A Topological Filter for Learning with Label Noise

Pengxiang Wu
Department of Computer Science
Rutgers University
NJ, USA
pw241@cs.rutgers.edu

Songzhu Zheng
Department of Applied Mathematics and Statistics
Stony Brook University
NY, USA
zheng.songzhu@stonybrook.edu

Mayank Goswami
Department of Computer Science
City University of New York
NY, USA
mayank.isi@gmail.com

Dimitris Metaxas
Department of Computer Science
Rutgers University
NJ, USA
dnm@cs.rutgers.edu

Chao Chen
Department of Biomedical Informatics
Stony Brook University
NY, USA

Abstract

The noisy labels can deteriorate the performance of deep neural networks due to their high learning capacity. To combat this problem, in this work we propose a new method for filtering the noise. Unlike most existing methods relying on the posterior probability of a noisy classifier, we focus on the much richer spatial behavior of data in the latent representation space. By leveraging the high-order topological information of clean data, we are able to collect most of the clean data, where a high quality model is trained eventually. Theoretically we prove that this topological approach is guaranteed to collect all the clean data with high confidence. Empirical results on different datasets show that our method outperforms the state-of-the-arts and is robust to a broad spectrum of noise types and levels.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

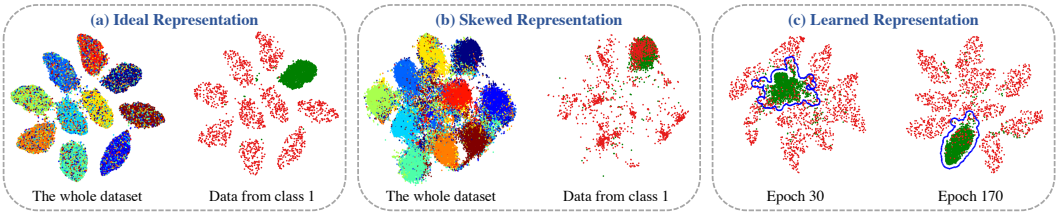


Figure 1: Different representations of a 40% uniformly corrupted CIFAR-10 dataset (visualized using t-SNE). (a) The ideal feature representation (trained on a clean dataset). In the right picture, green points are clean data of class 1, and red points are noisy data whose corrupted labels are 1. (b) A skewed representation of a classifier trained using the corrupted dataset. (c) The learned representations by our algorithm. We show the data of class 1 using the continuously improved representations. The collected data by our method are highlighted with the blue contour.

1 Introduction

Corrupted labels are ubiquitous in real world data, and can severely impair the performance of deep neural networks due to their strong memorization ability [26, 9, 45]. Label noise may arise in mistakes of human annotators or automatic label extraction tools, such as crowd sourcing and web crawling for images [42, 37]. Improving the robustness of deep neural networks to label corruption is critical in many applications [25, 40], yet still remains a challenging problem and largely under-studied.

To combat label noise, state-of-the-art methods often segregate the *clean data* (i.e., samples with uncorrupted labels) from the *noisy* ones. These methods collect clean data iteratively and eventually train a high-quality model. The major challenge is to ensure that the data selection procedure is (1) careful enough to not accumulate errors; and (2) aggressive enough to collect sufficient clean data to train a strong model. Existing methods under this category [23, 17, 13, 38, 27] typically select clean data based on the prediction of the noisy classifier. It is generally assumed that if the noisy classifiers have strong and consistent confidence on a particular label, this label is likely true. However, most of these heuristics do not have a theoretical foundation and thus are not guaranteed to generalize to unseen datasets or noise patterns.

In this paper, we propose to investigate the problem in a novel *topological* perspective. We stipulate that while a noisy classifier's prediction is useful, its latent space representation of the data also contains rich information and should be exploited. Our method is motivated by the following observation: given an ideal feature representation, the clean data are clustered together while the corrupted data are spread out and isolated. This intuition is illustrated in Figure 1(a). We show the spatial distribution pattern of a corrupted dataset with an ideal representation, i.e., the penultimate layer activation (the layer before softmax) of a neural net trained on the original uncorrupted dataset. As is shown in Figure 1(a)(left), the data are well separated into clusters, corresponding to their true labels. Meanwhile, noisy-labeled data (colorful crumbs sprinkled on each cluster) are surrounded by uncorrupted data and thus are isolated.

The above observation inspires us to utilize the spatial topological pattern for label noise filtering. We propose a new method, *TopoFilter*, that collects clean data by selecting the largest connected component of each class and dropping isolated data. Our method leverages the group behavior of data in the latent representation, which has been neglected by previous classifier-confidence-dependent approaches. The challenge is that the ideal representation is unavailable in practice. Training on noisy data leads to skewed representation (Fig. 1(b)); and the topological intuition does not seem to hold.

To address this issue, we propose an algorithm that uses the topological intuition even with the "imperfect" representation. Our algorithm essentially "peels" the outer most layer of the largest component so that only the core of the component is kept. One particular strength of our method is that it is *theoretically guaranteed to be correct*. We prove (1) *purity*: the collected data have a high chance to be uncorrupted; and (2) *abundancy*: the algorithm can collect a majority of the clean data. These two guarantees ensure the algorithm can collect clean data both carefully and aggressively. Our proof only imposes very weak assumptions on the representation. As long as the density of the data has a compact support, and the true conditional distributions of different labels are continuous. This ensures that the theorem still holds on the skewed representation (from a noisy classifier).

We wrap our data collection algorithm to jointly learn the representation and select clean data. To learn the representation, we train a deep net classifier only using the collected clean data. As the classifier continuously improves, it further facilitates the data collection and finally converges to a strong one, as illustrated in Fig. 1(c). We empirically validate the proposed method on several datasets of different noise types, including CIFAR-10, CIFAR-100 and Clothing1M [41]. Our method consistently outperforms the existing methods under a wide range of noise types and levels.

To summarize, we propose the first theoretically guaranteed algorithm for label noise that exploits a topological view of the noisy data representation. Our paper offers both the algorithmic intuition and the theoretical rationale on how spatial pattern and group behavior of data in the latent space can be informative of the model training. We believe the geometry and topology of data in the latent space should be further explored for better understanding and regulating of neural networks.

Related works. One representative class of methods for handling label noise aim to improve the robustness by modeling the noise transition process [33, 28, 12, 15]. However, the estimation of noise transformation is non-trivial, and these methods generally require additional access to the true labels or depend on strong assumptions, which could be impractical. In contrast to these works, our method does not rely on noise modeling, and is thus more generic and flexible.

A number of approaches have sought to develop noise-robust loss to help resist label corruption. One typical idea is to reduce the influence of noisy samples with carefully designed losses [30, 1, 46, 35, 39, 21, 11, 6] or regularizations [16, 24, 19]. Closely related to this philosophy, some other approaches focus on adaptively re-weighting the contributions of the noisy samples to the loss. The re-weighting functions could be pre-specified based on heuristics [5, 38] or learned automatically [17, 31, 32]. Our method is independent of the loss function, and can be combined with any of them.

Another alternative direction seeks to improve the label quality by correcting the noisy labels to the underlying true ones. Representative works include [41, 36, 37, 20, 34, 43]. To predict the true labels, these approaches generally require additional clean labels, complex interventions into the learning process, or an expensive detection process for noise model estimation, which limits their applicability. Moreover, they are based on heuristics without theoretical guarantees, and tend to be sensitive to the hyper-parameters (e.g., learning rate and loss coefficients).

Our work can be categorized as a data-selection method. One typical series of methods belonging to this category choose the clean data based on the prediction disagreements among different networks [23, 27]. In a different spirit, another line of researches train the networks only on samples with small losses and exchange the error flows between networks [13, 8, 44]. One major weakness with these methods is that the training typically involves multiple networks, and is thus computationally expensive and hard to tune. Moreover, the data-selection process in these methods are generally based on heuristics without guarantees. In contrast, our method only needs one network and has theoretical guarantees. Therefore it can be efficient, easy to tune, and easy to generalize.

A few existing works also seek to handle the label noise by probing the spatial properties of data. Wang *et al.* [38] propose to detect noisy data using spatial outlier detection. Gao *et al.* [10] use k-nearest neighbor to correct noisy labels. Both of these methods rely on local spatial information. They fail to explore global structural information that could reveal critical common patterns, such as topology. Lee *et al.* [18] model the spatial distributions with a generative model and train a robust generative classifier using all noisy data. For completeness, we also refer to works studying KNN-induced connectivity [22, 7], which only focus on the unsupervised setting.

2 Method

Our algorithm jointly trains a neural network and collects clean data. At each epoch, clean data are collected based on the their spatial topology in the latent space of the current network. Meanwhile, only clean data are used to further train the network. In the beginning, we use an early-stopped noisy classifier to learn the representation. It has been observed that an early-stopped model will learn meaningful feature without overfitting the noise [45, 2]. Such a network, although not powerful enough, can provide a reasonable initial representation for our data-collection algorithm to start.

Below we present our algorithm. We first provide a baseline, called *TopoCC*. It collects clean data only using the largest connected component. However, this is insufficient due to the imperfect representation. Next, we present our main algorithm, called *TopoFilter*, that further "peels" the largest component and only keeps its core.

Algorithm 1 TopoFilter

1: **Input:** Noisy training data $\mathcal{D}$, milestone m, training epochs N, class number $\mathcal{C}$, filtering parameter ζ
2: Initialize $S \leftarrow \emptyset$, $\widehat{\mathcal{D}} \leftarrow \mathcal{D}$
3: **for** $t = 1, \cdots, N$ **do**
4: Train network on $\widehat{\mathcal{D}}$
5: **if** $t \geq m$ **then**
6: Extract feature vectors $\boldsymbol{x}$ from training data $\mathcal{D}$
7: Compute KNN graph G over $\boldsymbol{x}$
8: **for** $i = 1, \cdots, \mathcal{C}$ **do**
9: Construct subgraph G_i by selecting feature vectors $\boldsymbol{x}^{(i)}$ from i-th class and removing all edges associated with $\boldsymbol{x}^{(k)}$ for $k \neq i$
10: Compute the largest connected component S_i over G_i
11: $S \leftarrow S \cup S_i$
12: **end for**
13: Find outliers O within S based on ζ-filtering
14: $\widehat{\mathcal{D}} \leftarrow S - O$
15: **end if**
16: **end for**

Our algorithm for data selection is as follows. Let $\boldsymbol{v}$ be the input data and F represent the neural network. Given latent features $\boldsymbol{x} = F(\boldsymbol{v})$, we probe the spatial data distributions by building a k-nearest neighbor (KNN) graph G upon $\boldsymbol{x}$. From G we further derive the subgraph G_i for class i by removing the vertices belonging to other classes and their associated edges. On each G_i, we find the largest connected component S_i and consider the data belonging to S_i as clean. Eventually we have a collection of potentially clean data $S = \cup_i S_i$. Intuitively, the clean data will be regularly and densely distributed in the feature space. They will form a salient topological structure (connected component), which could thus be captured by the algorithm. Plugging this data-collection procedure into our joint training algorithm gives the baseline *TopoCC*.

However, simply relying on connected components is insufficient; the geometry and thus connectivity of the data is not fully reliable due to the imperfect representation. In particular, near the outer most layer of the largest connected component, the data can easily be corrupted ones. We need to remove these data in order to improve the purity of the selected data. In particular, for a given sample $\boldsymbol{x}$ belonging to one of the largest connected components S_i, with label $\widetilde{y}$, find its k-nearest neighbors $KNN(\boldsymbol{x})$ from S (the union of largest components for each class). Then we consider $\boldsymbol{x}$ as clean if at least a fraction ζ of its neighbors have the same label $\widetilde{y}$. As is illustrated in Section 2.1 and Section 3, this additional filtering of the largest component, called the ζ-filtering, is essential to the success of our method. We name this method *TopoFilter*. Details are in Algorithm 1.

2.1 Purity and abundancy of the collected data

Below we give a detailed analysis on the behavior of our method. We give two theorems in this paper that lower bound the purity and size of the final data selected by our algorithm, respectively.

Purity. Theorem 1 shows that all data points of type i collected by the algorithm have a high chance to have underlying true label as i. The first step of our algorithm, collecting the largest connected component, helps in pruning out points which have low posterior probability of being type i. This is crucial because although *the network just outputs some approximation of the posterior probability, topology helps in guaranteeing that all points we want to collect are picked out after the first step* . The second step (line 13 in Algorithm 1), ζ-filtering, then does a more careful pruning, and removes points with a possible low posterior probability that we may have collected in the first step. As a result, the purity (P[true label=i, given observed label =i]) of labels for the final preserved data points will be high.

Abundancy. Theorem 5 derives a lower bound on the number of data points kept by our algorithm. Essentially, our algorithm guarantees that almost all points $\boldsymbol{x}$ in the original dataset with posterior probability at least ζ are retained.

Setting and notation. Assume that the data points and labels lie in $\mathcal{X} \times \mathcal{Y}$, where the features $\mathcal{X} \subset R^d$ and labels $\mathcal{Y} := [\mathcal{C}] := \{1, 2, 3, \cdots, \mathcal{C}\}$. Assume the (data, true label) pairs follow some distribution $\mathcal{F} \sim \mathcal{X} \times \mathcal{Y}$. Let $f(\boldsymbol{x}) := \sum_{i \in [\mathcal{C}]} \mathcal{F}(\boldsymbol{x}, i)$ be the density at $\boldsymbol{x}$. Due to label noise, label $y = i$ is flipped to $\tilde{y} = j$ with probability τ_{ij} and is assumed to be independent of $\boldsymbol{x}$.

Let $\mathbf{X} \subset \mathcal{X}$ be the finite set of features in the data sample, and let $G(\mathbf{X}, k)$ be the mutual k-nearest neighbor graph on $\mathbf{X}$ using the Euclidean metric on $\mathcal{X}$, whose edge set $E = \{(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbf{X}^2 \mid \boldsymbol{x}_1 \in KNN(\boldsymbol{x}_2)$ or $\boldsymbol{x}_2 \in KNN(\boldsymbol{x}_1)\}$. Also, $\forall i \in [\mathcal{C}]$, let $G_i(\mathbf{X}, k)$ be the induced subgraph of $G(\mathbf{X}, k)$ consisting only of vertices $\boldsymbol{x} \in \mathbf{X}$ with label $\widetilde{y}(\boldsymbol{x}) = i$.

Let $\eta_i(\boldsymbol{x}) = P(y = i \mid \boldsymbol{x})$ and $\widetilde{\eta}_i(\boldsymbol{x}) = P(\widetilde{y} = i \mid \boldsymbol{x})$ be the prior and posterior probability of labels given a feature $\boldsymbol{x}$, respectively. For simplicity, we focus on the binary label case for now. Then for $i \in \{0, 1\}$, these two probabilities are related by $\widetilde{\eta}_i(\boldsymbol{x}) = (1 - \tau_{01} - \tau_{10})\eta_i(\boldsymbol{x}) + \tau_{1-i,i}$. Define the super level set $L(t) = \{\boldsymbol{x} \mid \max(\eta_1(\boldsymbol{x}), \eta_0(\boldsymbol{x})) \geq t\}$. Lastly, the indicator function $I_{\mathcal{A}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \mathcal{A}$ and $I_{\mathcal{A}}(\boldsymbol{x}) = 0$ otherwise.

Define the following three sets, which form a partition of $\mathcal{X}$:

$$
\begin{aligned}
A_i^+ &= \left\{\boldsymbol{x} : \widetilde{\eta}_i(\boldsymbol{x}) > \max(\tfrac{1}{2}, \tfrac{1+\tau_{i,1-i}-\tau_{1-i,i}}{2}))\right\} = \left\{\boldsymbol{x} : \eta_i(\boldsymbol{x}) > \max(\tfrac{1}{2}, \tfrac{1/2-\max(\tau_{10},\tau_{01})}{2(1-\tau_{10}-\tau_{01})})\right\}, \\
A_i^- &= \left\{\boldsymbol{x} : \widetilde{\eta}_i(\boldsymbol{x}) < \min(\tfrac{1}{2}, \tfrac{1+\tau_{i,1-i}-\tau_{1-i,i}}{2}))\right\} = \left\{\boldsymbol{x} : \eta_i(\boldsymbol{x}) < \min(\tfrac{1}{2}, \tfrac{1/2-\max(\tau_{10},\tau_{01})}{2(1-\tau_{10}-\tau_{01})})\right\}, \\
A^b &= \mathcal{X} \setminus (A_i^+ \cup A_i^-).
\end{aligned}
$$

Consider an algorithm $\mathcal{A}$ that takes as input a random sample of size n, $S_n = \{(\boldsymbol{x}_i, \tilde{y}(\boldsymbol{x}_i))\}_{i=1}^n$, and let $\mathbf{X} := \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{X}$. Algorithm $\mathcal{A}$ then outputs $\cup_{i \in \{0,1\}} C_i$, where $C_i \subseteq \mathbf{X}_i := \{\boldsymbol{x} : \widetilde{y}(\boldsymbol{x}) = i\}$ is the claimed "clean" set for label i.

Definition 1 (Purity). *We define two kinds of purity of $\mathcal{A}$ on S_n. One captures the worst-case behavior of the algorithm, while the other captures the average-case behavior.*

1. Minimum Purity $\ell_{S_n,\mathcal{A}} := \min_{i \in \{0,1\}} \min_{\boldsymbol{x} \in C_i} P(y = i \mid \widetilde{y} = i, \boldsymbol{x}) = \min_{i \in \{0,1\}} \min_{\boldsymbol{x} \in C_i} \tau_{ii} \frac{\eta_i(\boldsymbol{x})}{\widetilde{\eta}_i(\boldsymbol{x})}$.

2. Average Purity $\ell'_{S_n,\mathcal{A}} := \sum_{i \in \{0,1\}} \frac{1}{C_i} \sum_{\boldsymbol{x} \in C_i} \tau_{ii} \frac{\eta_i(\boldsymbol{x})}{\widetilde{\eta}_i(\boldsymbol{x})}$.

Assumptions. To establish our theorems, we will assume the following reasonable conditions:

A1: $f(\boldsymbol{x})$ (the density on the feature space) has compact support.

A2: $\forall i \in \{0, 1\}$, $\eta_i(\boldsymbol{x})$ is continuous. This is a significantly weaker conditions on η_i than is assumed in many prior works on KNN classifiers (such as Hölder continuous [3], Lipschitz continuous [10], etc.). This condition upper bounds the measure of region that are close to the decision boundary of Bayes decision rule. This assumption is also adopted in [7, 4, 29] and is reasonably well accepted.

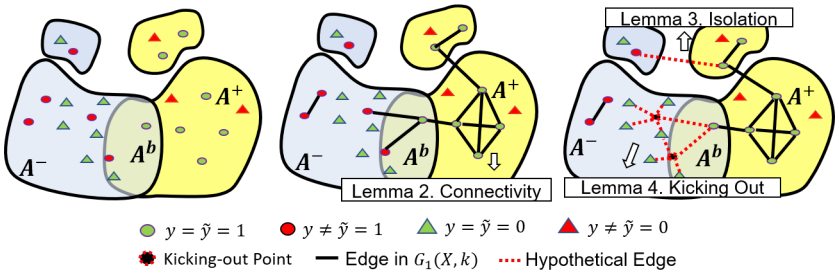


Figure 2: Algorithm illustration. Points from A_i^+ are all connected; points in $A_i^- \cup A^b$ are kicked out.

Denote by $\mathcal{A}_0$ the naive algorithm which takes input S_n and simply outputs $C_i = \mathbf{X}_i$ for $i = 0, 1$, i.e., does no processing and treats corrupted labels as clean. The purity of $\mathcal{A}_0$ is the "default" purity of the data set. Denote our algorithm with parameter ζ by $\mathcal{A}_\zeta$.

Theorem 1. ***(Purity Guarantee)*** *$\forall \delta > 0$, $\forall \zeta > \frac{1+|\tau_{10}-\tau_{01}|}{2}$ and $\forall q > 1$, there exist constants $N > 0$, $C_1 > 0$, $C_2 \in (0, 1)$, and an increasing function $g_1(\zeta) \in \left[\frac{[\zeta-\max(\tau_{10},\tau_{01})]\min(\tau_{11},\tau_{00})}{\zeta(1-\tau_{10}-\tau_{01})}, 1\right]$ and a constant $C_\zeta > 0$, such that $\forall n \geq N$ and $\forall k \in [C_1 \log^q n, C_2 n]$:*

1. $P\left[(\ell_{S_n,\mathcal{A}_\zeta} - \ell_{S_n,\mathcal{A}_0}) > g_1(\zeta)\right] \geq 1-\delta$, *and*

2. $P\left[(\ell'_{S_n,\mathcal{A}_\zeta} - \ell'_{S_n,\mathcal{A}_0}) > C_\zeta\right] \geq 1-\delta$.

Sketch of Proof. The complete proof is in the supplementary materials; here we provide a sketch and the main lemmas used to prove Theorem 1. Firstly, we show that for a given $t \in [0,1)$, when the sample size n is large enough and the number of neighbors k is set to be $\Omega(\log^q(n))$, then all data points from $X_i(t) := L(t) \cap \mathbf{X}_i$ will be connected in $G_i(\mathbf{X}, k)$.

Lemma 2. ***(Connectivity).*** *$\forall\delta > 0$ and $\forall t \in [0,1)$, there exist constants $N_0 > 0$, and $C_1 > 0$ such that $\forall n \geq N_0$, $\forall i \in \{0,1\}$, $\forall q > 1$, and $\forall k > C_1 \log^q(n)$, $X_i(t)$ is connected in $G_i(\mathbf{X}, k)$ with probability at least $1-\delta$.*

Let $\zeta' = \frac{1}{2}(\frac{1}{2} + \frac{(1+|\tau_{10}-\tau_{01}|)}{2})$. Notice that because $\zeta > \frac{1+|\tau_{10}-\tau_{01}|}{2}$, $\zeta > \zeta'$ and $L(\zeta) \subset L(\zeta') \subset A_i^+$. Next we prove that when k is not too large, there will be no points in $A_i^+ \cap L(\zeta)$ that have an edge to a point in $L(\zeta')^c$. Denote $\mathbf{X}_i^c(\zeta') := \overline{L(\zeta')^c} \cap \mathbf{X}_i$. Define $r_0^i = \min \left\|\boldsymbol{x}_1^i - \boldsymbol{x}_2^i\right\|$ for $\boldsymbol{x}_1^i \in X_i(\zeta)$ and $\boldsymbol{x}_2^i \in X_i^c(\zeta')$. Also observe that A_i^+, A_i^- and A^b form a partition of the domain, which along with the assumption A1 of compact support implies that $r_0 > 0$. Let V_d to be the volume of d-dimensional unit ball. Let $p_\zeta^{(i)} = \min\limits_{\boldsymbol{x} \in L(\zeta) \cap A_i^+} f(\boldsymbol{x}) V_d (r_0^i)^d$ and $p_{\zeta'}^{(i)} = \inf\limits_{\boldsymbol{x} \in L(\zeta')^c \cap A_i^{+c}} f(\boldsymbol{x}) V_d (r_0^i)^d$. Since $f(\boldsymbol{x})$ has compact support, $p_\zeta^{(i)} > 0$ and $p_{\zeta'}^{(i)} > 0$.

Lemma 3. ***(Isolation).*** *$\forall\delta > 0$, $\forall\zeta > \frac{1+|\tau_{10}-\tau_{01}|}{2}$, there exists $N > 0$ such that $\forall i \in \{0,1\}$, $\forall n \geq N$ and $\forall k < \min\limits_{i\in\{0,1\}} \min\left(p_\zeta^{(i)}, p_{\zeta'}^{(i)}\right)(n-1)+1$,*

$$P\left(\nexists e = (u,v) \in G_i(\mathbf{X}, k) : u \in \mathbf{X}_i(\zeta), v \in \mathbf{X}_i^c(\zeta')\right) \geq 1-\delta.$$

Then we show after the ζ-filtering step, with large probability there will be no points from $L(\zeta')^c$ in our final set. Denote $C^{(i)}(\zeta)$ to be the data of type i finally kept by the algorithm using parameter ζ.

Lemma 4. ***(ζ-filtering).*** *$\forall\delta > 0$, $\forall i \in \{0,1\}$ and $\zeta \in \left(\frac{1+|\tau_{10}-\tau_{01}|}{2}, 1\right)$, there exists $N > 0$ and $C_2 > 0$, such that $\forall n \geq N$ if we set $k > C_2 \log(2n/\delta)$ then:*

$$P\left(C^{(i)}(\zeta) \cap L(\zeta')^c = \emptyset\right) \geq 1-\delta.$$

To obtain Theorem 1, we combine the above lemmas as follows. The minimum purity of a data point retained by our algorithm is lower bounded by the purity of a point in the level set $\{x : \tilde{\eta}(\boldsymbol{x}) = \zeta'\}$, which followed by algebraic calculation gives us the first part of the theorem. For the second part of the theorem, we need to calculate the average purity for our algorithm, which requires a more involved integral calculation over $L(\zeta')$. The complete proof is in the supplementary materials.

Our next theorem (proof in supplementary materials) gives a lower bound on the number of points that will be eventually kept by our algorithm $\mathcal{A}_\zeta$. As we will see, there will be a trade off between the size of the retained set and its purity. A larger ζ will result in smaller connected component but higher purity, while small ζ gives large connected component but lower purity.

Theorem 5. ***(Abundancy)*** *Let $n_c = \#\left\{\bigcup_i C^{(i)}(\zeta)\right\}$. Then $\forall\delta > 0$, $\forall\zeta > \frac{1+|\tau_{10}-\tau_{01}|}{2}$, $\forall\epsilon > 0$, there exist constants $C_1 > 0$, $C_2 \in (0,1)$ and $N > 0$, such that $\forall n \geq N$, and $\forall k \in [C_1 \log^q n, C_2 n]$, we have $P\left[|n_c/n - \mu(L(\zeta))| \leq \epsilon\right] \geq 1-\delta$.*

Remark on choosing ζ: In the beginning epochs, because of the corruption of the distribution, one does not expect high confidence in the network classifier. In order to guarantee a minimum level of purity, we set ζ to be high, say $3/4$. While in these rounds the purity is high and the abundancy is lower bounded, we still want to increase abundancy further. In the later epochs, after training on this clean(er) data, we develop more confidence in our network classifier, and hence we reduce ζ, letting ζ go to $(1/2+\epsilon)$ for a very small $\epsilon > 0$, which corresponds to collecting data from the boundary region of the Bayesian classifier. More discussion can be found in the supplemental material.

3 Experiments

Synthetic datasets. We test the proposed method on CIFAR-10 and CIFAR-100, which are popularly used for the study of label noise. We preprocess each image by normalizing it with the training set mean and standard deviation. For each of the datasets, we split 20% from the training set as validation data. The validation set could be noisy or clean, whereas we only use clean testing data. We employ ResNet-18 [14] as the experimental network, which achieves reasonable performance on the two datasets, with 92.0% and 70.4% test accuracies on CIFAR-10 and CIFAR-100, respectively.

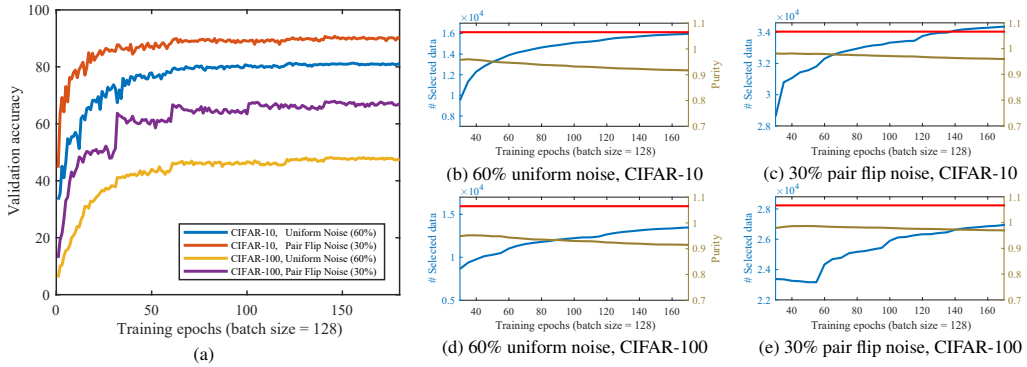
Generating corrupted labels. Similar to [28], we artificially corrupt the labels by constructing the noise transition matrix T, where $T_{ij} = P(\widetilde{y} = j|y = i) = \tau_{ij}$ defines the probability that a true label $y = i$ is flipped to j. Then for each sample with label i, we replace its label with the one sampled from the probability distribution given by the i-th row of matrix T. In this work, we consider two types of noise, both of which can be formulated using the transition matrices. (1) *Uniform flipping*: the true label i is corrupted uniformly to other classes, i.e., $T_{ij} = \tau/(\mathcal{C} - 1)$ for $i \neq j$, and $T_{ii} = 1 - \tau$, where τ is the constant noise level; (2) *Pair flipping*: the true label i is flipped to j or stays unchanged with probabilities $T_{ij} = \tau$ and $T_{ii} = 1 - \tau$, respectively. Pair flipping is used to simulate the real mistakes made by human labelers on similar classes. For more details we refer the readers to [28].

Baselines. We compare the proposed method with the following representative approaches. (1) *Standard*, which is simply the standard deep network trained on noisy datasets; (2) *Forgetting* [2]; (3) *Bootstrap* [30]; (4) *Forward Correction* [28]; (5) *Decoupling* [23]; (6) *MentorNet* [17]; (7) *Co-teaching* [13]; (8) *Co-teaching+* [44]; (9) *IterNLD* [38]; (10) *RoG* [18]; (11) *PENCIL* [43]; (12) *GCE* [46]; (13) *SL* [39]. These methods are from different research directions.

We implement our method with PyTorch. For data selection, we compute the KNN graph with CUDA and calculate the largest connected component in C++: the overall computational cost per iteration is less than 1s on an Intel Xeon Gold 5218 CPU and a single NVIDIA RTX 4000 GPU. We use a batch size of 128 and train the networks for 180 epochs to ensure convergence. We train the network with Adam optimizer using its default parameters. The data selection is performed every 5 epochs. All experiments are randomly repeated 5 times, and the mean and standard deviation values are reported. All the methods use clean validation set for model selection. Note that, our method is robust to the validation set, as shown below.

Results. Table 1 shows the performance of different methods. We observe that TopoFilter consistently outperforms the competitive methods across different noise settings. This suggests the benefits of leveraging spatial pattern for label denoising. Notice that, although the posterior probabilities employed by a few works are closely related to the penultimate layer features used in our method, they intrinsically undergo a dimension reduction process and may lose some critical information. This would explain the superior performance of our method to some degree.

From Table 1 we also observe that simply using largest connected components (TopoCC) or spatial outliers (IterNLD) are less effective. This is because the data in the connected components could still contain noise, and thereby hurts the model performance eventually. Similarly, the outlier detection is



(a) (b) 60% uniform noise, CIFAR-10 (c) 30% pair flip noise, CIFAR-10 (d) 60% uniform noise, CIFAR-100 (e) 30% pair flip noise, CIFAR-100

Figure 3: (a) Validation accuracies. (b-e) The size of selected data (the blue curve) and its purity (the brown curve). The red line denotes the upper-bound size of clean data.

Table 1: Test accuracies (%) on CIFAR-10 and CIFAR-100 under different noise types and fractions. The average accuracies and standard deviations over 5 trials are reported.

Dataset	Method	Uniform Flipping				Pair Flipping		
		20%	40%	60%	80%	20%	30%	40%
CIFAR-10	Standard	85.7 ± 0.5	81.8 ± 0.6	73.7 ± 1.1	42.0 ± 2.8	88.0 ± 0.3	86.4 ± 0.4	84.9 ± 0.7
	Forgetting	86.0 ± 0.8	82.1 ± 0.7	75.5 ± 0.7	41.3 ± 3.3	89.5 ± 0.2	88.2 ± 0.1	85.0 ± 1.0
	Bootstrap	86.4 ± 0.6	82.5 ± 0.1	75.2 ± 0.8	42.1 ± 3.3	88.8 ± 0.5	87.5 ± 0.5	85.1 ± 0.3
	Forward	85.7 ± 0.4	81.0 ± 0.4	73.3 ± 1.1	31.6 ± 4.0	88.5 ± 0.4	87.3 ± 0.2	85.3 ± 0.6
	Decoupling	87.4 ± 0.3	83.3 ± 0.4	73.8 ± 1.0	36.0 ± 3.2	89.3 ± 0.3	88.1 ± 0.4	85.1 ± 1.0
	MentorNet	88.1 ± 0.3	81.4 ± 0.5	70.4 ± 1.1	31.3 ± 2.9	86.3 ± 0.4	84.8 ± 0.3	78.7 ± 0.4
	Co-teaching	89.2 ± 0.3	86.4 ± 0.4	79.0 ± 0.2	22.9 ± 3.5	90.0 ± 0.2	88.2 ± 0.1	78.4 ± 0.7
	Co-teaching+	89.8 ± 0.2	86.1 ± 0.2	74.0 ± 0.2	17.9 ± 1.1	89.4 ± 0.2	87.1 ± 0.5	71.3 ± 0.8
	IterNLD	87.9 ± 0.4	83.7 ± 0.4	74.1 ± 0.5	38.0 ± 1.9	89.3 ± 0.3	88.8 ± 0.5	85.0 ± 0.4
	RoG	89.2 ± 0.3	83.5 ± 0.4	77.9 ± 0.6	29.1 ± 1.8	89.6 ± 0.4	88.4 ± 0.5	86.2 ± 0.6
	PENCIL	88.2 ± 0.2	86.6 ± 0.3	74.3 ± 0.6	45.3 ± 1.4	90.2 ± 0.2	88.3 ± 0.2	84.5 ± 0.5
	GCE	88.7 ± 0.3	84.7 ± 0.4	76.1 ± 0.3	41.7 ± 1.0	88.1 ± 0.3	86.0 ± 0.4	81.4 ± 0.6
	SL	89.2 ± 0.5	85.3 ± 0.7	78.0 ± 0.3	44.4 ± 1.1	88.7 ± 0.3	86.3 ± 0.1	81.4 ± 0.7
	TopoCC	89.6 ± 0.3	86.0 ± 0.5	78.7 ± 0.5	43.0 ± 2.0	89.8 ± 0.3	87.3 ± 0.3	85.4 ± 0.4
	TopoFilter	$\mathbf{90.2 \pm 0.2}$	$\mathbf{87.2 \pm 0.4}$	$\mathbf{80.5 \pm 0.4}$	$\mathbf{45.7 \pm 1.0}$	$\mathbf{90.5 \pm 0.2}$	$\mathbf{89.7 \pm 0.3}$	$\mathbf{87.9 \pm 0.2}$
CIFAR-100	Standard	56.5 ± 0.7	50.4 ± 0.8	38.7 ± 1.0	18.4 ± 0.5	57.3 ± 0.7	52.2 ± 0.4	42.3 ± 0.7
	Forgetting	56.5 ± 0.7	50.6 ± 0.9	38.7 ± 1.0	18.4 ± 0.4	57.5 ± 1.1	52.4 ± 0.8	42.4 ± 0.8
	Bootstrap	56.2 ± 0.5	50.8 ± 0.6	37.7 ± 0.8	19.0 ± 0.6	57.1 ± 0.9	53.0 ± 0.9	43.0 ± 1.0
	Forward	56.4 ± 0.4	49.7 ± 1.3	38.0 ± 1.5	12.8 ± 1.3	56.8 ± 1.0	52.7 ± 0.5	42.0 ± 1.0
	Decoupling	57.8 ± 0.4	49.9 ± 1.0	37.8 ± 0.7	17.0 ± 0.7	60.2 ± 0.9	54.9 ± 0.1	47.2 ± 0.9
	MentorNet	62.9 ± 1.2	52.8 ± 0.7	36.0 ± 1.5	15.1 ± 0.9	62.3 ± 1.3	55.3 ± 0.5	44.4 ± 1.6
	Co-teaching	64.8 ± 0.2	60.3 ± 0.4	46.8 ± 0.7	13.3 ± 2.8	63.6 ± 0.4	58.3 ± 1.1	48.9 ± 0.8
	Co-teaching+	64.2 ± 0.4	53.1 ± 0.2	25.3 ± 0.5	10.1 ± 1.2	60.9 ± 0.3	56.8 ± 0.5	48.6 ± 0.4
	IterNLD	57.9 ± 0.4	51.2 ± 0.4	38.1 ± 0.9	15.5 ± 0.8	58.1 ± 0.4	53.0 ± 0.3	43.5 ± 0.8
	RoG	63.1 ± 0.3	58.2 ± 0.5	47.4 ± 0.8	20.0 ± 0.9	67.1 ± 0.6	65.6 ± 0.4	58.8 ± 0.1
	PENCIL	64.9 ± 0.3	61.3 ± 0.4	46.6 ± 0.7	17.3 ± 0.8	67.5 ± 0.5	66.0 ± 0.4	61.9 ± 0.4
	GCE	63.6 ± 0.6	59.8 ± 0.5	46.5 ± 1.3	17.0 ± 1.1	64.8 ± 0.9	61.4 ± 1.1	50.4 ± 0.9
	SL	62.1 ± 0.4	55.6 ± 0.6	42.7 ± 0.8	19.5 ± 0.7	59.2 ± 0.6	55.1 ± 0.7	44.8 ± 0.1
	TopoCC	64.1 ± 0.8	57.3 ± 1.6	45.1 ± 1.1	19.1 ± 0.6	66.3 ± 0.8	62.3 ± 0.9	58.3 ± 0.9
	TopoFilter	$\mathbf{65.6 \pm 0.3}$	$\mathbf{62.0 \pm 0.6}$	$\mathbf{47.7 \pm 0.5}$	$\mathbf{20.7 \pm 1.2}$	$\mathbf{68.0 \pm 0.3}$	$\mathbf{66.7 \pm 0.6}$	$\mathbf{62.4 \pm 0.2}$

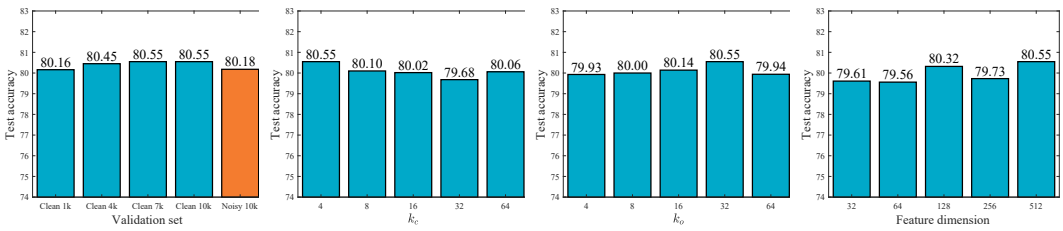


Figure 4: Parameter analysis: (a) validation set; (b) k_c; (c) k_o; (d) Feature dimension. We use uniform flipping noise (60%) and CIFAR-10. For each figure, we change one of the parameters while keeping the others fixed. (to $k_c = 4$, $k_o = 32$, feature dimension = 512, validation set = clean 10k).

not reliable as the noisy data could form a small cluster and thus do not appear to be outliers spatially, as illustrated in Fig. 1(a)(b).

Behavior analysis. In Fig. 3(a) we show the validation accuracy of TopoFilter on the two datasets with different noise settings. As is shown, the accuracy of TopoFilter does not drop throughout the training process. This indicates that our method filters out the noise successfully and stably. This is further confirmed in Fig. 3(b)-(e), where the collected data pool preserves high purity during training with its size approaching the limit steadily.

Parameter analysis. In Fig. 4(a) we show that TopoFilter is robust to the size and purity of validation set. Notably, it achieves almost the same performance even with noisy or small clean validation data. In Fig. 4(b)(c) we demonstrate that TopoFilter is insensitive to the parameters k_c and k_o, up to a wide range. Here k_c and k_o represent the parameters for computing the k-nearest neighbors in largest connected component and outlier detection, respectively. This is consistent with our theoretical findings in Section 2.1. In Fig. 4(d), we show that TopoFilter is robust to the feature dimensions. See Appendix for more results.

Table 2: Classification accuracy (%) on Clothing1M test set.

Method	Standard	Forward	D2L	Joint Opt.	PENCIL	MLNT	DY	GCE	SL	**TopoFiler**
Accuracy	68.94	69.84	69.47	72.23	73.49	73.47	71.00	69.75	71.02	**74.10**

Real-world corrupted dataset. To test the effectiveness of TopoFilter in real setting, we conduct experiments on the Clothing1M dataset [41]. This dataset consists of 1 million clothing images obtained from online shopping websites with 14 classes. The labels in this dataset are extremely noisy (with an estimate accuracy of 61.54%) and their structure is unknown. This dataset also provides 50k, 14k and 10k manually verified clean data for training, validation, and testing, respectively. Following [34, 43, 39], we evaluate the classification accuracy on the 10k clean data and do not use the 50k clean training data. Similarly, we use ResNet-50 with weights pre-trained on ImageNet and train the model with SGD. We set the batch size 32, learning rate 0.001, and preprocess the images following the same procedure in [34, 43, 39]. We train the model for 10 epochs and collect the clean data per epoch. The cost of computing k-nearest neighbors and connected components in data selection is about 25s.

We compare our method with the following ones: (1) *Standard*; (2) *Forward Correction* [28]; (3) *D2L* [21] (4) *Joint Optimization* [34]; (5) *PENCIL* [43]; (6) *MLNT* [19]; (7) *DY* [1]; (8) *GCE* [46]; (9) *SL* [39]. As is shown in Table 2, TopoFilter obtains the best performance compared to the baseline methods. This demonstrates its applicability to the real-world scenarios beyond the synthetic noise.

4 Conclusion

In this work, we propose a novel method named TopoFilter to facilitate the training of deep neural networks on data with corrupted labels. Our method leverages the topological property of the data in feature space, and jointly learns the data representation and collects the clean data during training. Theoretically, we show that TopoFilter is able to select the most of clean data with high confidence. Our empirical results on different datasets demonstrate the advantages of TopoFilter in improving the robustness of deep models to label noise.

Broad Impact

Label noise is ubiquitous in real-world data. This noise may arise from the cheap but imperfect annotations, such as crowdsourcing and online queries. Moreover, even by human annotators, the data labeling process is still error-prone. Another typical source of noise is the data poisoning, where corruptions are intentionally injected into the labels. Training with noisy labels would severely deteriorate the performance of deep models, due to their strong memorization ability and overfitting on corrupted information [45, 2]. Therefore, limiting the adverse influence of noisy labels is of great practical significance and has gained increasing attention from the community.

In this work we attack the label noise from the perspective of data topology. Different from previous works which mostly inspect the sample losses or predicted posteriors, we show that the spatial behavior of the data could be well exploited, a point that has been largely ignored before. Importantly, in theory we prove that our topology-motivated method is able to exhaustively select the clean data with high probability. In this way we keep the network away from the negative influence of corrupted labels and promote the training healthily and steadily. Our method is simple yet with theoretical insights, and would provide contributions supplementary to the existing works. We believe it deserves the attention from the machine learning community.

References

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

[2] Devansh Arpit, Stanislaw K. Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *ICML*, pages 233–242, 2017.

[3] Dara Bahri, Heinrich Jiang, and Maya Gupta. Deep k-nn for noisy labels. *arXiv preprint arXiv:2004.12289*, 2020.

[4] Mikhail Belkin, Daniel J Hsu, and Partha Mitra. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In *NeurIPS*, 2018.

[5] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *NeurIPS*, pages 1002–1012, 2017.

[6] Nontawat Charoenphakdee, Jongyeong Lee, and Masashi Sugiyama. On symmetric losses for learning from corrupted labels. In *ICML*, 2019.

[7] Kamalika Chaudhuri and Sanjoy Dasgupta. Rates of convergence for the cluster tree. In *Advances in neural information processing systems*, pages 343–351, 2010.

[8] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *ICML*, 2019.

[9] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE Trans. Neural Netw. Learning Syst.*, 25(5):845–869, 2014.

[10] Wei Gao, Bin bin Yang, and Zuo cheng Zhou. On the resistance of nearest neighbor to random noisy labels. 2016.

[11] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *AAAI*, 2017.

[12] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017.

[13] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, pages 8536–8546, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[15] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*, pages 10477–10486, 2018.

[16] W Hu, Z Li, and D Yu. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. In *ICLR*, 2020.

[17] 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 *ICML*, pages 2309–2318, 2018.

[18] Kimin Lee, Sukmin Yun, Kibok Lee, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. In *ICML*, 2019.

[19] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, pages 5051–5059, 2019.

[20] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation. In *ICCV*, pages 1928–1936, 2017.

[21] Xingjun Ma, Yisen Wang, Michael E. Houle, Shuo Zhou, Sarah M. Erfani, Shu-Tao Xia, Sudanthi N. R. Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *ICML*, pages 3361–3370, 2018.

[22] Markus Maier, Matthias Hein, and Ulrike Von Luxburg. Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. *Theoretical Computer Science*, 410(19):1749–1764, 2009.

[23] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *NeurIPS*, pages 960–970, 2017.

[24] Aditya Krishna Menon, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Can gradient clipping mitigate label noise? In *ICLR*, 2020.

[25] Volodymyr Mnih and Geoffrey E. Hinton. Learning to label aerial images from noisy data. In *ICML*, 2012.

[26] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306, 2010.

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

[28] 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 *CVPR*, pages 2233–2241, 2017.

[29] Xingye Qiao, Jiexin Duan, and Guang Cheng. Rates of convergence for large-scale nearest neighbor classification. In *NeurIPS*, 2019.

[30] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR Workshop*, 2014.

[31] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, pages 4331–4340, 2018.

[32] 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 *NeurIPS*, pages 1917–1928, 2019.

[33] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. In *ICLR Workshop*, 2014.

[34] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *CVPR*, pages 5552–5560, 2018.

[35] Sunil Thulasidasan, Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. Combating label noise in deep learning using abstention. In *ICML*, 2019.

[36] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *NeurIPS*, pages 5596–5605, 2017.

[37] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge J. Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, pages 6575–6583, 2017.

[38] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels. In *CVPR*, pages 8688–8696, 2018.

[39] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *CVPR*, pages 322–330, 2019.

[40] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A light CNN for deep face representation with noisy labels. *IEEE Trans. Information Forensics and Security*, 13(11):2884–2896, 2018.

[41] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, pages 2691–2699, 2015.

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

[43] Kun Yi and Jianxin 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.

[44] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *ICML*, 2019.

[45] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[46] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, pages 8778–8788, 2018.