Robust Learning with Noisy Label Detection and Counterfactual Correction

Wenting Qi
Department of Computer Science
University at Albany, SUNY
Albany, New York, USA
wqi@albany.edu

Charalampos Chelmis*†
Department of Computer Science
University at Albany, SUNY
Albany, New York, USA
cchelmis@albany.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10020228

***Abstract*—Data quality is of paramount importance in the training process of any machine learning model. Recently proposed methods for noisy learning focus on detecting noisy labeled data instances by using a fixed loss value threshold, and exclude detected noisy data instances in subsequent training steps. However, a predefined, fixed loss value threshold may not always be optimal, and excluding the detected noisy data instances can hurt the size of the training set. In this paper, we propose a new method, NDCC, that automatically selects a loss threshold to identify noisy labeled data instance, and uses counterfactual learning to repair them. To the best of our knowledge, NDCC is the first work to explore the feasibility of using counterfactual learning in the noisy learning domain. We demonstrate the performance of NDCC on Fashion–MNIST and CIFAR–10 datasets under a variety of label noise environments. Experimental results show the superiority of the proposed method compared to the state–of–the–art, especially in the presence of severe label noise.**

***Index Terms*—data quality, noisy learning, deep learning**

I. Introduction

Machine learning models have been applied in a wide range of applications, including, but not limited to, traffic prediction [1], face recognition [2] and product recommendation [3]. Deep neural networks have achieved remarkable performance to a variety of tasks due in part, to large quantities of human–annotated data [4]. However, the label annotation often introduces label noise. Furthermore, over–parameterized machine learning models, such as Deep Neural Networks, can overfit on noisy data instances by memorizing them during training [5], [6]. Learning and assessing machine learning models using noisy labels can result in biases and misleading accuracy reporting, with potentially detrimental results. There are two common types of noise, namely: feature noise and label noise [7]. In this work, we focus on label noise which has been shown to be more harmful than feature noise [8].

To train a model with a noisy dataset, one commonly adopted approach is noise sample selection [9], which distinguishes the noisy from clean data instances, then excludes noisy instances from the training process [10]. In line with prior art, this work leverages loss to distinguish between noisy from clean data instances (i.e., data instances exhibiting low loss value being more likely to be clean) [11]. The challenge is how to quantify the loss value during the training process. [12] ranks the loss value for all data instances and pre–sets the loss threshold with a specific noise rate (NR) to identify noisy data instances as those whose loss value is lower than the threshold. The main problem with that approach is twofold: (i) in the real–world, the noise rate is hard to estimate a priori, and (ii) different choices of loss functions result in different loss value rankings. To overcome these issues, we use peer loss [13] for noisy label detection. [13] sets peer loss threshold to 0 to distinguish the noisy from clean data instances. However, our experiments (See Figure 1) show that 0 may not always be the optimal peer loss threshold. We therefor propose an automated threshold selection method to overcome this issue.

Upon detecting suspected noisy labeled data instances, these instances are typically excluded from the training process [14]. However, for small or severely noisy labeled datasets, excluding noisy data can dramatically reduce the size of the training set, to the point it becomes useless for training purposes. Furthermore, despite having noisy labels, the feature values of noisy labeled data instances are clean and could still be useful for training. We believe this work to be the first to explore the feasibility of *correcting noisy labeled data instances by finding the true label* using counterfactual learning. Specifically, for each detected noisy labeled instance, counterfactual data instances are computed for all possible labels. The label that achieves the minimum value of counterfactual score is then selected as the true label (refer to Section IV-B for detailed explanation and examples).

This work focuses on *training a robust learning model in the presence of noisy labeled data in the training set, through detecting and correcting noisy labeled data instances.* In summary, the main contributions of this paper are:

- Proposing a novel method for automating the selection of the noisy peer loss threshold in the noisy label detection.
- Introducing a practical approach for identifying noisy labeled data in the training process, and estimating the

This material is based upon work supported by the National Science Foundation under Grant No. ECCS–1737443 and IIS–1850097.
*Corresponding author.
†Both authors contributed equally.

most probable true label for each detected noisy data instance using counterfactual learning.
- Demonstrating the superiority of the proposed solution against baselines using benchmark datasets under different noisy environments.

To ensure the reproducibility of our work, our method available at https://github.com/IDIASLab/NDCC.

II. Related Work

With the increase of complexity and scale of datasets, the possibility of including unreliable labels or noisy labels also increases. Training machine learning models with noisy labels significantly impacts their prediction performance. For this reason, a large variety of deep learning models for robust learning in noisy data environments has already been developed [15], [16]. For instance, the loss function–based approach in [15] minimizes the risk for unseen clean data with the presence of noisy labels in the training data. However, such loss function–based approaches are restricted to a particular framework, and thus, lack adaptability. [13] uses peer loss to select clean data instances by fixing the loss threshold to 0. However, the optimal loss threshold may not always be fixed or predetermined. Instead of using a fixed threshold, this work learns the loss threshold for noisy labeled data instances detection during the training process itself.

After detecting suspected noisy labeled data instances, many methods (e.g., [16]) exclude such instances in subsequent training steps. However, dropping suspected noisy label data instances can result in a diminished training set, and wastes the clean features of noisy labeled samples. [17] assigns more weight on clean data instances than on suspected noisy data instances. At the same time, mistreating noisy data instances as clean can lead to a highly inaccurate model. We instead propose a counterfactual based method to correct the labels of suspected noisy labeled data instances. Counterfactual learning has been widely explored in explainable machine learning. Specifically, [18] leverages counterfactual methods to produce example–based explanations by feature perturbation. Feature perturbation may lead to different prediction results given a learning model; data instances with perturbed feature values (in our case labels) are considered counterfactual [19]. To the best of our knowledge, this work is the first to incorporate counterfactual learning into noisy learning.

III. Preliminaries

Let $D = (X, Y)$ denote a clean training dataset and $\tilde{D} = (X, \tilde{Y})$[1] a noisy dataset. N is the total number of data instances in D and $\tilde{D}$ (i.e., $X = \{\mathbf{x}_i\}_{i=1}^{N}$), and $\mathbf{x}_i \in X$ is an M dimensional feature vector. The total number of classes in both Y and $\tilde{Y}$ are K, and j denotes the class index. The label of $\mathbf{x}_i$ is denoted as $\mathbf{y}_i \in \mathbb{B}^K$ with value 1 at entry j indicating belonging to the jth class, otherwise 0. For example, for $K = 5$, $\mathbf{y}_i = [0, 1, 0, 0, 0]$ indicates that $\mathbf{x}_i$ belongs to Class 2. The task is to train a model f using $\tilde{D}$, since the clean dataset D is unavailable, to predict the true label y of previously unseen data instances. Let $\bar{y}$ denote the predicted outcome. To minimize the influence of noisy data on the model performance, we propose strategies to detect noisy data instances, and assign them with the most likely true label while learning f. We leverage counterfactual learning to search for the most likely true label for each noisy data instance. Specifically, each noisy data instance is associated with K counterfactual data instances $(\hat{\mathbf{x}}_i^j, \hat{\mathbf{y}}_i^j)$, each is generated for each labels $\hat{\mathbf{y}}_i^j$, where $j \in 1, 2, 3, .., K$. By comparing the counterfactual properties (see Section IV-B) with $(\mathbf{x}_i, \tilde{\mathbf{y}}_i)$ and each $(\hat{\mathbf{x}}_i^j, \hat{\mathbf{y}}_i^j)$, we find the most likely true label $\hat{\mathbf{y}}_i^j$ and substitute the noisy label with the most likely true label $\hat{\mathbf{y}}_i^j$.

[1]Data instances in $\tilde{D}$ are either clean or noisy labeled. Same with $\tilde{Y}$.

IV. Proposed Framework

We propose Noisy label Detection and Counterfactual Correction (NDCC), a novel framework for training a robust classifier over a noisy labeled dataset. The objective function of NDCC follows:

$$\arg\min_{\mathbf{W}, \hat{\mathbf{x}}_i^j} \sum_{i=1}^{N} \phi_i h_c(\mathbf{W}, \mathbf{x}_i) + (1 - \phi_i) h_n(\mathbf{W}, \hat{\mathbf{x}}_i^j), \quad (1)$$

where

$$h_c(\mathbf{W}, \mathbf{x}_i) = l(f(\mathbf{W}, \mathbf{x}_i), \tilde{y}_i) - \frac{1}{K} \sum_{j=1}^{K} l(f(\mathbf{W}, \mathbf{x}_i), y_i^j), \quad (2)$$

and

$$\begin{aligned} & h_n(\mathbf{W}, \hat{\mathbf{x}}_i^j) = dist(\mathbf{x}_i, \hat{\mathbf{x}}_i^j), \\ & s.t. \quad f(\mathbf{W}, \hat{\mathbf{x}}_i^j) = j, \end{aligned} \quad (3)$$

ϕ is the noisy labeled data instanced indicator, and $dist$ denotes Euclidean distance. The detailed explanations for Eqs. (2) and (3) are provided in Sections IV-A and IV-B, respectively. Overall, the problem in Eq. (1) can be viewed as a combinatorial optimization problem, which is difficult to solve directly. We therefore solve Eq. (1) by alternatively searching for the optimal solutions of $\mathbf{W}$ and $\hat{\mathbf{x}}_i^j$.

Initially, the noisy dataset $\tilde{D} = (X, \tilde{Y})$ is provided as input to the noisy label detection, which then outputs suspected noisy label data instances $(X_n, \tilde{Y})$, and sets the noisy indicator $\phi = 0$ for each $\mathbf{x}_i \in X_n$, and 1 for each $x_i \in X_c$. Therefore, $\phi_i h_c(\mathbf{W})$ in Equation (1) reflects the loss for clean data instances, whereas $(1 - \phi_i) h_n(\mathbf{W}, \hat{\mathbf{x}}_\mathbf{i})$ reflects the loss for noisy labeled data instances. The label counterfactual correction module assigns each $\mathbf{x}_i \in X_n$ with the most likely true label $\hat{\mathbf{y}}_i$, then substitutes $\tilde{D}$ with the label–revised dataset $\hat{D}$, to be used in subsequent rounds of training $f(W)$. Note that $\hat{D}$ can be updated multiple times through the training process, as additionally noisy labeled data instances are identified.

A. Noisy Label Detection (h_c)

Loss can identify noisy labeled data instances [10], [11]. Specifically, [10] pointed out that the loss of clean data instances is expected to be lower than that of noisy labeled data instances, mainly because noisy labeled data instances are

often outliers with respect to the distribution of clean data, and the learning model tends to make predictions different from the noisy labels.

The question then is how to determine a loss threshold to distinguish between clean and noisy labeled data instances. Of particular relevance to this problem, [20] proposed peer loss defined $L_{PL} = l(f(\mathbf{W}, \mathbf{x}_i), \mathbf{y}_i) - l(f(\mathbf{W}, \mathbf{x}_i), \mathbf{y}_i^j)$, where $l(f(\mathbf{W}, \mathbf{x}_i), \mathbf{y}_i)$ is the loss with respect to given label $\mathbf{y}_i$, and $l(f(\mathbf{W}, \mathbf{x_i}), \mathbf{y}_i^j)$ is the loss with respect to a possible random label $\mathbf{y}_i^j$ differing from $\mathbf{y}_i$. Based on the peer loss, [13] defined the loss value threshold, which takes all possible label values into consideration in order to locate data instances with high loss for further distinguishing the noisy labeled data instances. Inspired by this idea, the objective function for detecting noisy labeled data instances is defined as [13]:

$$h_c(\mathbf{W}, \mathbf{x}_i) = l(f(\mathbf{W}, \mathbf{x_i}), \tilde{\mathbf{y}}_i) - \frac{1}{K}\sum_{j=1}^{K}(l(f(\mathbf{W}, \mathbf{x}_i), \mathbf{y}_i^j), \quad (4)$$

where f is the learning model, with parameters $\mathbf{W}$, $l(f(\mathbf{W}, \mathbf{x}_i), \tilde{\mathbf{y}}_i)$ denotes the loss value of the observed label, and $\frac{1}{K}\sum_{j=1}^{K}(l(f(\mathbf{W}, \mathbf{x}_i), \mathbf{y}_i^j)$ is the average loss value of all possible K labels.

1) Auto Noisy Threshold Selection Criterion: After computing h_c, the following question is how to use it to detect noisy labeled data instances. [13] sets 0 as the loss threshold to distinguish the clean and the noisy labeled data instances. Specifically, data instances whose $h_c \geq 0$ are considered to be noisy labeled. This is because the loss of the observed label $\tilde{y}_i$ is larger than the average loss of the other possible labels y_i^j [13]. However, 0 need not be the optimal loss value threshold. For instance, Figure 1 shows the peer loss of $1,000$ randomly selected data instances in CIFAR–10, under symmetric noise (NR = 0.1) (i.e., symmetric[2] and asymmetric noise[3]). The red dot line (peer loss threshold of 0) is evidently not optimal – the black dot line can detect more noisy labeled data instances.

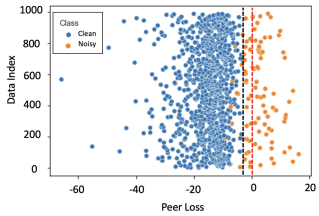

Fig. 1. Peer loss value distribution for random selected 1,000 data instances in CIFAR–10 under symmetric noise (NR = 0.1). $x-$axis corresponds to peer loss score, and $y-$axis corresponds to each data instance.

This work proposes to automate the peer loss threshold selection. Specifically, we wish to select noisy labeled data instances whose loss is large but not exactly larger than its average label loss threshold, as shown in Figure 1.

[2]The true label flips to all other labels with equal probability.

[3]A noisy label is generated by flipping the true label j to class $j+1$ [13].

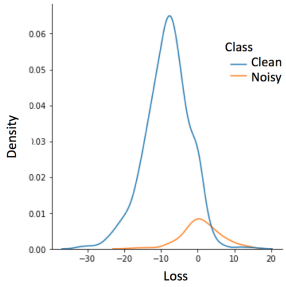

Fig. 2. Peer loss value distribution without pre–trained model with random selected 1,000 data instances in CIFAR–10 under symmetric noise (NR = 0.1). $x-$axis corresponds to peer loss score, and $y-$axis corresponds to frequency with respect to particular peer loss score in $x-$axis.

Before elaborating our proposed method, we note that using a randomly initialized deep neural network as a starting point can lead to erroneous loss estimation. For instance, Figure 2 shows that the loss of clean and noisy data instances may overlap. Erroneous loss estimation can lead to missdetection of clean instances as noisy (and visa versa), introducing even more noisy labeled data instances into the training dataset. Therefore, the starting point of a classification model is crucial. Inspired by [21], which showed that a small portion of clean labels improves the model robustness in noisy detection, we pre–train a model g (see Section V-A3 for a detailed discussion on g), using a small portion of data instances, denoted as D_{pre}, in which labels are guaranteed to be accurate. In the real–world, a small portion of clean data instances can be obtained using pre–annotation by experts [22].

We leverage this small portion of clean data instances to auto–detect and revise noisy data instances in the overall training set. Specifically, we first calculate h_c of each data instance in D_{pre} using g and denote it as l_{pc}. Next, since having knowledge of the type of noise present in the training dataset is unrealistic, we randomly select 10% of the data instances in D_{pre}, and artificially introduce noise by randomly switching their label to a different one. The noisy version of the pre–train dataset is denoted as $\tilde{D}_{pre} = \tilde{D}^c_{pre} \cup \tilde{D}^n_{pre}$, where $\tilde{D}^c_{pre}$ ($\tilde{D}^n_{pre}$) is the set of clean (noisy) data in $\tilde{D}_{pre}$. Next, we calculate h_c on $\tilde{D}_{pre}$ and record the loss as l_{pn}. The difference between l_{pc} and l_{pn} (i.e., $l_{diff} = l_{pc} - l_{pn}$) is used to define the loss varying area $\tilde{D}_{ns} = \{\mathbf{x}_i | l_{diff}(\mathbf{x}_i) \leq \min_{\mathbf{x}_q \in \tilde{D}^c_{pre}} l_{diff}(\mathbf{x}_q), \forall \mathbf{x}_i \in \tilde{D}_{pre}\}$. The rationale for calculating $\tilde{D}_{ns}$ is that $\tilde{D}_{ns}$ may contain the majority of noisy data instances, since l_{diff} is smaller for noisy data instances compared with clean data instances because of higher l_{pn}. As illustrated by Figure 3, the absolute value of the loss difference l_{diff} for noisy data instances is higher than the clean data instances.

In subsequent steps in the training process (i.e., without using $\tilde{D}_{pre}$), we have no prior indication about which data instances are clean or noisy. Our experiments indicate that noisy data instances are more likely to reside in $\tilde{D}_{ns}$, in a real training experiment with $\tilde{D}$. This observation lets us estimate the peer loss threshold by calculating the average loss, as

follows:

$$thr = \frac{1}{|\tilde{D}_{ns}|} \sum_{i} h_c(\mathbf{W}_g, \mathbf{x}_i), \mathbf{x}_i \in \tilde{D}_{ns}, \quad (5)$$

where $\mathbf{W}_g$ denotes the parameters of the clean pre–trained model g. The initial starting threshold thr is set to 0, as per [13].

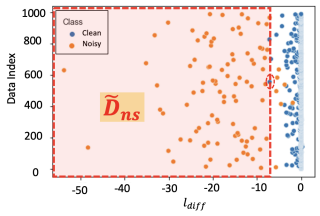

Fig. 3. Pre–train experiment with $\tilde{D}_{pre}$. $x-$axis corresponds to l_{diff}, and $y-$axis corresponds to each data instance. The red circled data instances define the upper bound of $\tilde{D}_{ns}$.

B. Noisy Label Correction h_n

The noisy label correction process is designed to pair the noisy labeled data instances with their most likely true label using counterfactual learning. Counterfactual learning is used to explain algorithmic decisions by feature perturbation [19], [23]. This work generates a counterfactual data instance with other possible labels for each detected noisy labeled data instance $(\mathbf{x}_i, \tilde{\mathbf{y}}_i) \in X_n$. Specifically, the noisy label detection module in IV-A provides the loss for each data instance.

The following question is how to generate the counterfactual data instances for a detected noisy labeled data instance. One commonly used counterfactual generation criterion is the **Proximity Score** [23] which evaluates the distance between the counterfactual data $\hat{\mathbf{x}}_i^j$ and the original feature vector $\mathbf{x}_i$. A smaller distance between the data instance $\mathbf{x}_i$ and its counterfactual data instance $\hat{\mathbf{x}}_i^j$ represents a higher probability that the true label of $\mathbf{x}_i$ is the target class[4] j for $\hat{\mathbf{x}}_i^j$. The proximity measure has the following form:

$$h_n = dist(\hat{\mathbf{x}}_i^j, \mathbf{x}_i), \quad (6)$$

where $dist$ denotes Euclidean distance. This work first selects the data instance with the minimum loss value (i.e., highest confidence of correct classification) as the counterfactual starting point (i.e., $\hat{\mathbf{x}}_2^A$, $\hat{\mathbf{x}}_2^B$, and $\hat{\mathbf{x}}_2^C$) for each possible label, as illustrated in Figure **??** step–1. Next, we minimize the proximity score by perturbing the feature values of $\hat{\mathbf{x}}_i^j$ (i.e., $\hat{\mathbf{x}}_2^A$, $\hat{\mathbf{x}}_2^B$, and $\hat{\mathbf{x}}_2^C$) and forcing it to get closer to the target noisy data instance. However, without any limitation, $\hat{\mathbf{x}}_i^j$ will eventually be equal to $\mathbf{x}_i$, achieving a proximity score of 0. To tackle this issue, we add a stopping criterion for the counterfactual data instance generation process by using a validity sore. Specifically, **Validity Score** [18] measures the degree of validity of a counterfactual data instance. A higher validity value represents higher confidence (e.g., a lower loss) of the predictor outputting the target label $\hat{\mathbf{y}}_i^j$ for the counterfactual data instance $\hat{\mathbf{x}}_i^j$, with $\hat{\mathbf{x}}_i^j$ being absolutely valid if the prediction outcome is the same as the target label (i.e., $f(\hat{\mathbf{x}}_i^j) = \hat{\mathbf{y}}_i^j$). In our work, we take the validity score as our stopping criterion and set it as 1 (i.e., highest value) to guarantee the generated counterfactual data instance $\hat{\mathbf{x}}_i^j$ belongs to the particular class j.

[4]In counterfactual learning, the generated data instance can be classified into a particular class (i.e., target class) by the learning model.

C. NDCC Algorithm

Initially, a pre–trained model g is used to generate $\tilde{D}_{ns}$ for automatically selecting the loss threshold in each following learning round. Then, potentially noisy labeled data instances in $\tilde{D} = (X, \tilde{Y})$ are detected. The most likely true label for each noisy data instance is determined using counterfactual label correction, and the dataset is updated with the revised labels. The revised dataset is used to train model f. The noisy loss threshold for the next iteration is determined using the trained model f. The algorithm terminates when the maximum training epoch, T, is reached, or the dataset is no longer updated.

V. Evaluation

A. Experiment Setting

1) Datasets: We evaluate the proposed NDCC framework on two widely used benchmark datasets [24].

CIFAR–10 [25]: Image dataset in the CIFAR family. The size of the training set and the test set are $50,000$ and $10,000$. Each data instance is a $32 \times 32 \times 3$ colorful image, associated with 10 classes (i.e., airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). **Fashion–MNIST [26]:** Real–world image dataset collected from Zalando's article. The training set contains $60,000$ data instances and the test set contains $10,000$ data instances. Each data instance is a 28×28 grayscale image, associated with a label from 10 classes (i.e., t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot).

2) Noise Environments: Both benchmark datasets are clean, or with a negligible number of noisy labeled data instances [14]. To evaluate the effectiveness of NDCC in noisy labeled environments, we consider two types of noise: *symmetric* and *asymmetric*. In both cases, τ denotes the noise rate. We consider $\tau \in \{0.2, 0.4, 0.6, 0.8\}$ to evaluate NDCC on scenarios involving a variable number of noisy labeled data instances (ranging from small to large).

3) Experimental Setup: All experiments use ResNet34 and the following hyper–parameter values: mini–batch size (32), number of training epochs (90), optimizer (AdamW [27]), learning rate (0.01). In NDCC, we set $T = 3$ and $T_n = 30$ to ensure that the overall number of training epochs for NDCC is the same as with the baseline methods (i.e., 90). The counterfactual training epoch T_{cf} is set to 50. In both CIFAR–10 and Fashion–MNIST experiments, we randomly select $2,000$ data instances as the clean pre–trained dataset D_{pre}, and use D_{pre} to train g. Among the rest of the data instances, we randomly select $10,000$ data instances as the training set D. We use the default test set for both CIFAR–10 and Fashion–MNIST.

TABLE I
TRUE DETECTION RATE X_{dt} (HIGHER IS BETTER)

Method/NS Environment (τ)	Fashion–MNIST								CIFAR–10							
	Sym				Asym				Sym				Asym			
	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
CORES	0.56	0.55	0.57	0.59	0.51	0.50	0.56	0.57	0.60	0.58	0.61	0.59	0.64	0.61	0.62	0.57
NDCC	**0.76**	**0.75**	**0.75**	**0.77**	**0.67**	**0.68**	**0.72**	**0.74**	**0.80**	**0.81**	**0.81**	**0.84**	**0.78**	**0.76**	**0.78**	**0.82**

TABLE II
COUNTERFACTUAL TRUE CORRECTION RATE $\hat{X}_{cf_c}$ (HIGHER IS BETTER)

Method/NS Environment	Fashion–MNIST								CIFAR–10							
	Sym				Asym				Sym				Asym			
	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
NN-correction	0.24	0.11	0.04	0.01	0.19	0.12	0.05	0.02	0.28	0.13	0.04	0.04	0.22	0.09	0.04	0.01
NDCC	**0.62**	**0.61**	**0.59**	**0.60**	**0.57**	**0.55**	**0.54**	**0.56**	**0.68**	**0.70**	**0.72**	**0.71**	**0.67**	**0.65**	**0.68**	**0.70**

TABLE III
DECREASED NOISY RATE d_τ (LOWER IS BETTER)

Method/NS Environment	Fashion–MNIST								CIFAR–10							
	Sym				Asym				Sym				Asym			
	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
CORES	0.07	0.16	0.22	0.23	0.06	0.14	0.19	0.23	**0.08**	**0.17**	0.22	0.24	**0.09**	**0.18**	0.21	0.23
NN-correction	0.02	0.01	-0.03	-0.02	0.01	0.01	-0.03	-0.02	0.01	0.03	-0.03	-0.01	0.02	-0.02	-0.02	-0.02
NDCC	**0.08**	**0.21**	**0.30**	**0.38**	**0.06**	**0.18**	**0.27**	**0.35**	0.06	**0.17**	**0.29**	**0.44**	0.07	0.16	**0.26**	**0.42**

TABLE IV
EXPERIMENTS RESULT OF TEST ACCURACY.

Method/NS Environment	Fashion–MNIST								CIFAR–10							
	Sym				Asym				Sym				Asym			
	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
CE	0.63	0.49	0.28	0.11	0.58	0.42	0.27	0.12	0.52	0.41	0.26	0.12	0.59	0.43	0.27	0.12
CORES	0.71	0.65	0.56	0.39	0.75	0.69	0.64	0.48	0.67	0.62	0.52	0.41	0.69	**0.65**	0.56	0.42
AUM	**0.75**	**0.69**	0.58	0.35	**0.79**	**0.71**	0.63	0.41	**0.68**	**0.63**	0.55	0.37	**0.71**	**0.65**	0.57	0.36
NN-Correction	0.63	0.45	0.22	0.10	0.60	0.43	0.25	0.12	0.54	0.40	0.19	0.09	0.60	0.37	0.21	0.09
NDCC	0.72	0.65	**0.59**	**0.52**	0.75	0.70	**0.65**	**0.54**	0.65	0.60	**0.56**	**0.49**	0.67	0.63	**0.58**	**0.51**
CE-Clean	0.78	0.76	0.69	0.64	0.81	0.77	0.72	0.65	0.70	0.66	0.60	0.55	0.73	0.69	0.64	0.57

4) Baselines: **CE (Cross Entropy)** uses cross entropy loss, and has no particular strategy for handling noisy labeled data instances. **CE–Clean** uses solely clean data instances for training, and thus achieves the theoretical best performance. **CORES (Confidence Regularized Sample Sieve) [13]** uses peer loss to detect suspected noisy labeled data instances without unsupervised training. **AUM (Area Under the Margin) [14]** uses the AUM statistic to exploit the differences between the clean and noisy labeled data instances. AUM excludes the detected noisy data instances from the training process. **NN–Correction (Nearest neighbor noisy label correction) [28]** uses the same noisy detection module as NDCC, but noise label correction is performed using k–nearest neighbors.

5) Evaluation Metrics: We divide the evaluation process into three parts: (i) noise detection, (ii) noise correction, and (iii) overall accuracy on the clean test set under different types of label noise in the training set.

Recall X_n denotes the accumulated detected noisy data set with respect to all learning rounds T, and $\tilde{D}$ is the noisy input data set. Let $X_{\tilde{D}}$ denote the true noisy data set. We introduce the following score to evaluate noise detection performance: (i) **True detection rate**: $X_{dt} = \frac{|X_n \cap X_{\tilde{D}}|}{|X_{\tilde{D}}|}$ measures the ratio of truly identified noise data instances; (ii) **Miss detection rate**: $X_{dm} = \frac{(|\tilde{D} - X_n) \cap X_{\tilde{D}}|}{|X_{\tilde{D}}|}$ measures the ratio of noisy data instances identified as clean. In Section V-B, we only discuss X_{dt}, since $X_{dm} = 1 - X_{dt}$.

In noise correction, we check whether NDCC can correctly assign the true labels to corresponding detected noisy data instances. Let $\hat{X}_r$ denote the data set where NDCC correctly pair detected noisy data instances with their true labels, and $\hat{X}_w$ denote the detected noisy data instances that are assigned wrong labels. We define the **True counterfactual label correction rate** $\hat{X}_{cf_c} = \frac{|\hat{X}_r|}{|X_n|}$.

Finally, we measure the **decreased noisy labeled rate** d_τ after applying the noisy label detection of baselines and NDCC, and test the accuracy of each trained learning model f. d_τ for CORES is computed as $d_\tau = \tau - \frac{|X_{\tilde{D}}|X_{dm}}{|\tilde{D}| - |X_n|}$, where $|\tilde{D}| - |X_n|$ denotes the number of currently available training data instance, excluding the detected noisy data instances, and $|X_n|X_{wd} + |X_{\tilde{D}}|X_{dm}$ denotes the remaining miss detected noisy data instances. d_τ for NDCC and NN–Correction is defined as:$d_\tau = \tau - \frac{|X_{\tilde{D}}|X_{dm} + |X_n|\hat{X}_{cf_w}}{|\tilde{D}|}$, where $|X_n|\hat{X}_{cf_w}$ is seen as noisy data instance because of correction failure.

B. Experiments Results

1) Noisy Label Detection: We begin by comparing NDCC and CORES. Table I shows the true detection rates X_{dt}, for CORES and NDCC. A larger value of X_{dt} indicates better performance, as the goal is to detect as many true noisy labeled data instances as possible. Compared with CORES,

NDCC's true detection rate X_{dt} is obvious higher, illustrating that automatically selecting the loss threshold is beneficial, as opposed to using a fixed threshold, as in [13].

2) Noisy Label Correction: We next measure the effectiveness of NDCC's counterfactual label correction module by comparing the label correction results between NN–Correction and NDCC. Table II shows that, for both Fashion–MNIST and CIFAR–10, NDCC's $\hat{X}_{cf_c}$ is much higher than NN–Correction, especially as τ increases. The performance of NN–Correction is unsatisfactory because clusters become unreliable in the presence of noisy labeled data instances. Instead, NDCC's superiority is confirmed with a stable $\hat{X}_{cf_c}$ score, even in severe noisy environments (i.e., $\tau = 0.6, 0.8$). Finally, Table III shows the decreased noisy rate that different methods achieve. NDCC outperforms all baselines in all noisy environments across both datasets.

3) Overall Evaluation: Table IV shows the accuracy of NDCC and the baselines. CE, which does not at all perform noisy detection, is expected to be the least performing method. CE–Clean intentionally uses only clean data instances for training, and is therefore expected to perform ideally. For both Fashion–MNIST and CIFAR–10, NDCC outperforms the baselines when noise becomes severe (i.e., $\tau \geq 0.6$) in both asymmetric and asymmetric case.

VI. Conclusion

We presented a new method for robust learning in the presence of noisy labeling data. Specifically, we proposed an automatic noisy peer loss threshold selection method to separate noisy labeled data instances from clean data instances. We additionally proposed to leverage counterfactual learning to correct detected noisy labeled data instances. Our experimental results show the superiority of the proposed approach as compared with the state–of–the–art, particularly in severe label noise environments.

In future work, we wish to reduce our method's dependency on a pre–trained model with carefully labeled training data. Even though this is a commonly adopted strategy in noisy learning, we believe that eliminating the need for manual annotation and human inspection can benefit noisy learning by allowing models to be trained on less circumscribed domains.

References

[1] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 5668–5675.

[2] G. Guo and N. Zhang, "A survey on deep learning based face recognition," *Computer vision and image understanding*, vol. 189, p. 102805, 2019.

[3] H. Tuinhof, C. Pirker, and M. Haltmeier, "Image-based fashion product recommendation with deep learning," in *International Conference on Machine Learning, Optimization, and Data Science*. Springer, 2018, pp. 472–481.

[4] M. M. Kamani, S. Farhang, M. Mahdavi, and J. Z. Wang, "Targeted data-driven regularization for out-of-distribution generalization," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 882–891.

[5] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[6] D. Arpit, S. Jastrzundefinedbski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 233–242.

[7] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2013.

[8] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artificial intelligence review*, vol. 22, no. 3, pp. 177–210, 2004.

[9] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[10] W. Shin, J.-W. Ha, S. Li, Y. Cho, H. Song, and S. Kwon, "Which strategies matter for noisy label classification? insight into loss and uncertainty," *arXiv e-prints*, pp. arXiv–2008, 2020.

[11] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," *arXiv preprint arXiv:1804.06872*, 2018.

[12] N. M. Müller and K. Markert, "Identifying mislabeled instances in classification datasets," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[13] H. Cheng, Z. Zhu, X. Li, Y. Gong, X. Sun, and Y. Liu, "Learning with instance-dependent label noise: A sample sieve approach," in *International Conference on Learning Representations*, 2021.

[14] G. Pleiss, T. Zhang, E. Elenberg, and K. Q. Weinberger, "Identifying mislabeled data using the area under the margin ranking," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 044–17 056, 2020.

[15] A. Ghosh, H. Kumar, and P. S. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[16] E. Malach and S. Shalev-Shwartz, "Decoupling" when to update" from" how to update"," *Advances in neural information processing systems*, vol. 30, 2017.

[17] J. Cao, S. Kwong, and R. Wang, "A noise-detection based adaboost algorithm for mislabeled data," *Pattern Recognition*, vol. 45, no. 12, pp. 4451–4465, 2012.

[18] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[19] W. Qi and C. Chelmis, "Improving algorithmic decision–making in the presence of untrustworthy training data," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 1102–1108.

[20] Y. Liu and H. Guo, "Peer loss functions: Learning from noisy labels without knowing noise rates," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6226–6236.

[21] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 839–847.

[22] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," *Advances in neural information processing systems*, vol. 31, 2018.

[23] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 607–617.

[24] Y. Heng, Z. Gao, Y. Jiang, and X. Chen, "Exploring hidden factors behind online food shopping from amazon reviews: A topic mining approach," *Journal of Retailing and Consumer Services*, vol. 42, pp. 161–168, 2018.

[25] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

[26] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification." *Journal of machine learning research*, vol. 10, no. 2, 2009.