Noisy Label Detection and Counterfactual Correction

Wenting Qi, *Student member, IEEE*, Charalampos Chelmis, *Member, IEEE*

***Abstract*—Data quality is of paramount importance to the training 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 need not be optimal, and excluding the detected noisy data instances can hurt the size of the training set. In this article, we propose NDCC, a new method that automatically selects a loss threshold to identify noisy labeled data instances, and uses counterfactual learning to repair them. To the best of our knowledge, NDCC is the first work to explore the use of 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.**

***Impact Statement*—The accuracy of machine learning models depends on training data quality. Quite unsurprisingly then, it drops dramatically (up to 53% in our experiments) as the percentage of noisy labels increases. The method presented here is shown to maintain high performance even in the presence of highly corrupted data (i.e., 80% noisy labels) by performing joint noisy detection and correction. Specifically, the proposed method increases the accuracy rate of noisy label detection (up to 25%), while achieving a high noisy correction rate (up to 72%). When presented with severe label noise (i.e., 80% noisy labels), the proposed method lowers the noise rate to 52.5%. Beyond improving the accuracy of machine learning models that are trained with noisy label data, this research highlights the need to treat (as opposed to discard) noisy label instances during the training process.**

***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], product recommendation [3] and online fraud detection [4]. Deep neural networks, one of the most popular branches of machine learning, have achieved remarkable performance to a variety of tasks due in part, to large quantities of human–annotated data [5], [6]. However, the label annotation process is labor–intensive, and often introduces label noise for reasons including insufficient information for low quality data, subjectivity in the labeling process, and limited number of expert annotators due to budgetary constraints [7]. After the completion of the data labeling process, identifying and correcting wrong labels is resource– and time–consuming. Furthermore, over–parameterized machine learning models, such as Deep Neural Networks, can overfit on noisy data instances by memorizing them during training [8], [9]. Learning and assessing machine learning models using noisy labels can result in biases and misleading accuracy reporting, with potentially detrimental results, such as wrong disaster diagnosis [10] or perpetuating biases in resource allocation (e.g., loan application) [11]. There are two common types of noise, namely: feature noise and label noise [12]. In this work, we focus on label noise which has been shown to be more harmful than feature noise [13].

To facilitate training a learning model over a noisy dataset, one commonly adopted approach is noise sample selection [14], which distinguishes the noisy from clean data instances during the training process, then excludes noisy instances from the training process [15]–[17]. 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) [18], [19]. The challenge is how to quantify the loss value during the training process. [20] 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 have different impacts on the loss value ranking. To overcome these issues, we use peer loss [21] in loss value evaluation for noisy label detection. Specifically, peer loss is the loss value computed by substituting the current label with other possible labels in the label set, and does not require knowledge of the noise rate. Furthermore, since the comparison is among the same data instance with different label values, different loss functions do not affect the comparison result. [21] sets peer loss threshold to 0 to distinguish the noisy from clean data instances. However, our experiments (See Figure 4) show that 0 may not always be the optimal peer loss threshold. This article proposes 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 [22]. 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. This work is the first to explore the

Manuscript received (date to be filled by Editor). This material is based upon work supported by the National Science Foundation.

C. Chelmis is with the Department of Computer Science, University at Albany, SUNY, NY, 12222 USA (e-mail: cchelmis@albany.edu).

W. Qi is with the Department of Computer Science, University at Albany, SUNY, NY, 12222 USA (e-mail: wqi@albany.edu).

This paragraph will include the Associate Editor who handled your paper.

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.* A new framework is proposed to (i) identify potentially noisy labeled data instances in the training set, (ii) estimate the true label of each detected noisy labeled data instance through counterfactual data generation, and (iii) output a robust learning model and revised dataset (i.e., with corrected labels). We evaluate the ability of the proposed framework to handle varying degrees of noisy labeled data using two benchmark datasets. In summary, the main contributions of this article 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 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, we will make the source code of our method available on GitHub upon acceptance of this manuscript.

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 [23], [24]. For instance, the loss function–based approach in [23] 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. Some methods (e.g., [16], [21], [22], [24]) focus on selecting the true labeled instances from a noisy labeled dataset to mitigate the negative influence of noisy data instances. For instance, [21] 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], [22], [24]) 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 to shed light into how/why the output of a machine learning model would change if the input (i.e., features) were to change [25], [26]. Specifically, [27] leverages counterfactual learning 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 [28]. To the best of our knowledge, this work is the first to incorporate counterfactual learning directly into noisy learning.

III. Preliminaries and Problem Statement

A. Notation

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 $\hat{\mathbf{y}}_i^j$. Table I summarizes the notation used hereafter.

B. Problem Statement

The goal of this work is to learn a robust classifier f over a noisy labeled dataset by minimizing the influence of noisy labeled data instances during training. To achieve this goal, we split the problem into three sub–problems: (i) learn a classifier f that accurately maps X to Y, (ii) detect noisy labeled data instances, and (iii) assign the most likely true label to suspected noisy labeled data instances through counterfactual learning. For clarity, *clean* data instances refer to data instances with correct labels; *noisy* data instances refer to data instances with wrong labels; and observed labels can be either clean or noisy.

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

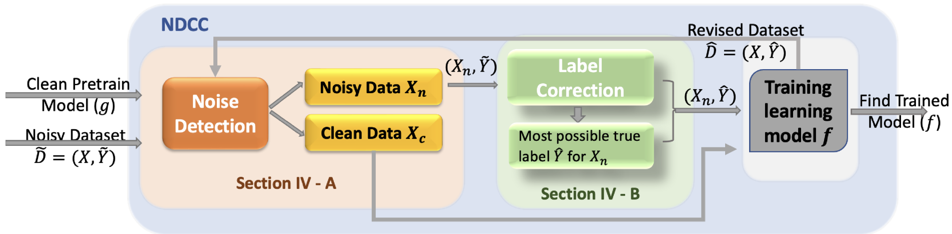


Fig. 1. Visualization of NDCC.

TABLE I
EXPLANATION OF MAIN SYMBOLS USED IN THIS ARTICLE.

Symbol	Description
N	Total number of data instances. (A data instance is denoted by index i)
M	Total number of features for each data instance. (A feature is denoted by index m)
K	Total number of classes
j	jth class
$\tilde{D}$	Noisy dataset
D_{pre}	Clean pretrained dataset in Algorithms 1 and 3
$\hat{D}$	Revised dataset
X_n/X_c	Detected noisy/clean dataset in Algorithm 3
h_c/h_n	Objective function for noise detection/correction
g	Pre–trained model obtained by training with D_{pre} in Algorithms 1 and 3
$f(\mathbf{W})$	Learning model with weight matrix $\mathbf{W} \in \mathbb{R}^{K \times M}$
l	Categorical cross entropy loss
$\hat{\mathbf{x}}_i^j$	Counterfactual data of $\mathbf{x}_i$ with target label j
$\hat{\mathbf{y}}_i^j$	Counterfactual label when considering target label j for $\mathbf{x}_i$
$\tilde{\mathbf{y}}_i$	Noisy label vector of $\mathbf{x}_i$
$\tilde{\mathbf{y}}_i^j$	Noisy label of $\mathbf{x}_i$ with label j
ϕ_i	Indicator of data instance i being clean or noisy
T_{pre}	Training epoch for pre–train model g in Algorithms 1 and 3
T_{cf}	Counterfactual search epoch in Algorithms 2 and 3
T_n	Training epoch in Algorithm 3 step 25
T	Training epoch for NDCC in Algorithm 3

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:

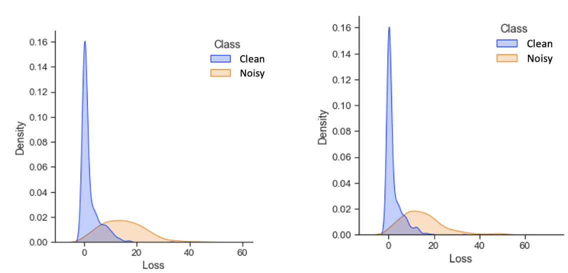
$$\arg\min_{\mathbf{W}, \hat{\mathbf{x}}_i^j} \sum_{i=1}^{N} \phi_i h_c(\mathbf{W}, \mathbf{x}_i) + (1 - \phi_i) h_n(\mathbf{W}, \hat{\mathbf{x}}_i^j). \quad (1)$$

ϕ is the noisy labeled data instanced indicator, and $dist$ denotes Euclidean distance. The detailed expression and explanations for h_n and h_c are provided in Sections IV-A and IV-B, respectively, and the rationale for combining the above two objective functions is provided in Section IV-C. 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 $\hat{\mathbf{x}}_i^j$. The convexity is proven in Appendix A. In each searching round, we accomplish the alternative search in two steps, as follows: (i) noisy label calibration (i.e., search solutions for $\hat{\mathbf{x}}_i^j$), which requires noisy label detection (Section IV-A) and label correction by counterfactual generation (Section IV-B), and (ii) model training (i.e., search solutions for $\mathbf{W}$).

Figure 1 provides a high level view of NDCC. Initially, the noisy dataset $\tilde{D} = (X, \tilde{Y})$ is provided as input to the noisy label detection module, 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 [15], [19], [29]. Specifically, [15] 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. Experiment results presented in Figure 2 support this claim by showing that the loss value for a large number of noisy labeled data instances is higher than that of clean data instances, even under different noisy environments (i.e., symmetric[2] and asymmetric noise[3]).



(a) Symmetric noise, NR = 0.4 (b) Asymmetric noise, NR = 0.4

Fig. 2. Loss value distribution for CIFAR–10 with respect to different noisy environments. x−axis represents the loss score, and y−axis represents the frequency of a particular loss score in the x−axis.

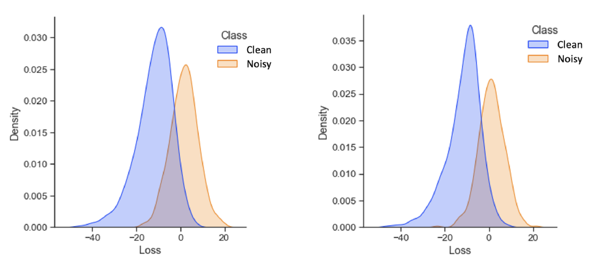
[2]The true label flips to all other labels with equal probability.
[3]A noisy label is generated by flipping the true label j to class $j + 1$ [21].

The question then is hoe to determine a loss threshold to distinguish between clean and noisy labeled data instances. A relative "large" or "small" loss can be manually specified by inspecting the overall loss value distribution. However, a classifier cannot automatically determine whether the loss score is "large" or "small" without knowing the overall loss value distribution. Furthermore, having a pre–set and fixed loss threshold is impractical, as the loss distribution may vary across different classification tasks. Additionally, the correct labels of noisy data instances are usually unavailable, making it impossible to pre–select a suitable loss threshold.

Of particular relevance to this problem, [30] 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, [21] 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 [21]:

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

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. The experimental results in Figure 3 confirms that the peer loss of the majority of the clean data instances is smaller than that of noisy labeled data instances.



(a) Symmetric noise, NR = 0.4 (b) Asymmetric noise, NR = 0.4

Fig. 3. Peer loss value (i.e., computed by Eq. (2)) distribution for CIFAR–10 with respect to different noisy environments. $x-$axis corresponds to peer loss score, and $y-$axis corresponds to frequency with respect to particular peer loss score in $x-$axis.

1) Auto Noisy Threshold Selection Criterion: After computing h_c, the following question is how to use it to detect noisy labeled data instances. [21] 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 [21]. However, 0 need not be the optimal loss value threshold. For instance, Figure 4 shows the peer loss of $1,000$ randomly selected data instances in CIFAR–10, under symmetric noise (NR = 0.1). The red dot line (peer loss threshold of 0) is evidently not optimal – the black dot line can detect more noisy labeled data instances.

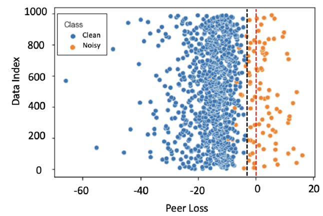


Fig. 4. Peer loss value distribution for random selected 1,000 data instances in CIFAR–10 under symmetric noise (NR = 0.1). $x-$axis corresponds to peer loss score, and $y-$axis corresponds to each data instance.

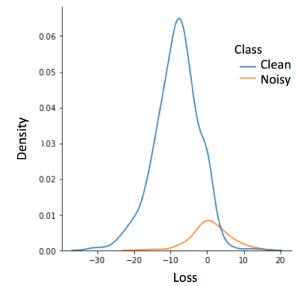


Fig. 5. Peer loss value distribution without pre–trained model with random selected 1,000 data instances in CIFAR–10 under symmetric noise (NR = 0.1). $x-$axis corresponds to peer loss score, and $y-$axis corresponds to frequency with respect to particular peer loss score in $x-$axis.

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 4.

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 5 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 [31], 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 [32].

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 6, 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. Figure 7 shows 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:

$$thr = \frac{1}{|\tilde{D}_{ns}|}\sum_i h_c(\mathbf{W}_g, \mathbf{x}_i), \mathbf{x}_i \in \tilde{D}_{ns}, \quad (3)$$

where $\mathbf{W}_g$ denotes the parameters of the clean pre–trained model g. The auto–learning noisy selection process is summarized in Algorithm 1, in which the initial starting threshold thr is set to 0, as per [21].

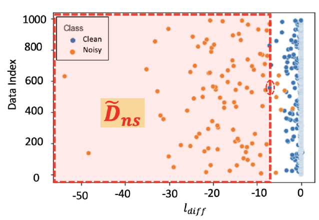
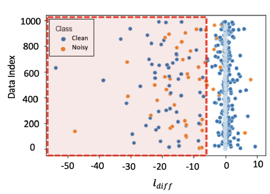
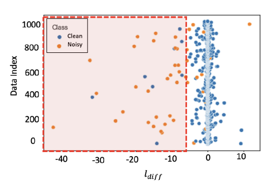


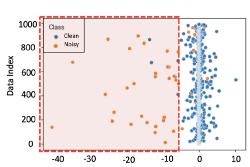
Fig. 6. Pre–train experiment with $\tilde{D}_{pre}$. $x-$axis corresponds to l_{diff}, and $y-$axis corresponds to each data instance. The red circled data instances define the upper bound of $\tilde{D}_{ns}$.



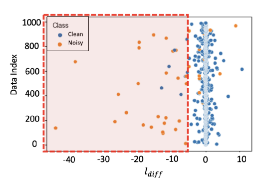
(a) First training round



(b) Second training round



(c) Third training round



(d) Fourth training round

Fig. 7. Test $\tilde{D}_{ns}$ on training simulation with $\tilde{D}$ in different learning rounds. $x-$axis corresponds to loss score, and $y-$axis corresponds to frequency with respect to particular loss score in $x-$axis. The training around corresponding to T in algorithm 3.

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 [27], [28], [33]. 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, as shown in Figure 8(a). In this example, data instance 2 has the highest loss of 0.9. Thus, $(\mathbf{x}_2, \tilde{\mathbf{y}}_2)$ is suspected to be noisy labeled. The true label $\mathbf{y}_2$ for $\mathbf{x}_2$ belongs to the label set $Y = \{A, B, C\}$. We consider the noisy label B as a viable label candidate because the noise detection module may make a wrong detection by treating clean data instances as noisy. Therefore, the target counterfactual data instances include $(\hat{\mathbf{x}}_2^{j=A}, \hat{\mathbf{y}}_2^{j=A})$, $(\hat{\mathbf{x}}_2^{j=B}, \hat{\mathbf{y}}_2^{j=B})$, $(\hat{\mathbf{x}}_2^{j=C}, \hat{\mathbf{y}}_2^{j=C})$.

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** [33] 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

Algorithm 1 $\tilde{D}_{ns}$ Computation

Input: Clean pre–trained model g, clean data subset D_{pre}, input noise rate τ_p, learning epoch T_{pre}

Select the number of $|D_{pre}| \times \tau_p$ data instances from D_{pre}, and randomly re–assign them with other labels which are different from their original ones.
Output the loss value l_{pc} by Eq. 2 with g
Training data instances in $\tilde{D}_{pre}$ with clean pre–train model g for T_p learning epochs and output the trained model $\tilde{g}$
Output the loss value l_{pn} by Eq. 2 with $\tilde{g}$
Compute $l_{diff} = l_{pc} - l_{pn}$
Set the smallest value of l_{diff} of the clean data instances as the loss value threshold for $\tilde{D}_{ns}$

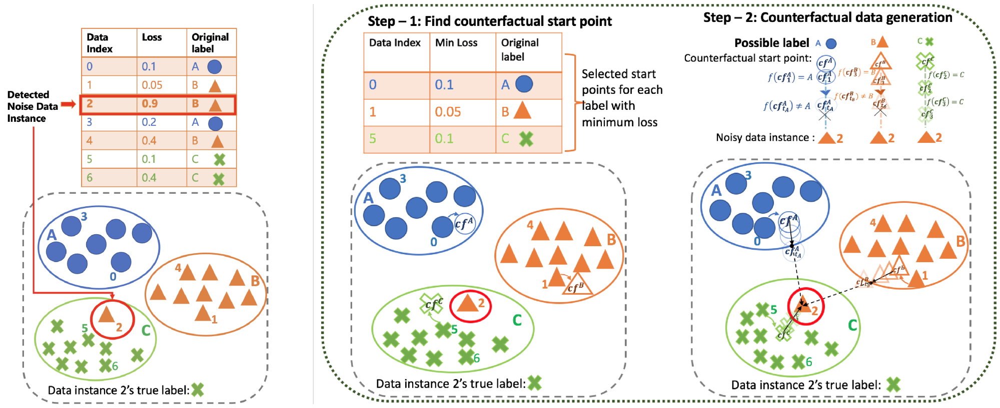
Output: $\tilde{D}_{ns}$

Algorithm 2 Counterfactual Data Generation

Input: $(x_i, \tilde{y}_i)$, target label set $Y = \{1, 2, ..., j, ...K\}$, T_{cf} maximum epoch number, learning model $f(\mathbf{W})$, and counterfactual starting point set $\{x^1_{cf_0}, x^2_{cf_0}..., x^j_{cf_0}, ..., x^K_{cf_0}\}$

Set optimal counterfactual set $X_{cf} = \emptyset$
for each $j \in Y$ **do**
  Set $x^j_{cf_0}$ as the counterfactual starting point and $t = 1$
  while $(f(W, x^j_{cf_t}) = \mathbf{y}^j_i$ and $t \leq T_{cf})$ **do**
    Optimize the loss using $x^j_{cf_t}$ and x_i based on Eq. (4)
    $t = t + 1$
  end while
  Return counterfactual data $x^t_{cf_j}$ that minimizes the loss of Eq. (4) as $\hat{\mathbf{x}}^j_i$
  Add $\hat{\mathbf{x}}^j_i$ into X_{cf}
end for

Output: Output $\hat{\mathbf{x}}^j_i$ in X_{cf} which achieves the minimal value of h_n, and the corresponding value of $h_n(\hat{\mathbf{x}}^j_i)$



(a) Example for loss value (b) Example for counterfactual label searching

Fig. 8. Diagram to show the steps of counterfactual noisy label correction.

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:

$$h_n = dist(\hat{\mathbf{x}}_i^j, \mathbf{x}_i), \tag{4}$$

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 8(b) 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** [27] 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. Taking label A in Figure 8(b) as an example, after perturbing the feature of $\hat{\mathbf{x}}_2^A$ for t_A times, we obtain the counterfactual data instance $\hat{\mathbf{x}}_{2_{t_A}}^A$, which triggers the stopping criterion because $f(\hat{\mathbf{x}}_{2_{t_A}}^A) \neq A$. The final output counterfactual data instance for label A is $\hat{x}_{2_{t_A-1}}^A$ (i.e., $f(\hat{x}_{2_{t_A-1}}^A) = A$). The same process is carried out for labels B and label C. After obtaining all the valid counterfactual data instances, the most likely true label for the noisy data instance $\mathbf{x}_i$ is determined to be the label of the counterfactual data instance with the smallest proximity score. The overall noisy label correction process is described in Algorithm 2.

[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

Algorithm 3 describes the process of learning a model from potentially noisy labeled data based on Eq. (1). Initially, a pre–trained model g is used to generate $\tilde{D}_{ns}$ for automatically selecting the loss threshold in each following learning round (step 2). Then, potentially noisy labeled data instances in $\tilde{D} = (X, \tilde{Y})$ are detected (steps $6-14$). 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 (steps $19-23$). The revised dataset (step 24) is used to train model f (step 25). The noisy loss threshold for the next iteration is determined using the trained model f (step $27-$ -29). The algorithm terminates when reaches the maximum training epoch T or the dataset is no longer updated.

V. Evaluation

A. Experiment Setting

1) Datasets: To measure the efficiency of the proposed NDCC framework, we evaluate it on two widely used benchmark datasets [34]–[36].

CIFAR–10 [37]: 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 [38]:** 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 [22]. To evaluate the effectiveness of NDCC in noisy labeled environments, we consider two types of noise: (i) **Symmetric Noise**: the true label flips to all other labels with equal probability. The symmetric noise simulates the label noise caused by a random mistake in the labeling process; (ii) **Asymmetric Noise**: a noisy label is generated by flipping the true label to the next class (i.e., label $i \leftarrow i+1; mod\ K$) [21]. In both cases, τ denotes the noise rate. We consider $\tau \in \{0.2, 0.4, 0.6, 0.8\}$

Algorithm 3 NDCC

Input: Noisy dataset $\tilde{D}$, pre–train clean dataset $\tilde{D}_{pre}$, pre–train clean model g, learning model f, pre–set learning epoch T_{pre}, maximum epoch for learning model T_n, maximum epoch for counterfactual searching T_{cf}, and learning round T

```
t ← 1, thr_1 ← 0, W_t, D̃^t ← D̃
Compute the noisy loss region D̃_ns using Algorithm 1
Calculate l_{pc_i} by Eq. 2 with g for each x_i ∈ D̃^t)
while (t ≤ T and D̃^t equals to D̃^{t−1}) do
    /*Noise Detection Section*/
    for x_i ∈ D̃^t do
        X_c^t, X_n^t ← ∅
        Calculate h_c(x_i, W_t) using Equation (2)
        if h_c(x_i, W_t) ≤ thr_t then
            Set φ_i^t ← 1, and add x_i into X_c^t
        else
            Set φ_i^t ← 0, and add x_i into X_n^t
        end if
    end for
    if X_n^t = X_n^{t−1} and t >= 2 then
        break
    end if
    /*Noise Correction Section*/
    Select data instances with minimum value of h_n for
    each label and set these as counterfactual starting points
    {x^1_{cf_0}, x^2_{cf_0} ..., x^j_{cf_0}, ..., x^K_{cf_0}}
    for Each x_i ∈ X_n^t do
        Output x̂_i^j and h_n(x̂_i^j) using algorithm 2
        Update label of x_i with j
    end for
    Get label revised dataset as D̃^{t+1}
    Train the learning model f(W_t) with updated D̃^t for
    T_n epochs by minimizing the cross entropy loss and
    output f(W_{t+1})
    /*Updating Loss Threshold*/
    Calculate l^t_{pn_i} by Eq. 2 with f(W_{t+1}) for x_i ∈ D̃^{t+1}
    Calculate l^t_{diff_i} = l_{pc_i} − l^t_{pn_i} for x_i in D̃^{t+1}
    Aggregate data instances whose l^t_{diff_i} is inside D̃_ns and
    compute thr_{t+1} by Eq. (3)
    t ← t + 1
end while
```

Output: $f(\mathbf{W}_T)$ and $\tilde{D}^T$

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 [39]), learning rate (0.01). In NDCC, we set $T = 3$ and $T_n = 30$ in Algorithm 3 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} in Algorithm 2 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 with the following hyper–parameters: mini–batch size (32), number of training epochs (50), optimizer (AdamW), learning rate (0.01). 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.

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) [21]** uses peer loss to detect suspected noisy labeled data instances without unsupervised training. **AUM (Area Under the Margin) [22]** 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) [40]** uses the same noisy detection module as NDCC, but noise label correction is performed using k–nearest neighbors.

B. Complexity Analysis

1) $_{ns}$ Computation: The noisy threshold selection comprises three steps. N_{pre} denotes the number of the data instances in D_{pre}, and $N_{pre} < N$. K denotes the total number of classes. First, random re–sign the noisy label to random selected data instances (e.g., $|D_{pre}| \times \tau_p$) and output the loss value as shown in Algorithm 1 steps: 1–2, with the complexity of $\mathcal{O}(N_{pre}\tau_p)$. Next, the complexity for training pre–trian model with ResNet can be estimated as $\mathcal{O}(N_{pre}whC_hk^2dc)$, C_h is the number of channels, w and h are the width and height of the input data, k is the size of the filter, d is the spatial dimension of the filters, c denotes the number of filter, respectively (i.e., Algorithm 1 steps: 3). Final, the loss difference l_{diff} is calculated and $\tilde{D}_{ns}$ is computed, with the complexity of $\mathcal{O}N_{pre} + 1$. Thus, the complexity of $_{ns}$ computation is $\mathcal{O}(N_{pre}whC_hk^2dc)$.

2) Counterfactual Data Generation: In implementing NDCC, we use counterfactual data for label correction. Let N_{ns} denotes the input data instances for counterfactual label correction. For each data instances, the complexity for generate counterfactual data instance is $\mathcal{O}(wh)$ in each iteration, as shown in Algorithm 8 steps:5–6. Therefore, with maximum T_{cf} learning epochs for total number of N_{ns} data instances, the overall complexity for counterfactual data generation is $\mathcal{O}(N_{ns}whT_{cf})$.

3) NDCC: In implementing NDCC, we use counterfactual data for label correction. Let N_{ns} denotes the input data instances for counterfactual label correction. For each data instances, the complexity for generate counterfactual data instance is $\mathcal{O}(wh)$ in each iteration, as shown in Algorithm 8 steps:5–6. Therefore, with maximum T_{cf} learning epochs for total number of N_{ns} data instances, the overall complexity for counterfactual data generation is $\mathcal{O}(N_{ns}whT_{cf})$.

4) 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

TABLE II
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 III
WRONG DETECTION RATE X_{wt} (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	**0.13**	**0.07**	**0.02**	**0.01**	**0.13**	**0.09**	**0.05**	**0.03**	**0.18**	**0.10**	**0.09**	0.07	**0.20**	**0.14**	**0.09**	**0.03**
NDCC	0.17	0.15	0.07	0.04	0.16	0.14	0.13	0.09	0.26	0.17	0.14	**0.04**	0.27	0.21	0.17	0.09

TABLE IV
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 V
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**

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) **Wrong detection rate**: $X_{dw} = \frac{|X_n \cap (\tilde{D} - X_{\tilde{D}})|}{|X_n|}$ measures the ratio of misidentified clean data instances as noisy; (iii) **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-C, we only discuss X_{dt} and X_{dw}, since X_{dm} can be directly derived from X_{dt} by $X_{dm} = 1 - X_{dt}$.

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 following two scores: (i) **True counterfactual label correction rate** $\hat{X}_{cf_c} = \frac{|\hat{X}_r|}{|X_n|}$, and (ii) **False counterfactual label correction rate** $\hat{X}_{cf_w} = \frac{|\hat{X}_w|}{|X_n|}$.

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.

C. Experiments Results

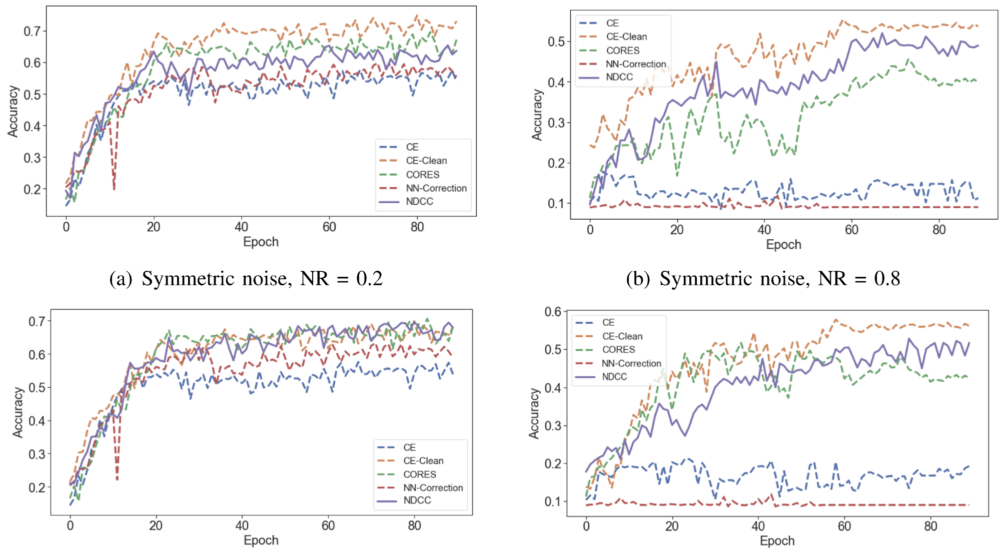
1) Noisy Label Detection: We begin by comparing NDCC and CORES. Table II and III show the true and wrong detection rates, X_{dt} and X_{dw}, for CORES and NDCC. A larger value of X_{dt} and smaller value of X_{dw} indicate better performance, as the goal is to detect as many true noisy labeled data instances as possible, while keeping the number of wrong detections low. Compared with CORES, NDCC's true detection rate X_{dt} increases almost three times more than X_{dw}, illustrating that automatically selecting the loss threshold is beneficial, as opposed to using a fixed threshold, as in [21].

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 IV 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 V shows the decreased noisy rate that different methods achieve. NDCC outperforms all baselines in all noisy environments across both datasets.

3) Overall Evaluation: Table VI 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. Figures 9(b) and

TABLE VI
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



(a) Symmetric noise, NR = 0.2
(b) Symmetric noise, NR = 0.8
(c) Asymmetric noise, NR = 0.2
(d) Asymmetric noise, NR = 0.8

Fig. 9. Test accuracy plots with increasing learning epochs in CIFAR–10 dataset with noise rate (NR) equals 0.2 (left column) and 0.8 (right column).

9(d) in particular, show that NSCF achieves close to the best performance, which would only be achievable if all training data instances were clean. In light noisy environments (i.e., $\tau \leq 0.4$), the performance of NDCC is close to AUM, the best performing baseline. Figures 9(a) and 9(c) show that both NDCC and CORES perform similarly to CE–Clean. The reason is that clean data instances comprise a large portion of training data instances under small noisy rate environment. However, compared with NN–Correction and CE, the accuracy of NDCC is higher, illustrating the effect of neither dealing with noisy labeled instances at all (i.e., CE) as well as using a naive label correction approach (i.e., NN–Correction). In summary, the experimental results confirm that NDCC can both effectively detect (and correct) noisy labeled data instances, and train robust classifiers even in the presence of sever label noise in the training set.

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 by pairing them with their most likely true labels. 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 (e.g., car financing) that are much "messier" than domains with clear ground truth (e.g., computer vision or natural language processing). We additionally wish to evaluate the scalability of our proposed approach using larger and more diverse datasets.

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] Z. Zhang, X. Zhou, X. Zhang, L. Wang, and P. Wang, "A model based on convolutional neural network for online transaction fraud detection," *Security and Communication Networks*, vol. 2018, 2018.

[5] 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.

[6] Y. Tang, F. Borisyuk, S. Malreddy, Y. Li, Y. Liu, and S. Kirshner, "Msuru: Large scale e-commerce image classification with weakly supervised search data," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2518–2526.

[7] L. Cai and Y. Zhu, "The challenges of data quality and data quality assessment in the big data era," *Data science journal*, vol. 14, 2015.

[8] 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.

[9] 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.

[10] K. Huang, H.-G. Stratigopoulos, and S. Mir, "Fault diagnosis of analog circuits based on machine learning," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 2010, pp. 1761–1766.

[11] R. Fu, Y. Huang, and P. V. Singh, "Crowds, lending, machine, and bias," *Information Systems Research*, vol. 32, no. 1, pp. 72–92, 2021.

[12] 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.

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

[14] 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.

[15] 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.

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] S. Verma, J. Dickerson, and K. Hines, "Counterfactual explanations for machine learning: A review," *arXiv preprint arXiv:2010.10596*, 2020.

[26] J. Hartford, G. Lewis, K. Leyton-Brown, and M. Taddy, "Deep iv: A flexible approach for counterfactual prediction," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1414–1423.

[27] 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.

[28] 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.

[29] J. Huang, L. Qu, R. Jia, and B. Zhao, "O2u-net: A simple noisy label detection approach for deep neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

[35] 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.

[36] R. K. Ando, T. Zhang, and P. Bartlett, "A framework for learning predictive structures from multiple tasks and unlabeled data." *Journal of Machine Learning Research*, vol. 6, no. 11, 2005.

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

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

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

[40] 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.

Charalampos Chelmis , Assistant Professor in Computer Science at the University at Albany, State University of New York, and Director of the Intelligent Big Data Analytics, Applications, and Systems (IDIAS) Lab, conducts research on data–intensive computing involving high–dimensional and/or interrelated data, and social good applications. He has served and is serving as Co–Chair, TPC member or reviewer in numerous international conferences and journals including TheWebConf, ASONAM, and ICWSM. He is currently Associate Editor of Social Network Analysis and Mining Journal (SNAM), and has served as Guest Editor for the Encyclopedia of Social Network Analysis and Mining. He earned his Ph.D. and M.Sc. degrees in Computer Science in 2013 and 2010, respectively from the University of Southern California, and B.S. in Computer Engineering and Informatics from the University of Patras, Greece in 2007.

Wenting Qi , received the B.S. degree in automation from the Beijing University of Technology, China in 2017, and earned M.S. degree in Electrical Engineering from the University of Southern California in 2019, Los Angeles, CA, USA. She is currently working towards the Ph.D. degree in Computer Science at the University at Albany, Albany, NY, USA. Her research interests include noisy detection, hierarchical classification, and explainable machine learning.

APPENDIX

We show that the optimal solutions for $\mathbf{W}$ and $\hat{\mathbf{x}}$ can be acquired in an iterative manner by alternating search, according to Equation 1 and the following lemma.

Lemma 1: The optimal solution found by minimizing $\mathbf{W}$ first and then minimizing $\hat{\mathbf{x}}_\mathbf{i}$ is the same as the optimal solution found by jointly minimizing $\mathbf{W}$, $\hat{\mathbf{x}}_\mathbf{i}$.

The parameters $\mathbf{W}$ of the learning model and the optimal counterfactual data instance $\hat{\mathbf{x}}_\mathbf{i}$ with respect to x_i are independent. The global optimum $\hat{\mathbf{x}}_\mathbf{i}$ is unknown and pre–existed, and we wish to use learning model with parameters $\mathbf{W}$ to find it. Therefore, the simultaneously obtained optimal solutions are $\mathbf{W}^*$ and $\hat{\mathbf{x}}_i^*$. Global minimum $g(\mathbf{W}^*, \hat{\mathbf{x}}_\mathbf{i}^*)$ satisfies the following equation:

$$g(\mathbf{W}^*, \hat{\mathbf{x}}_\mathbf{i}^*) \leq \min_{\hat{\mathbf{x}}_\mathbf{i}} \min_{\mathbf{W}} g(\mathbf{W}, \hat{\mathbf{x}}_\mathbf{i}). \tag{5}$$

Because for any $\hat{\mathbf{x}}_i$, $\min_{\hat{\mathbf{x}}_i} g(\mathbf{W},\, \hat{\mathbf{x}}_i) \leq g(\mathbf{W}^*,\, \hat{\mathbf{x}}_i)$,

$$\min_{\hat{\mathbf{x}}_\mathbf{i}} \min_{\mathbf{W}} g(\mathbf{W}, \hat{\mathbf{x}}_\mathbf{i}) \leq \min_{\hat{\mathbf{x}}_\mathbf{i}} g(\mathbf{W}^*, \hat{\mathbf{x}}_\mathbf{i}) \tag{6}$$

and

$$\min_{\hat{\mathbf{x}}_\mathbf{i}} g(\mathbf{W}^*, \hat{\mathbf{x}}_\mathbf{i}) \leq g(\mathbf{W}^*, \hat{\mathbf{x}}_\mathbf{i}^*). \tag{7}$$

Combining Equations (6) and (7), we get

$$\min_{\hat{\mathbf{x}}_\mathbf{i}} \min_{\mathbf{W}} g(\mathbf{W}, \hat{\mathbf{x}}_\mathbf{i}) \leq g(\mathbf{W}^*, \hat{\mathbf{x}}_\mathbf{i}^*). \tag{8}$$

Finally, comparing Equations (5) and (8), we get

$$g(\mathbf{W}^*, \hat{\mathbf{x}}_\mathbf{i}^*) = \min_{\hat{\mathbf{x}}_\mathbf{i}} \min_{\mathbf{W}} g(\mathbf{W}, \hat{\mathbf{x}}_\mathbf{i}) \tag{9}$$

Using Lemma 1, instead of showing $g(\mathbf{W}, \hat{\mathbf{x}}_\mathbf{i})$ is convex for $\mathbf{W}$ and $\hat{\mathbf{x}}_\mathbf{i}$ simultaneously, we can show that $g(\mathbf{W}, \hat{\mathbf{x}}_\mathbf{i})$ is convex with respect to $\mathbf{W}$ and $\hat{\mathbf{x}}_\mathbf{i}$ separately.