded GCNs to select useful data in the large-scale web-collected face datasets. Unfortunately, most methods limit to specific domain, or depend on a clean subset.

3. Proposed Approach

In this section, we first provide the background of label noise task, then introduce the proposed approach in detail.

3.1. Background and Problem Definition

Given the training set $\mathcal{D}_{\text{train}}$, the goal is to learn the model $\boldsymbol{f} : \boldsymbol{x} \rightarrow \boldsymbol{y}$, which is capable of generalizing well to the unseen test set $\mathcal{D}_{\text{test}}$, where $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$. In particular, we consider a classification problem with a training set $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, where $\boldsymbol{x}_i$ denotes the i^{th} sample and $\boldsymbol{y}_i \in \{0, 1\}^c$ is a one-hot vector representing the corresponding noisy label over c classes. Let $\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta})$ denotes the discriminative function of a neural network parameterized by $\boldsymbol{\theta}$, which maps an input to an output of the c-class softmax layer. The conventional objective for supervised classification is to minimize an empirical risk, such as the cross entropy loss:

$$\mathcal{L}_c = -\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}_i \cdot \log\left(f\left(\boldsymbol{x}_i, \boldsymbol{\theta}\right)\right) \tag{1}$$

where $\cdot$ denotes dot product.

As data labels are corrupted in various real-world scenarios, that $\boldsymbol{y}_i$ contains noise, the neural network can overfit and perform poorly on the test set [38]. Formally, at each training step, we consider a mini-batch of data $(\boldsymbol{X}, \boldsymbol{Y})$ sampled from the $\mathcal{D}_{\text{train}}$, where $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_p\}$ is set of p samples, and $\boldsymbol{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_p\}$ is the corresponding labels. We sampling generate multiple mini-batches of potentially noisy labels $\{\hat{\boldsymbol{Y}}_1, ..., \hat{\boldsymbol{Y}}_L\}$ with similar label distribution as $\boldsymbol{Y}$. We will describe the procedure for generating one set of labels $\hat{\boldsymbol{Y}}_l = \{\hat{\boldsymbol{y}}_1^l, ..., \hat{\boldsymbol{y}}_p^l\}$. In standard training, we aim to minimize the expected loss for the $\boldsymbol{X}$, where each input example is weighted equally. Here we aim to learn a reweighting of the inputs, where we minimize a weighted loss:

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{p} w_i \boldsymbol{f}_i(\theta) \tag{2}$$

where w_i denotes weight of example $\boldsymbol{x}_i$, the initialization details for w_i will be covered in section 3.4. In this paper, a novel weighting method is proposed to express weight of the sample via the edge of the graph neural network and we introduce it into the loss function. The details will be described in the next section.

3.2. Overview of DualGraph

This section describes DualGraph for learning with noisy labels. First of all, the train set $\mathcal{D}_{\text{train}}$ and validation set $\mathcal{D}_{\text{val}}$ in mini-batch can form an undirected acyclic graph structure $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ where $\boldsymbol{\mathcal{V}} := \{\boldsymbol{V}_i\}_{i=1,...,|p|}$ and $\boldsymbol{\mathcal{E}} :=$

$\{\boldsymbol{E}_{ij}\}_{i,j=1,\ldots,|p|}$ denote the set of nodes and edges of the graph, respectively. Note here that both edges and nodes are tensor representations. Even though noisy labels are explicitly injected, we remedy noise via distribution graph presented in the next section. Furthermore, as shown in Figure 2, the DualGraph consists of several iterations and each iteration consists of an *Instance graph* $\boldsymbol{\mathcal{G}}^{I(k)} = (\boldsymbol{\mathcal{V}}^{I(k)}, \boldsymbol{\mathcal{E}}^{I(k)})$ and a *Distribution graph* $\boldsymbol{\mathcal{G}}^{D(k)} = (\boldsymbol{\mathcal{V}}^{D(k)}, \boldsymbol{\mathcal{E}}^{D(k)})$, where k denotes the k-th iteration. Initially, the feature embeddings of all samples are extracted via a convolutional neural network, these embeddings are used to compute instance similarities $\boldsymbol{E}^{I(k)}$. Secondly, the instance similarities $\boldsymbol{E}^{I(k)}$ are delivered to build $\boldsymbol{\mathcal{G}}^{D(k)}$. Initialize node feature $\boldsymbol{V}^{D(k)}$ by aggregating $\boldsymbol{E}^{I(k)}$ following the position order $\boldsymbol{\mathcal{G}}^{I(k)}$, and edge feature $\boldsymbol{E}^{D(k)}$ represents the distribution similarity between node feature $\boldsymbol{V}^{D(k)}$. Finally, the obtained $\boldsymbol{E}^{D(k)}$ is feed to $\boldsymbol{\mathcal{G}}^{I(k)}$ to construct a more discriminative representation of the node, and then the above process is repeated iteration by iteration.

3.3. Instance Graph

First of all, a mini-batch of data $(\boldsymbol{X}, \boldsymbol{Y})$ sampled from $\boldsymbol{\mathcal{D}}_{\text{train}}$ can form an undirected acyclic graph structure, named *Instance Graph*. Specifically, initialize $\boldsymbol{V}_i^{I(0)}$ by the output of the embedding network, for each sample $\boldsymbol{x}_i$:

$$\boldsymbol{V}_i^{I(0)} = \boldsymbol{f}_{emb}(\boldsymbol{x}_i; \theta_{emb}) \tag{3}$$

where $\boldsymbol{f}_{emb}$ denotes the embedding network with the parameter set θ_{emb} and $\boldsymbol{V}_i^{I(0)} \in \mathbb{R}^m$, m denotes the dimension of embedded features. Specifically, parameter set θ_{emb} of $\boldsymbol{f}_{emb}$ will be jointly optimized in iterate optimization mechanism.

Each edge in the instance graph stands for the instance similarity. For generation $k \geq 0$, given $\boldsymbol{V}_i^{I(k-1)}$, $\boldsymbol{V}_j^{I(k-1)}$, $\boldsymbol{E}_{ij}^{I(k-1)}$ and the edge $\boldsymbol{E}_{ij}^{I(k)}$ can be updated as equation (4) and (5).

$$\boldsymbol{E}_{ij}^{I(k)} = \begin{cases} \boldsymbol{f}_{E^{I(0)}}(\boldsymbol{dis}^{I(0)}) & \text{if } k = \mathbf{0} \\ \boldsymbol{f}_{E^{I(k)}}(\boldsymbol{dis}^{I(k)}) \cdot \boldsymbol{E}_{ij}^{I(k-1)} & \text{if } k > \mathbf{0} \end{cases} \tag{4}$$

$$\boldsymbol{dis}^{I(k)} = \frac{(\boldsymbol{V}_i^{I(k)} - \boldsymbol{V}_j^{I(k)})^2}{\max\left(\|(\boldsymbol{V}_i^{I(k)} - \boldsymbol{V}_j^{I(k)})^2\|_2, \epsilon\right)} \tag{5}$$

where $\boldsymbol{dis}^{I(k)}$ measures the (dis)similarity between pair of nodes i-th with j-th, and $\boldsymbol{f}_{E^I} : \mathbb{R}^m \to \mathbb{R}$ is a neural network that transforms instance similarity to a certain scale, further $\boldsymbol{E}_{i,j}^{I(k)} \in \mathbb{R}$. Specifically, $\boldsymbol{f}_{E^I}$ is a transformation network consisting of two Conv-ReLU-BN [9, 33] blocks, one sigmoid activation, and one dropout [28] layer with the parameter set $\theta_E^{I(k)}$.

The distribution graph $\boldsymbol{\mathcal{G}}^{D(k)}$ is generated and updated, after edge feature $\boldsymbol{E}^{I(k)}$ in the instance graph are generated and updated.

3.4. Distribution Graph

As shown in Figure 2, the distribution graph aims at integrating instance-level relations from the instance graph to generate distribution features and reweights the samples according to the distribution relation, trying to eliminate abnormal noise points from the perspective of distribution. Each distribution feature $\boldsymbol{V}_{ij}^{D(k)}$ in $\boldsymbol{\mathcal{G}}^D$ is a p dimension vector, where the value in j-th entry represents the relation between sample x_i and sample x_j. For iteration $k = 0$:

$$\boldsymbol{V}_i^{D(0)} = \begin{cases} \mathbf{1} & \text{if } y_i = y_j \\ \mathbf{0} & \text{if } y_i \neq y_j \\ \frac{\mathbf{1}}{\boldsymbol{p}} & \text{if } y_i \text{ is unlabeled} \end{cases} \tag{6}$$

where $\boldsymbol{V}_i^{D(0)} \in \mathbb{R}^p$, p stands for the number of training examples in a mini-batch.

Algorithm 1: DualGraph

Input: $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}}; \boldsymbol{\mathcal{D}}_{\text{train}})$, where $\boldsymbol{\mathcal{D}}_{\text{train}} = (\boldsymbol{X}, \boldsymbol{Y})$
Output: $\{\widetilde{\boldsymbol{y}}_i\}_{i=1}^s$
Parameters: $\theta_{emb} \cup \{\theta_E^I, \theta_V^I, \theta_E^D, \theta_V^D\}$
Initialize: $\boldsymbol{V}_i^{I(0)}, \boldsymbol{V}_i^{D(0)}$ by Equation (3) and (6)
for $k = 0, \cdots, K$ **do**
  `// instance similarity update`
  for $i = 1, \cdots, |\boldsymbol{E}^I|$ **do**
    $\boldsymbol{E}_{ij}^{I(k)} \leftarrow \left(\text{Equation (4)}; \theta_E^{I(k)}\right)$
  end
  `// distribution feature generation`
  for $i = 1, \cdots, |\boldsymbol{V}^D|$ **do**
    $\boldsymbol{V}_i^{D(k)} \leftarrow \left(\text{Equation (7)}; \theta_V^{D(k)}\right)$
  end
  `// distribution similarity update`
  for $i = 1, \cdots, |\boldsymbol{E}^D|$ **do**
    $\boldsymbol{E}_{ij}^{D(k)} \leftarrow \left(\text{Equation (8)}; \theta_E^{D(k)}\right)$
  end
  `// rebuild instance graph`
  for $i = 1, \cdots, |\boldsymbol{V}^I|$ **do**
    $\boldsymbol{V}_i^{I(k)} \leftarrow \left(\text{Equation (10)}; \theta_V^{I(k)}\right)$
  end
end
`// prediction`
$\{\widetilde{\boldsymbol{y}}_i\}_{i=1}^p \leftarrow \text{Predict } \left(\{\hat{\boldsymbol{y}}_i\}_{i=1}^p\right)$

For iterations $k > 0$, the distribution graph node $\boldsymbol{V}_i^{D(k)}$ can be updated as follows:

$$\boldsymbol{V}_i^{D(k)} = \boldsymbol{f}_{V^D}\left(\left[\sum_{j=1}^p \boldsymbol{E}_{ij}^{I(k)} \| \sum_{j=1}^p \boldsymbol{V}_j^{D(k-1)}\right]; \theta_V^{D(k)}\right) \tag{7}$$

where $\boldsymbol{f}_{V^D} : (\mathbb{R}^p, \mathbb{R}^p) \to \mathbb{R}^p$ is the aggregation network

that composed of a fully-connected and leaky-ReLU [33] layer with the parameter set $\theta_V^{D(k)}$.

Each edge in distribution graph represents the similarity between distribution features of different examples. For generation $k \geq \mathbf{0}$, given $\boldsymbol{V}_i^{D(k)}$, $\boldsymbol{V}_j^{D(k)}$, $\boldsymbol{E}_{ij}^{D(k-1)}$ and the distribution similarity $\boldsymbol{E}_{ij}^{D(k)}$ can be updated as Equation (8).

$$\boldsymbol{E}_{ij}^{D(k)} = \begin{cases} \boldsymbol{f}_{E^{D(0)}}(\boldsymbol{dis}^{D(0)}) & \text{if } k = \mathbf{0} \\ \boldsymbol{f}_{E^{D(k)}}(\boldsymbol{dis}^{D(k)}) \cdot \boldsymbol{E}_{ij}^{D(k-1)} & \text{if } k > \mathbf{0} \end{cases} \tag{8}$$

$$\boldsymbol{dis}^{D(k)} = \left(\boldsymbol{V}_i^{D(k)} - \boldsymbol{V}_j^{D(k)}\right)^2 \tag{9}$$

where $\boldsymbol{f}_{E^{D(0)}} : \mathbb{R}^p \to \mathbb{R}$ is a transformation operation consisting of two Conv-ReLU-BN [9, 33] blocks, one sigmoid activation, and one dropout [28] layer with the parameter set $\theta_E^{D(k)}$.

The information in distribution graph is applied to reconstruct the instance graph at the end of each iteration. Then node features $\boldsymbol{V}_i^{I(k)}$ in the instance graph captures the distribution relations through aggregating all the node features with $\boldsymbol{E}_{ij}^{D(k)}$ as follows:

$$\boldsymbol{V}_i^{I(k)} = \boldsymbol{f}_{V^I}\left(\sum_{j=1}^{p}(\boldsymbol{E}_{ij}^{D(k)} \| \boldsymbol{V}_j^{I(k-1)}), \boldsymbol{V}_i^{I(k-1)}; \theta_V^{I(k)}\right) \tag{10}$$

where $\boldsymbol{f}_{V^I} : (\mathbb{R}^m, \mathbb{R}^m) \to \mathbb{R}^m$ is the aggregation network for instance graph with the parameter set $\theta_V^{I(k)}$ and $\boldsymbol{V}_i^{I(k)} \in \mathbb{R}^m$. Such a cyclic operation is executed several times until convergence.

3.5. Loss Function

The class prediction of each node can be computed by feeding the corresponding edges in the final generation of DualGraph into softmax function:

$$\boldsymbol{P}\left(\widetilde{\boldsymbol{y}}_i \mid \boldsymbol{x}_i\right) = \text{Softmax}\left(\sum_{j=1}^{p} \boldsymbol{E}_{ij}^{I(K)} \cdot \text{ one-hot } \left(\hat{\boldsymbol{y}}_j\right)\right) \tag{11}$$

where $\boldsymbol{P}\left(\widetilde{\boldsymbol{y}}_i \mid \boldsymbol{x}_i\right)$ is the probability distribution over classes given example $\boldsymbol{x}_i$, $\hat{\boldsymbol{y}}_j$ is the label of j-th example and $\widetilde{\boldsymbol{y}}_j$ is the label for network prediction. $\boldsymbol{E}_{ij}^{I(k)}$ stands for the instance similarity in the instance graph at the final iteration. Specifically, $\boldsymbol{E}_{ij}^{I(K)}$ is instance similarity reweighted by $\boldsymbol{\mathcal{G}}^{D(K)}$, and K denotes total iteration of the DualGraph.

It is noted that we make classification predictions in the instance graph for each sample. Therefore, the loss of the instance graph at iteration k-th is defined as follows:

$$\mathcal{L}_k^I = \mathcal{L}_{CE}\left(\boldsymbol{P}\left(\widetilde{\boldsymbol{y}}_i \mid \boldsymbol{x}_i\right), \boldsymbol{y}_i\right) \tag{12}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss function. $\boldsymbol{P}\left(\widetilde{\boldsymbol{y}}_i \mid \boldsymbol{x}_i\right)$ and $\boldsymbol{y}_i$ are model probability predictions of sample $\boldsymbol{x}_i$ and the ground-truth label respectively.

To learn discriminative distribution features, we incorporate the distribution loss which plays a significant role in promoting better discrimination of noise information. We define the distribution loss for iteration k-th as follows:

$$\mathcal{L}_k^D = \mathcal{L}_{CE}\left(\text{Softmax}\left(\sum_{j=1}^{p} \boldsymbol{E}_{ij}^{D(k)} \cdot \text{one-hot}\left(\boldsymbol{y}_j\right)\right), \boldsymbol{y}_i\right) \tag{13}$$

where $\boldsymbol{E}_{ij}^{D(k)}$ stands for the distribution similarity in the distribution graph at the k-th iteration.

The total objective function is a weighted summation of all the losses mentioned above:

$$\mathcal{L} = \sum_{k=1}^{K}\left(\lambda_I \mathcal{L}_k^I + \lambda_D \mathcal{L}_k^D\right) \tag{14}$$

in which λ_I and λ_D are two hyperparameters.

4. Experiments

In this section we first compare DualGraph with some state-of-the-art approaches, then analyze the impact of Graph Module and Iterate Optimization Mechanism by ablation study.

4.1. Experiment setup

Datasets. We verify the effectiveness of our proposed algorithm on three benchmark datasets: *CIFAR-10*, *CIFAR-100* [13] and *Clothing1M* [32], and the detailed characteristics of these datasets can be found in supplementary materials. These datasets are popularly used for the evaluation of learning with noisy labels in previous literatures [7, 11, 29, 36, 35]. Especially, *Clothing1M* is a large-scale real-world dataset with noisy labels, which is widely used in the related works [6, 29, 35]. The *Clothing1M* dataset contains **1** million images of clothing obtained from several online shopping websites that are classified into the following 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 surrounding texts of the images that are provided by the sellers, and therefore contain many errors [32].

Since all datasets are clean except *Clothing1M*, following [21, 22], we need to corrupt these datasets manually by the label transition matrix $\boldsymbol{Q}$, where $\boldsymbol{Q}_{ij} = \Pr[\hat{\boldsymbol{y}} = j | \boldsymbol{y} = i]$ given that noisy $\hat{\boldsymbol{y}}$ is flipped from clean $\boldsymbol{y}$. Assume that the matrix $\boldsymbol{Q}$ has two representative structures: (1) Symmetry flipping [24], randomly replacing the labels for a percentage of training data with all possible labels; (2) Asymmetry flipping [21], simulation of fine-grained classification with noisy labels, where labellers may make mistakes only within very similar classes. And the detailed characteristics of $\boldsymbol{Q}$ can be found in supplementary materials.

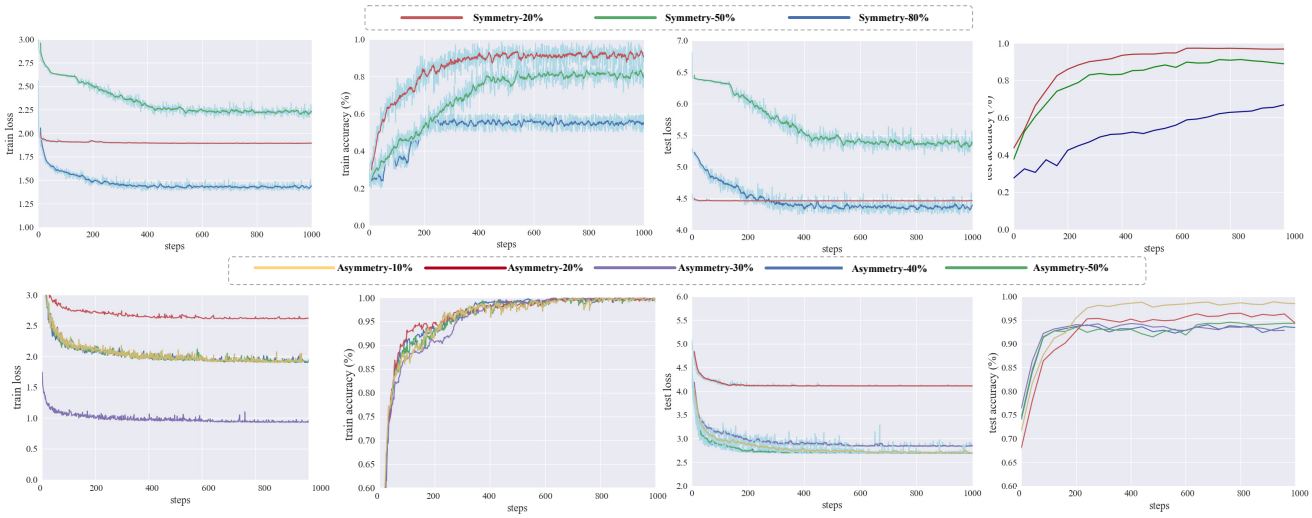

Figure 3. **Top:** Results on CIFAR-10 dataset with Symmetric Flipping; **Bottom:** Results on CIFAR-10 dataset with Asymmetric Flipping. From left to right: train loss vs. steps, train accuracy(%) vs. steps, test loss vs. steps, test accuracy(%) vs. steps. (1 step here represents 100 steps in the experiment).

Table 1. Average test accuracy (%) on CIFAR-10.

Flipping-Rate	F-correction	Co-teaching	Co-teaching+	Joint-optim	PENCIL	DualGraph
Symmetry-20%	83.40 ± 0.20	85.23 ± 0.27	89.49 ± 0.34	91.90 ± 0.20	92.64 ± 0.14	**96.7** ± 0.53
Symmetry-50%	79.18 ± 0.60	76.30 ± 0.13	85.68 ± 0.54	89.60 ± 0.30	90.36 ± 0.23	**92.2** ± 1.82
Symmetry-80%	63.30 ± 0.42	48.58 ± 2.22	67.37 ± 2.74	73.59 ± 0.64	76.18 ± 1.33	**77.2** ± 2.93
Asymmetry-40%	75.71 ± 0.40	73.63 ± 0.35	68.84 ± 0.20	88.89 ± 0.35	91.01 ± 0.20	**94.1** ± 1.41

Table 2. More details in test accuracy (%) on CIFAR-10 with Asymmetric Flipping.

Flipping-Rate	10%	20%	30%	40%	50%
F-correction	92.4	91.4	91.0	90.3	83.8
Joint-optim	92.5	91.9	91.1	91.5	75.8
PENCIL	93.0	92.4	91.8	91.1	80.5
DualGraph	**97.1**	**96.8**	**94.8**	**94.1**	**92.0**

Following F-correction [21], only half of the classes in the dataset are with noisy labels in the setting of asymmetric noise, so the actual noise rate in the whole dataset τ is half of the noisy rate in the noisy classes. Specifically, when the asymmetric noise rate is 0.4, it means $\tau = 0.2$.

For experiments on *Clothing1M*, we adopt the following three settings by following previous work [6]. First, only noisy dataset is used for training without using any extra clean supervision in the training process. Second, verification labels are provided, but they are not used to train the network directly. For instance, they are used to train the accessorial network as [15] or to help select prototypes [6]. Third, both noisy dataset and 50k clean labels are available for training. The data preprocessing procedure includes resizing the image with a short edge of 256 and randomly cropping a 224×224 patch from the resized image.

Baselines. We compare DualGraph with the following state-of-the-art approaches, and implement all approaches with default parameters by PyTorch [20], and conduct all the experiments on NVIDIA 2080Ti GPU.

- F-correction [21], which corrects the prediction by the label transition matrix.
- Joint-optimization [29], which presents a framework for both training classifier and propagating noisy labels to cleaner labels.
- PENCIL [35], which adopts label probability distributions to supervise network learning and to update these distributions through back-propagation.
- Co-teaching [7], which trains two networks simultaneously and cross-updates parameters of peer networks.
- Co-teaching+ [36], which trains two deep neural networks and consists of disagreement-update step and cross-update step.

Network Structure and Optimizer. We use ResNet-12 network architecture for *CIFAR-10* and *CIFAR-100*. Especially, ResNet-12 denotes 4 layer blocks of depth 3 with

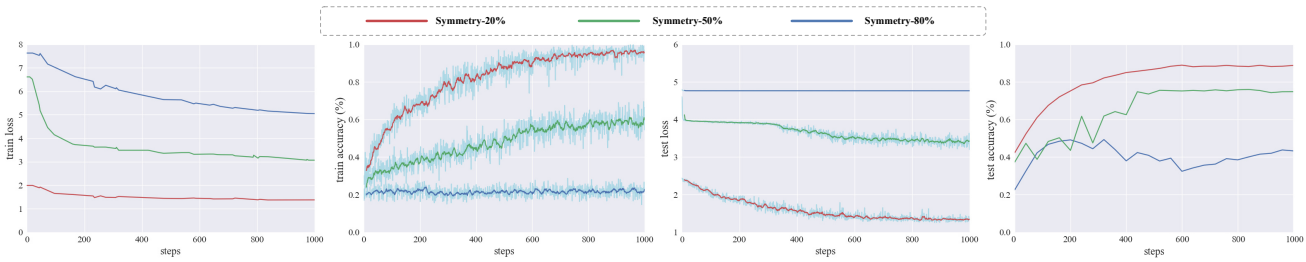

Figure 4. Results on CIFAR-100 dataset with Symmetric Flipping. From left to right: train loss vs. steps, train accuracy(%) vs. steps, test loss vs. steps, test accuracy(%) vs. steps. (1 step here represents 100 steps in the experiment).

Table 3. Average test accuracy (%) on CIFAR-100.

Flipping-Rate	F-correction	Co-teaching	Co-teaching+	Joint-optim	PENCIL	DualGraph
Symmetry-20%	68.74 ± 0.20	78.23 ± 0.27	78.71 ± 0.34	67.61 ± 0.20	73.86 ± 0.34	**88.71** ± 1.23
Symmetry-50%	42.19 ± 0.60	71.30 ± 0.13	57.05 ± 0.54	60.60 ± 0.30	69.12 ± 0.62	**75.80** ± 1.82
Symmetry-80%.	15.88 ± 0.42	26.58 ± 2.22	24.19 ± 2.74	29.60 ± 0.64	24.19 ± 2.74	**50.23** ± 2.93

Table 4. Comparison with state-of-the-art methods in test accuracy (%) on the *Clothing1M*.

#	Method	Data	Accuracy
1	F-correction [21]	1M noisy	69.84
2	Joint-optim [29]	1M noisy	72.16
3	PENCIL [35]	1M noisy	73.49
4	Co-teaching [7]	1M noisy	69.21
5	Meta-Weight-Net [26]	1M noisy	73.72
6	Self-Learning [6]	1M noisy	74.45
7	DualGraph	1M noisy	**80.84**
8	F-correction [21]	1M noisy + 25k verify	75.19
9	Self-Learning [6]	1M noisy + 25k verify	76.44
10	DualGraph	1M noisy + 25k verify	**83.73**
11	F-correction [21]	1M noisy + 50k clean	80.38
12	Self-Learning [6]	1M noisy + 50k clean	81.16
13	DualGraph	1M noisy + 50k clean	**89.82**

3×3 kernels and short connections. The detailed information can be found in supplementary materials. For *Clothing1M*, we use ResNet-34 pre-trained on ImageNet [3].

For experiments on *CIFAR-10* and *CIFAR-100*, Adam optimizer [12] is used with an initial learning rate of 10^{-3}, a weight decay of 10^{-5}, moreover the batch size is set to 64 and 40, respectively. Further, we run 100000 steps in total and linearly decay the learning rate by 0.1 per 15000 steps.

Equally, for experiments on *Clothing1M*, Adam optimizer [12] is used with an initial learning rate of 10^{-3}, a weight decay of 10^{-5}, moreover the batch size is set to 10, respectively. Further, we run 100000 steps in total and linearly decay the learning rate by 0.1 per 15000 steps.

In most of our experiments, the hyperparameters λ_I and λ_D of loss function (14) are set to 1.0 and 0.1 respectively.

4.2. Comparison with the State-of-the-Arts

Results on CIFAR-10. At the top of Figure 3, it shows the training indicators on *CIFAR-10* with symmetric flipping. The memorization effect of networks [2], *i.e.*, test accuracy first reaches a very high level and then gradually decreases. Thus, a robust training method should stop or alleviate the decreasing process. At this point, DualGraph stops the decreasing process and consistently achieves higher accuracy. We can compare the test accuracy of different algorithms in detail in Table 1. In the most natural Symmetry-20% case, all new approaches work well, which demonstrates their robustness. Among them, DualGraph and PENCIL work significantly better than other methods. When it goes to Symmetry-50% case and Asymmetry-40% case, Co-teaching begins to fail while DualGraph, PENCIL and Joint-optim still work fine. However, Co-teaching cannot resist the hardest Symmetry-80% case, where it only achieves 48.58%. In this case, DualGraph achieves the best average classification accuracy (77.2%) again.

The bottom of Figure 3 shows training indicators on *CIFAR-10* with asymmetric flipping. We report experiment results of *CIFAR-10* with asymmetric flipping and other comparative methods in Table 2. We see that in the Asymmetry-50% case, the proposed method outperforms the others by a large margin, *i.e.* improving the accuracy from 83.8% to 92.0%, better than Joint-optim by 16.2% and PENCIL by 11.5%. Furthermore, our model outperforms other comparative methods and achieves 4.1% test accuracy improvements in the asymmetry-20% case.

Results on CIFAR-100. Then, we show our results on *CIFAR-100*. The test accuracy is shown in Table 3. The training indicators on *CIFAR-100* with symmetric flipping are shown in Figure 4. Note that there are only 10 classes in *CIFAR-10* datasets. Thus, overall the accuracy is much lower than the previous Tables 1 and 2. But DualGraph still achieves high test accuracy on *CIFAR-100*. In the easiest Symmetry-20% and Symmetry-50% cases, DualGraph works significantly better than Co-teaching, Co-teaching+

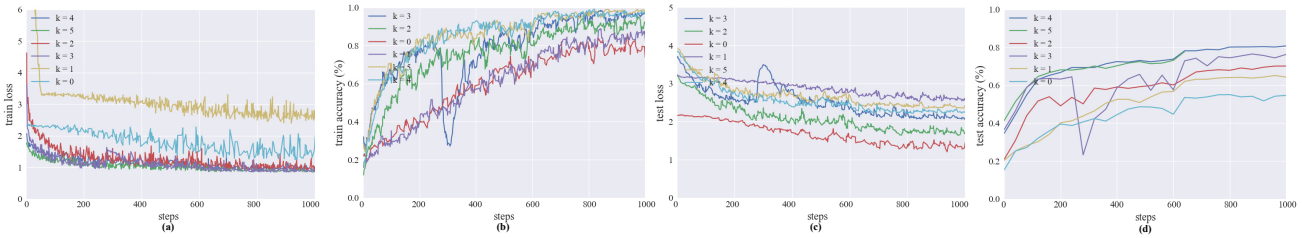

Figure 5. Results of ablation study on *Clothing1M*. Iteration number K in DualGraph from 0 to 5. **(a)** train loss vs. steps; **(b)** train accuracy(%) vs. steps; **(c)** test loss vs. steps; **(d)** test accuracy(%) vs. steps. (1 step here represents 100 steps in the experiment).

and other methods. In the hardest Symmetry-80% case, DualGraph is substantially ahead of Baselines. Furthermore, as shown in the last image in Figure 4, the test accuracy reaches a very high level, then decreases gradually, and finally stabilizes. Such experimental phenomena also indicate that our method resists memorization effect, moreover, is robust to noisy labels.

Results on Clothing1M. Finally, we demonstrate the efficacy of the proposed method on the real-world noisy labels using the *Clothing1M*. We see that in the first case (#1∼#7), the proposed method outperforms the others by a large margin, *e.g.* improving the accuracy from 69.21% to 80.84%, better than Joint Optimization [29] (#2) by 8.68%, PENCIL [35] (#3) by 7.35% and Self-Learning [6] (#6) by 6.30%.

For the second case, [21] (#8) used the information(25k verify) to estimate the transition matrix, while [6] (#9) used the verification labels to select the class prototypes. Our method utilizes this information to initialize the graph neural network so that the graph network has a stable initial topology. In this case, DualGraph still achieves the best result compared to all methods.

For the third case, all data (both noisy and clean) can be used for training. All the methods (#11, #12) first train a model on the noisy dataset and then the model is fine-tuned using vanilla cross-entropy loss on the clean dataset. Among all of these cases, our approach obtains state-of-the-art performances compared to previous methods, showing that our method is effective and suitable for board situations.

4.3. Ablation Studies

4.3.1 Impact of Distribution Graph.

The distribution graph module works as an important component of DualGraph, so it is necessary to investigate the effectiveness of $\mathcal{G}^{D(k)}$ quantitatively. In order to verify the effectiveness of the distributed graph under real-world scenario, we conduct experiments on the *Clothing1M* dataset by occluding the weight λ_D of DualGraph. The cyan line in Figure 5(a)(c) shows that without the supervision of distribution graph, the neural network cannot learn in the label noise environment. The lines in other colors represent the results of applying distribution graph under different K, which are greatly improved in comparison with the case of $K = 0$.

4.3.2 Impact of Iteration Number.

As literatures [25, 41] in show, stacking multiple GCN layers will result in over-smoothing, that is to say, all vertices will converge to the same value. In order to study the effect of different iteration number K on the results, we do experiment on the *Clothing1M* dataset by setting different values of K. We take K from 0 to 5, and the result is shown in Figure 5. Obviously, the verification accuracy and value of K kept in $\mathcal{G}^{D(k)}$ have a positive correlation, that is, the accuracy increases with K. As the value of K increases, the robustness of DualGraph under label noise gradually improves. But when the value of K increases to a certain extent, the difference in the forward propagation distribution of the graph neural network cannot add new information.

5. Conclusion

In this paper, we are the first to exploit the graph neural network that captures structural relationships among labels at two different levels. To leverage both instance-level and distribution-level representation of each example and process the representations at different levels independently, we propose the DualGraph. Moreover, we propose an iterate optimization mechanism that contains two phases, which train two graph neural networks alternatively. In the cyclic execution, to further improve robustness to noisy label, the distribution graph extracts instance similarity distribution to refine the similarity between samples of instance graph. Extended experiments have demonstrated that DualGraph can effectively improve the classification accuracy of noise labels from real-world data via capturing the Instance Similarity Distribution. For future work, we plan to explore using the proposed method to other domains with different model architectures, such as Recurrent Neural Networks for machine translation with corrupted ground-truth sentences.

Acknowledgement

This work is supported in part by National Key R&D Program of China (2017YFB1003000) and NSFC (62061146002,61632008,61921003,61871046).

References

[1] Gorkem Algan and Ilkay Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *arXiv preprint arXiv:1912.05170*, 2019. 1, 2

[2] Devansh Arpit, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning (ICML)*, volume 70, pages 233–242, 2017. 1, 2, 7

[3] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 1, 7

[4] J. Goldberger and E. Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[5] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *International journal of computer vision*, 106(2):210–233, 2014. 1

[6] Han, Jiangfan, Ping Luo, and Xiaogang Wang. Deep self-learning from noisy labels. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5138–5147, 2019. 1, 2, 5, 6, 7, 8

[7] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems (NIPS)*, pages 8527–8537, 2018. 1, 2, 5, 6, 7

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016. 1

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, volume 37, page 448456, 2015. 4, 5

[10] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Ondrej Chum, and Cordelia Schmid. Graph convolutional networks for learning with few clean and many noisy labels. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[11] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning (ICML)*, pages 2304–2313, 2018. 2, 5

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 7

[13] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012. 5

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1

[15] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5447–5456, 2018. 6

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1

[17] Eran Malach and Shai Shalev-Shwartz. Decoupling when to update from how to update. In *Advances in Neural Information Processing Systems (NIPS)*, pages 960–970, 2017. 1, 2

[18] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *International Conference on Machine Learning (ICML)*, volume 37, pages 125–134, 2015. 2

[19] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems (NIPS)*, volume 26, 2013. 2

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NIPS)*, pages 8024–8035. 2019. 6

[21] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1944–1952, 2017. 2, 5, 6, 7, 8

[22] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99. 2015. 1

[24] Brendan van Rooyen, Aditya Krishna Menon, and Robert C. Williamson. Learning with symmetric label noise: The importance of being unhinged. In *International Conference on Neural Information Processing Systems (NIPS)*, page 1018, 2015. 5

[25] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. 3, 8

[26] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in neural information processing systems (NIPS)*, 2019. 7

[27] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020. 1, 2

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 4, 5

[29] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5552–5560, 2018. 1, 2, 5, 6, 7, 8

[30] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[31] Xiaobo Xia, T. Liu, N. Wang, B. Han, C. Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? In *Advances in Neural Information Processing Systems (NIPS)*, 2019. 2

[32] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2699, 2015. 5

[33] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. 4, 5

[34] Yan Yan, Romer Rosales, Glenn Fung, Ramanathan Subramanian, and Jennifer Dy. Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327, 2014. 1

[35] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7017–7025, 2019. 1, 2, 5, 6, 7, 8

[36] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning (ICML)*, 2019. 1, 2, 5, 6

[37] Xiyu Yu, Tongliang Liu, Mingming Gong, and Dacheng Tao. Learning with biased complementary labels. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 68–83, 2018. 1

[38] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2, 3

[39] Yaobin Zhang, Weihong Deng, Mei Wang, Jiani Hu, Xian Li, Dongyue Zhao, and Dongchao Wen. Global-local gcn: Large-scale label noise cleansing for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[40] Jia-Xing Zhong, Nannan Li, Weijie Kong, Shan Liu, Thomas H Li, and Ge Li. Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1237–1246, 2019. 3

[41] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018. 3, 8