ABSTRACT  Obtaining data with correct labels is crucial to attain the state-of-the-art performance of Convolutional Neural Network (CNN) models. However, labeling datasets is significantly time-consuming and expensive process because it requires expert knowledge in a particular domain. Therefore, real-life datasets often exhibit incorrect labels due to the involvement of nonexperts in the data-labeling process. Consequently, there are many cases of incorrectly labeled data in the wild. Although the issue of poorly labeled datasets has been studied, the existing methods are complex and difficult to reproduce. Thus, in this study, we proposed a simpler algorithm called “Deep Clean Before Training Net” (DCBT-Net) that is based on cleaning wrongly labeled data points using the information from eigenvalues of the Laplacian matrix obtained from similarities between the data samples. The cleaned data were trained using deep CNN (DCNN) to attain the state-of-the-art results. This system achieved better performance than the existing approaches. In conducted experiments, the performance of the DCBT-Net was tested on three commercially available datasets, namely, Modified National Institute of Standards and Technology (MNIST) database of handwritten digits, Canadian Institute for Advanced Research (CIFAR) and WebVision1000 datasets. The proposed method achieved better results when assessed using several evaluation metrics compared with the existing state-of-the-art methods. Specifically, the DCBT-Net attained an average 15%, 20%, and 3% increase in accuracy score using MNIST database, CIFAR-10 dataset, and WebVision dataset, respectively. Also, the proposed approach demonstrated better results in specificity, sensitivity, positive predictive value, and negative predictive value evaluation metrics.

INDEX TERMS  Clustering, deep convolutional neural networks, eigenvalues and eigenvectors, image classification, noisy (corrupted) labels.
for labeling without human expert intervention are preferred. Crowdsourcing and online queries are clear examples of the cases, where data are labeled by nonexperts. However, due to insufficient knowledge in the field and subjective judgment of the nonexperts, datasets often have a large number of incorrect labels. Additionally, more serious problems such as data corruption may occur while the aforementioned methods are utilized for data labeling. Consequently, these issues negatively affect the performance of the DCNN model because it learns from incorrect or corrupted labels during the training phase and accepts them as correct making it incorrectly classify input data in real-life applications. This was also proven in [7], where the authors discovered that CNNs can retain very large datasets. Also, they noticed that in the existence of corrupt labels, a model failed to generalize unseen data with correct labels because it learned from incorrectly labeled data during the training stage.

Generally, corrupted labels in dataset might cause large problems in all types of supervised learning. Nevertheless, it is common to face poorly labeled datasets in the wild. As the goal of image classification is to classify images with high accuracy, we were required to find methods for addressing the problem of incorrect labels. The first and most simple method is to simply select correctly labeled images manually, although this process is exceedingly tedious. Moreover, when the dataset contains millions of samples, manual selection of correctly labeled data might become enormously time-consuming. Therefore, instead of choosing the samples with correct labels by hand, we utilize the power of a computing device and have it pick the data we expected. To achieve this, an engineer should construct a smart algorithm does not remove essential and important data samples. Thus, training machine learning (ML) algorithms with noisy data is a vigorous research field.

Training a model with corrupted labels has been broadly studied so far [8]–[10]. We can roughly divide existing techniques into two subgroups, namely, statistical and DL methods, which focus on training DNNs using poorly labeled data. They will be discussed in detail in the second section of this study; however, we concluded that majority of those techniques have complex explanations, therefore, they are quite difficult to replicate.

Although the existing techniques dealt with the data containing corrupted labels, there are some directions that can be enhanced. First, the majority of the methods depend heavily on very complex algorithms [11]. Second, due to the complexity of the methods, it was very difficult to reproduce and use these methods for other datasets. Third, most of them trained two DCNNs simultaneously, which are both time and computationally expensive [12]–[14]. To address these issues, we proposed a practical algorithm that utilized both unsupervised and supervised learning methods and attained the state-of-the-art results compared to the existing approaches. Generally, the proposed method dealt with the aforesaid aspects and contributed to enhancing the area in the following ways:

- Some of the existing methods require significantly complicated algorithms to clean the incorrectly labeled data [15]–[17] or train two simultaneous networks [12]–[14]. Consequently, these factors make the currently available models’ implementation complicated and expensive in terms of time and computation. In contrast, The DCBT-Net employs relatively simpler data cleaning algorithm to make the implementation easier and uses less computationally expensive network for more efficient training process. Also, due to the requirement of fewer complex steps, individuals with basic engineering skills can reproduce the DCBT-Net.
- The DCBT-Net deals with the various types of noise in the data, such as 20% and 50% incorrectly labeled training samples. The assessment of the proposed method using several ML evaluation metrics shows its superior performance over the currently available approaches. Specifically, it exhibits better performance on training DCNN using both 20% and 50% incorrectly labeled data compared with the existing methods.
- Although data labeling is a very tedious and expensive process, it may not always produce absolutely correct data due to subjective views and human errors. The proposed method alleviates the time-consuming and costly data labeling process since it can obtain competitive results with particular amount of incorrectly labeled data. Consequently, this makes the data collection and labeling processes quick and inexpensive. Specifically, the proposed method ensures state-of-the-art performance even with poorly labeled datasets.

This study is organized as follows. In Section II, we present an overview of the research works related to the issue of training ML and DCNN models with noisy labeled data. Section III provides information on the DCBT-Net. Section IV describes the conducted experiments and their results. Section V contains the comparison and discussion of the proposed model results with the existing models. Finally, Section VI summarizes this study with the conclusion and outlines potential future study directions.

II. RELATED WORK

Many studies have been conducted on the detrimental outcomes of training a model using data containing incorrect labels that produced various solutions for tackling the issue [18]. As stated above, existing techniques are classified into statistical and DL methods. Regarding the former, they mainly contributed to tackling corrupted labeled data problem theoretically [13]. For example, Natarajan et al. investigated the problem of binary classification using the existence of random noise and proposed a simple unbiased estimator with weighted surrogate loss [9]. Menon et al. used class probability estimation to study noisy labels and identified that balanced error can be optimized without the knowledge of corrupted labels with minimal range of classification risks [19]. Liu et al. also studied a classification problem with corrupted labels [20] and demonstrated that surrogate loss,
used with important weighing, can be successfully used for classification of the task with noisy labels on synthetic and real datasets. Bootkrajang et al. proposed new a regularization method [21] that deals with noise in high dimensions and demonstrated its usage in concrete applications.

The other subgroup of the methods that deal with corrupted labels comprise of solutions for DL models. Specifically, Reed et al. proposed an approach [22] that ignored the presence of incorrect labels learned from neural network parameters and the noise distribution at the same time. Mnih et al. proposed different loss functions [23] that tackled incorrectly labeled data issues and trained deep neural networks on complex datasets. Sukhbaatar et al. proposed a method [24] that matched the output of the model using a corrupted label distribution. A new crowd layer that enables training of end-to-end deep neural networks using corrupted labels was introduced by Rodrigues and Pereira [25]. Tanaka et al. presented a combined optimization frameworks [26] that can fix incorrect labels during training stages along with other model parameters. Veit et al. [27] showed the efficiency of training with the noisy data first and then fine-tuned it with clean data. The method resulted in impressive results for enormous datasets with approximately 10 million samples.

Apart from the above mentioned works, the most influential techniques for dealing with corrupted labels are the S-model [15], Bootstrap [16], F-correction [17], Decoupling [28], MentorNet [12], Co-teaching [13], and Group-teaching [14]. The earlier methods greatly contributed to the progress of training models using poorly labeled data. Co-teaching enhanced the Decoupling and MentorNet methods by addressing their shortcomings. Finally, inspired by Co-teaching, the authors of Group-teaching proposed a new method that is superior to the existing methods.

The authors of S-model proposed a technique that can be trained using noisy labeled data and showed that learning is possible without clean data. They also demonstrated that addition of SoftMax output layer allows to use the algorithm even with deep neural networks [15]. Bootstrap technique creators proposed an algorithm, which creates noise distribution matrix that maps predictions of the model to the targets. The loss computed from the mapping allows the model to explore the noisy data features [16]. Similarly, the F-correction method also depended on the noise transition matrix building. It was aimed at correcting loss by the noise transition matrix. In the first stage, a regular model was trained to build the noise transition matrix and another model was used to make predictions based on the earlier defined matrix [17].

The essence of the Decoupling technique is to let the classifier decide whether to update the model or not by handling each sample of the dataset serially. Also, the classifier performs huge number of updates at the beginning of the training and slowly decreases the updates at the end. To achieve this, the authors trained two deep models and updated them when their predictions differ [28]. However, the Decoupling technique could not deal with noisy labels in a detailed manner. Similarly, the MentorNet proposed simultaneous training of two networks, namely, an extra model - StudentNet and main model - MentorNet. The latter was updated based on feedback from the former and whenever this process occurred the authors corrected its parameter and used it along with StudentNet to reduce the objective function [12]. However, the method suffered from accumulated error resulting from the bias of sample-choice. The Co-teaching technique also trained two networks, but the novelty of the method was to filter various kinds of errors caused by noisy labels. By doing this, the authors enhanced the Decoupling and the MentorNet approaches, which slowly accumulated the error because the error from the first classifier was returned in each following mini-batch data [13]. The Co-teaching method showed its power in exceedingly corrupted data with the noise rate of 50%. However, two networks of the Co-teaching approach became consistent by being unable to select the correctly labeled samples for each other. This issue was addressed in the Group-teaching method, where the authors trained a group of convolutional neural networks and allowed them to teach each other by choosing possibly clean samples for each network in each mini-batch. Each network was trained using the backpropagation algorithm [29] and selected samples from other networks before updating [14]. Although, this method attained the state-of-the-art performance by obtaining superior results compared with the existing approaches, it lacked theoretical guidance on explaining why the method performed better.

III. THE PROPOSED METHODOLOGY

As stated in the first section of this study, we used a data cleaning algorithm followed by a DCNN model. Figure 1 presents the graphical illustration of the DCBT-Net. The proposed approach for dealing with noisy labels comprises of the following stages:

- In the first step, the dataset with incorrect labels is divided into several parts. The number of divided parts is determined by the number of classes in the dataset.
- After the dataset is divided into several parts, each undergoes data cleaning process, which separates the input into two clusters.
- After inspecting each of the clusters, the one with correct labels is selected for further training, while the second cluster samples are kept as test data.
- In cases for a network with correct labels for model training. After few pre-processing steps, the new dataset is trained using a DCNN model.
- The trained model is then utilized to predict samples that are unseen in the training stage.

A. DATA CLEANING

In the data cleaning process of the proposed method, we used an unsupervised learning algorithm based on graph Laplacian inspired by [30]. This process has three steps. First, considering a dataset $X$ with $m$ points: $X := \{x_1, x_2, \ldots, x_{m-1}, x_m\}$,
in the first step, we preprocessed the data points by representing the data in matrix form and formulating Laplacian matrix $L$. To obtain the Laplacian matrix adjacency matrix $A$ and degrees matrix $D$ must be formulated.

The adjacency matrix is a matrix containing values of edges (weights) based on the distance between the points of dataset $X$. It has zero value elements, except for the data-points $x_i$ and $x_j$, which are connected by an edge. Conversely, the degree matrix $D$ is a diagonal matrix containing the sum of the weights from a data point. After formulating the adjacency and degree matrices, the Laplacian matrix is computed using (1) as follows:

$$L = D - A$$  \hspace{1cm} (1)

Figure 2 shows adjacency, degree, and Laplacian matrices for the dataset $X$.

From Figure 2, the adjacency matrix exhibits edge values, such as $e_{12}$, $e_{1m}$, $e_{2m}$, where subindices represent the number of data points. Basically, if there is an edge between the points $x_i$ and $x_j$, then the edge value is $e_{ij}$, otherwise 0. Conversely, the degree matrix contains zero values, except for diagonal elements. Each diagonal element is computed by summing values of the corresponding edges. In fact, an $i^{th}$ diagonal element of the degree matrix is computed as shown in (2):

$$d_i = \sum_j e_{ij}$$ \hspace{1cm} (2)

Based on (1) and (2), we can formulate an equation for the Laplacian matrix as follows:

$$L_{ij} = \begin{cases} -e_{ij} & \text{when } e_{ij} \neq 0 \\ 0 & \text{when } e_{ij} = 0 \end{cases}$$ \hspace{1cm} (3)

In the second stage, we computed eigenvectors and eigenvalues of the Laplacian matrix obtained from the first step and represented each data sample in lower dimension using the eigenvectors. Table 1 shows the eigenvalues and eigenvectors of the Laplacian matrix.

| Eigenvalues | Eigenvectors |
|-------------|--------------|
| $\hat{\lambda}_1$ | $V_{11}$, $V_{21}$, ..., $V_{n-1}$, $V_n$ |
| $\hat{\lambda}_2$ | $V_{12}$, $V_{22}$, ..., $V_{n-2}$, $V_{n-1}$ |
| $\hat{\lambda}_m$ | $V_{1m}$, $V_{2m}$, ..., $V_{n-m}$, $V_{nm}$ |

After the eigenvalues of the Laplacian matrix were computed, we obtained the second eigenvalue with the corresponding eigenvector to perform bipartitioning of the selected values. In the last step, we conducted clustering on the 1-dimensional vector obtained from the second stage to assign the points to one of the two possible clusters. There are different options for setting the splitting threshold. In the DCBT-Net, we used threshold of 0, meaning that all negative components were grouped in the first cluster, and the remaining positive components were assigned to the second cluster.
In our study, we expect that the data points with correct and incorrect labels are separated into two different clusters, so we can select the cluster using the data points with correct labels. Algorithm 1 summarizes the data cleaning process.

### Algorithm 1 Data Cleaning Algorithm

1: **Input** dataset $X$ and labels $y$; 
2: for $i = 1, 2, \ldots$, length of $y$ do 
3: Split $X$ based on $y == i$; 
4: Obtain degree matrix $D$ based on $A$; 
5: Obtain Laplacian matrix $L$; 
6: Obtain the eigenvalues ($\lambda_1, \lambda_2, \ldots, \lambda_n$) and eigenvectors ($v_1, v_2, \ldots, v_n$) of $L$; 
7: Let $V \in \mathbb{R}^n$ be matrix containing the eigenvectors; 
8: for $j = 1, 2, \ldots$, length of $V$ do 
9: Cluster the data in $v_j$ into clusters $c_1$ and $c_2$; 
end 
10: Update $X[i]$ and $y[i]$ based on $c_1$; 
end 
11: **Output**: $X$ and $y$.

### B. DATA PREPARATION

After obtaining the dataset from the data cleaning process, it is prepared to be inputted into DCNN. First, the data points are standardized, which makes them follow a standard normal distribution. This can be attained by subtracting the mean of the data ($\mu$) from a data point and divide the result by the standard deviation of the data ($\sigma$) as shown in (4):

$$X = \frac{X - \mu_X}{\sigma_X} \quad (4)$$

Since some data points were lost in the data cleaning process, data augmentation was used to compensate the lost data by creating modified training examples. The data augmentation was performed based on the data points features; thus, data augmentation methods vary depending on the dataset.

### C. DATA LEARNING

After the data is “cleaned” and prepared for training, DCNN is formulated. Depending on the complexity of the dataset, DCNN may change, meaning that as the dataset is more complicated, more complex DCNN architecture is selected.

### IV. EXPERIMENTS AND RESULTS

In the theoretical calculations represented in the proposed methodology section, we observed prospective performance of the DCBT-Net. To confirm this, we conducted practical experiments on real-image datasets. In this section, we discuss the setup and results of the conducted experiments.

#### A. DATASETS

We tested the DCBT-Net on three publicly available datasets. The first was MNIST handwritten digits database [31], the second was CIFAR-10 dataset [32], and the third was WebVision1000 [33]. General information on the aforementioned datasets is provided in Table 2.

From Table 2, there are 60,000 and 50,000 images in the MNIST and CIFAR-10 datasets, respectively. Although the number of training images for training phase differs, the testing stage data point’s number is the same. Both datasets have 10,000 images for testing the model performance using unseen examples during the training phase. Regarding the WebVision1000 dataset, it contains more than 2.4 million noisy image from web, which are separated into 1,000 different classes as ImageNet database [6]. We use this dataset because it has large-scale images and the training data contains noisy labels. Since the WebVision1000 comprises of great number of training samples, we used a 100-class subset of the database due to limited computational resources. This subset contains approximately 240,000 different sizes of color images with noisy labels for training and 5,000 color images for testing phases, respectively.

Figure 3 shows the distribution of the number of training examples per class in the considered datasets.

Notably, Figure 3 shows that the distribution of the training examples of both datasets is balanced. The MNIST handwritten digit database has a different number of training samples ranging from 5,421 for class 5 to 6,742 for class 1, while the CIFAR-10 dataset has perfectly equal distribution of the training examples, meaning that each category contains 5,000 samples for training.

After analyzing the datasets, we applied noise to both datasets in the same way as in [13]. Because the considered datasets were correctly labeled, we modified the labels of the training sets using noise transition matrix $N$, each element of which corresponds to (5):

$$N_{ij} = P(\hat{y} = j | y = i) \quad (5)$$

Equation (5) shows that the noisy label $\hat{y}$ is transformed from correct label $y$. For the experiments, we selected two different versions of the same dataset with 20% and 50% noise rate, respectively. We did not increase the noise rate higher than 50% because this makes the learning impossible unless further assumptions are considered [13]. This was proven in [14], too. The maximum amount of noise in the data, basically, means that every second label has wrong labels in the aforementioned datasets.

| Dataset names   | Type of data | Image size | Images       |
|-----------------|--------------|------------|--------------|
| MNIST           | Handwritten digits | 28 x 28   | Training 60,000, Testing 10,000 |
| CIFAR-10        | Various objects | 32 x 32   | Training 50,000, Testing 10,000 |
| WebVision1000   | Noisy web data | Various   | Training 240,000, Testing 5,000 |

### TABLE 2. General description of the datasets used in the experiments.
FIGURE 3. The distribution of the training examples per class (a) MNIST handwritten digit database; (b) CIFAR-10 dataset.

To simulate the 20% and 50% incorrectly labeled dataset, we randomly selected $\varepsilon$ training images from each category, and randomly assigned them into $c-1$ class labels. The transformed matrix $T$ of the aforementioned process with noise rate of $\varepsilon$, and $c$ number of categories is provided in (6) [14].

$$T = \begin{bmatrix}
1 - \varepsilon & \varepsilon & \varepsilon & \varepsilon \\
\varepsilon & c - 1 & \varepsilon & \varepsilon \\
c - 1 & 1 - \varepsilon & \cdots & \cdots \\
\varepsilon & \cdots & \varepsilon & c - 1 \\
\cdots & \cdots & \cdots & \cdots \\
\varepsilon & \cdots & \varepsilon & c - 1 \\
c - 1 & \varepsilon & \cdots & 1 - \varepsilon \\
\varepsilon & \cdots & \cdots & \varepsilon \\
\cdots & \cdots & \cdots & \cdots \\
\varepsilon & \cdots & \cdots & \varepsilon \\
c - 1 & \varepsilon & \cdots & 1 - \varepsilon
\end{bmatrix}$$

(6)

Figure 4 shows the sample training images along with the corresponding class labels to ensure the training examples contain noise in labels.

Figure 4 shows randomly selected training examples from the datasets. Observably, the dataset was labeled incorrectly, since in (a) there are 13 images, which are 0, while the rest are totally different digits. The same tendency was observed in (b), which has only few images of number 7. Similarly, roughly half of the images in (c) and (d) differed from cars and dogs, respectively.

Because the considered datasets have ten categories, we first separate each of the datasets into ten parts as stated in Figure 1. After separating the training examples with noisy labels into ten different categories, we conducted the data cleaning process for each group of the separated dataset.

FIGURE 4. Training examples of the transformed data with wrong labels: (a) MNIST database training images labeled as 0; (b) MNIST database training images labeled as 7; (c) CIFAR-10 dataset training images labeled as car; (d) CIFAR-10 dataset training images labeled as dog.

The data cleaning Algorithm 1 of the DCBT-Net could distinguish correctly and incorrectly labeled training examples notably well. Figure 5 presents the results of this process.

As it can be observed from Figure 5, the data cleaning algorithm could not perfectly separate the correctly labeled examples from the incorrectly labeled examples due to imperfect nature of clustering algorithms. However, it is noted that the resulting datasets from the cleaning algorithm were considerably high in quality compared with the noisy labeled datasets. Regarding the images that cleaning algorithm failed to distinguish and kept as correctly labeled data points, they look quite similar to the ground truth labels. In these cases, the human eye might face difficulties in separating these training images. Additionally, it should be noted that the process of making two distinct clusters takes only several seconds, so for ten categories we spent a few minutes, which is several times faster than training DCNN model for the same purpose of data cleaning.

Graphical illustration of the proposed method for the MNIST handwritten digits database and CIFAR-10 dataset is presented in Figure 8 below.

B. DATA PREPARATION

After obtaining the datasets from the data cleaning process, we prepared them for training phase using DCNN. First, we standardized the datasets to follow a standard normal distribution. Second, since thousands of incorrectly...
labeled training images were lost in the data cleaning process, we decided to compensate these data points using data augmentation on the cleaned data [14]. Based on the characteristics of the training images, we augmented the training data through the following ways:

- changing the angle of images in the counterclockwise direction rotating by 50°;
- randomly rotating images by 20°;
- shifting the dimensions of the images by 10%;
- zooming images by 10%;

Additionally, we filled the pixels obtained from the data augmentation, which outside the boundary, with a constant value of 0. Figure 6 shows the results of the data augmentation applied on a sample image from the considered datasets.

C. BASELINE MODELS

We selected seven existing methods that attained state-of-the-art performance in training DCNNs using poorly labeled datasets, namely S-model, bootstrap, F-correction, decoupling, MentorNet, Co-teaching, and Group-teaching approaches. Because these models were discussed in detail in the related work section of this study, we will not specify them in this section.

D. DCBT-NET MODEL ARCHITECTURE

As stated in the proposed methodology section, architecture of the DCNN model depends on the complexity of a dataset. Although the existing methods used 9-layer CNN network for training models with the considered datasets [13], [14], we noticed that training such deep CNN for the considered datasets is both time-consuming and computationally expensive. Additionally, the nine-layer CNN resulted in poor performance in the conducted experiments. This is in line with the findings in [14], where the authors stated that simpler DCNN networks obtain better generalizability since more powerful ones are better at remembering too many noisy labels. Therefore, considering these facts, we formulated a comparatively simple DCNN model to test the DCBT-Net, which obtained better results in the conducted experiments. Figure 7 presents the architecture of the DCNN model.

Figure 7 shows the graphical representation of the DCNN model used in the conducted experiments. The conv blocks are sequences of convolutional layers, followed by batch normalization layer and activation function.
In the experiments, we utilized parametric rectified linear unit (PReLU) [34] activation function, since it resulted in the best performance compared to the other activation functions, such as rectified linear unit (ReLU) [1], exponential linear unit (ELU) [35], and leaky ReLU [36]. The first conv block comprises a convolution layer with 16 kernels of size $5 \times 5$, stride of 1, which is responsible for the step of kernels across the image and “same” padding that keeps the image size intact. The output of the convolution layer then passes through a batch normalization layer followed by PReLU activation function. The second conv block contains 32 kernels with $3 \times 3$ size that perform the convolution operation with stride of 1 and “same” padding. The second conv block is followed by maxpooling layer that reduces the size of the image, and consequently reduces computational time and cost. Then the dropout method [37] with rate of 0.2 was applied to reduce overfitting. This helps to randomly eliminate 20% of the neurons in the layer. The third conv block comprises 64 filters of size $3 \times 3$, followed by the maxpooling operation and the dropout layer. The output of the dropout layer then passes through the fourth conv block with 128 filters, followed by maxpooling operation and dropout layer with a rate of 0.3. Next, the output image is flattened into a vector before being inputted into fully connected layer with 256 units, which uses ReLU activation function. Finally, the last fully connected layer with SoftMax activation function outputs ten values with probabilities for each category of the datasets.

**E. TRAINING SETUP**

We implemented the baseline models along with the proposed scheme DCNN model using the 3.6.9 version of Python and the 2.4.0 version of the Keras framework. We conducted experiments using NVIDIA Tesla V100-SXM2 GPU. For all experiments, we initialized the weight parameters using Kaiming weight initialization strategy [34] and Adam optimizer [38] with a momentum of 0.9 and learning rate of $10^{-3}$. Additionally, we utilized the sparse categorical cross-entropy as the function to minimize. We trained the model for two hundred epochs with a batch size of 512 for the MNIST handwritten digits database and CIFAR-10 dataset.

Since large scale WebVision1000-100 database contained images with various sizes, we cropped them into $224 \times 224$ patches to obtain the same size input images. Moreover, for fair comparison, we formulated ResNet-18 architecture [3] as in [14] and trained the model for two hundred epochs with batch size of 32.

**F. EVALUATION METRICS**

As stated in the proposed methodology section, the distributions of both datasets are not skewed, meaning that the training examples for each class are normally balanced. Considering this fact, accuracy metric, which is the ratio of the correctly predicted samples to the total number of data (see (7)), might be enough to evaluate the model performance. However, we assessed the model using other metrics to show its decent performance.

$$AS = \frac{\sum_{n} \hat{y}_n \cdot y_n}{\sum_{n} y_n}$$

Equation (7) provides the formula for computing accuracy score ($AS$) of a model, where $\hat{y}$ is the predicted value and $y$ is the ground truth label.

The other metrics we utilized to evaluate the model performance are sensitivity or true positive rate (TPR), specificity or true negative rate (TNR), positive predictive value (PPV), and negative predictive value (NPV). However, before formulating the aforementioned metrics, we defined *true positives*...
FIGURE 9. Comparison of the existing models and the DCBT-Net model performance using test accuracy scores on MNIST handwritten digits database 20% (top left), 50% (top right) noise rates and CIFAR-10 dataset with 20% (bottom left) and 50% (bottom right) incorrect labels.

TABLE 3. Confusion matrix.

|                | Positive ground truth label | Negative ground truth label |
|----------------|-----------------------------|-----------------------------|
| Predicted positive label | TP                          | FP                          |
| Predicted negative label  | FN                          | TN                          |

(TPs), false positives (FPs), true negatives (TNs), and false negatives (FNs). These terms are presented in a table called the confusion matrix.

TP is defined as the correct positive prediction (the same as the ground truth). FP is the positive prediction that is false (the ground truth is negative). FN corresponds to the negative prediction that is false (the ground truth is positive). Finally, TN is the negative prediction that agrees with the ground truth label (ground truth is positive).

Using the measures provided in Table 3, we define TPR, TNR, PPV, and NPV as follows:

$$TPR = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP}$$

$$PPV = \frac{TP}{TP + FP}$$

$$NPV = \frac{TN}{TN + FN}$$

G. EXPERIMENTAL RESULTS ON MNIST HANDWRITTEN DIGITS DATABASE AND CIFAR-10 DATASET

We conducted experiments on the datasets after applying two different noise rates, namely, 20% and 50% on the small-scale datasets, such as MNIST and CIFAR-10. The results obtained from the experiments are represented in Table 4.

Table 4 shows the information on the performance of the baseline models with the proposed method in terms of accuracy on the test sets of the MNIST handwritten digits database and CIFAR-10 dataset. The results show that among the existing methods MentorNet, Co-teaching, and Group-teaching significantly outperformed their peers and obtained similar accuracy in both datasets with different noise rates. Among them, Group-teaching attained slightly better performance in comparison with MentorNet and Co-teaching methods. However, the DCBT-Net obtained considerably better performance in MNIST handwritten digits database by attaining nearly 4% and 3% increase in accuracy on the data trained with 20% and 50% incorrect labels. Similarly, the proposed method outperformed the existing methods and obtained 77.70% and 73.39% accuracy on 20% and 50% noisy labeled CIFAR-10 datasets, respectively.

TABLE 4. Performance of the existing methods and DCBT-Net on test sets of the considered datasets with 20% and 50% noise rate.

| Model names | MNIST | CIFAR-10 |
|-------------|-------|----------|
|              | Accuracy (%) |         |
|              | 20% noise | 50% noise | 20% noise | 50% noise |
| Standard     | 79.24±0.08 | 52.42±0.12 | 76.58±0.23 | 47.43±0.19 |
| Bootstrap    | 85.04±0.29 | 67.72±0.35 | 60.89±0.40 | 51.97±0.21 |
| Decoupling   | 90.46±0.32 | 81.77±0.43 | 57.77±0.19 | 50.42±0.10 |
| F-correction | 88.00±0.16 | 80.24±0.52 | 67.92±0.04 | 62.18±0.13 |
| MentorNet    | 93.24±0.28 | 90.49±0.10 | 74.02±0.16 | 69.62±0.11 |
| Co-teaching  | 93.38±0.32 | 91.12±0.14 | 75.37±0.10 | 72.28±0.22 |
| Group-teaching | 93.89±0.13 | 91.87±0.21 | 76.09±0.14 | 72.3±0.07 |
| DCBT-Net     | 97.73±0.28 | 94.13±0.17 | 77.70±0.09 | 73.39±0.12 |
Also, we plotted a line graph of the test set ASs on every epoch in Figure 9 above. This figure shows test AS curves of the standard model (trained using wrong labels), the DCBT-Net model, and existing state-of-the-art models on the considered datasets with 20% and 50% corrupted labels. Observably, all models’ AS curves had similar tendency for 20% and 50% noisy labeled datasets. Logically, the models trained on 20% incorrectly labeled datasets obtained considerably higher ASs than those of the models trained on 50% noisy data. Evidently, the standard model was slightly inferior in contrast to the other models since it utilized poorly labeled data for training the model. Concerning the other models, bootstrap, F-correction, and decoupling models obtained high test accuracy scores at the beginning of testing phase but their performance dramatically decreased at the middle and final parts of the testing phase, resulting in poor overall performance. On the other hand, Mentor-Net, Co-teaching, and Group-teaching methods showed stable performance with a slight decrease in the test accuracy scores as the evaluation progressed, where Group-teaching outperformed MentorNet and Co-teaching in both datasets with 20% and 50% wrong labels. Regarding the DCBT-Net, although, it could not generalize well at the beginning epochs of the test phase, the proposed method slowly enhanced its accuracy and achieved the highest score at the end of the evaluation process using both datasets. Notably, the proposed method obtained the best performance in terms of accuracy score in all variations of the considered noisy datasets. Considering that the DCBT-Net model learned from cleaned data, the gradual increase in its performance is reliable and, unlike the standard model, it is likely to succeed in real-life applications.

To ensure that the proposed model’s robustness, we also assessed its performance on 50% noisy datasets using various evaluation metrics, namely, AS, TPR, PPV, and NPV. The results of the assessment are presented in Table 5.

From Table 5, the model trained using the DCBT-Net model performed well in classifying the handwritten digits of MNIST database and objects in the CIFAR-10 datasets using poorly labeled data. Regarding MNIST database of handwritten digits, the proposed method obtained accuracy of 96% and above in seven out of ten classes and a slightly lower result of 92.8% for class 9. The other classes, namely, class 1 and 8 had 80.4% and 83.5% classification accuracy rates, respectively. Conversely, the DCBT-Net model attained relatively moderate results on the CIFAR-10 dataset, due to the complexity of the dataset. Seven out of ten categories had accuracy scores of more than 80%; however, the accuracy rates for bird and dog classes were similar at about 65%. The classifier faced challenges in recognizing the cat images; consequently, being unable to classify almost every second cat image, it obtained an accuracy of 54.9%.

Similar results were attained in evaluating the model using the other metrics, such as TPR, TNR, PPV, and NPV. Specifically, almost perfect scores were obtained in evaluating the model using TRN and NPV in both datasets. This fact demonstrates that the model is very good at identifying true negatives than true positives.

After successfully training the model and obtaining satisfactory results, we decided to plot confusion matrices to investigate the performance of network in the classification of each category.

Figure 10 provides information on the number of correctly predicted test images and misclassified test samples. Most incorrect predictions of the DCBT-Net model in MNIST handwritten digit database were in categories 1 and 8. The model misclassified these digits and mainly predicted 3 and 5, respectively. For CIFAR-10 dataset, the proposed model incorrectly predicted images of plane, bird, and cat. In plane images, incorrect predictions of the model were roughly equally classified into the other categories. However, testing images of horse and deer were incorrectly classified as a bird. Similarly, cat images were mainly confused with dog and frog samples.

| Dataset names | Class names | AS  | TPR  | TNR  | PPV  | NPV  |
|---------------|-------------|-----|------|------|------|------|
| MNIST         | 0            | 0.988 | 0.988 | 0.996 | 0.964 | 0.999 |
|               | 1            | 0.804 | 0.804 | 0.999 | 0.993 | 0.976 |
|               | 2            | 0.975 | 0.975 | 0.998 | 0.979 | 0.997 |
|               | 3            | 0.986 | 0.986 | 0.971 | 0.795 | 0.998 |
|               | 4            | 0.968 | 0.968 | 0.999 | 0.987 | 0.997 |
|               | 5            | 0.984 | 0.984 | 0.986 | 0.875 | 0.998 |
|               | 6            | 0.976 | 0.976 | 0.998 | 0.981 | 0.997 |
|               | 7            | 0.986 | 0.986 | 0.994 | 0.947 | 0.998 |
|               | 8            | 0.835 | 0.835 | 0.997 | 0.971 | 0.982 |
|               | 9            | 0.928 | 0.928 | 0.996 | 0.967 | 0.992 |
| CIFAR-10      | Plane        | 0.803 | 0.627 | 0.939 | 0.604 | 0.944 |
|               | Car          | 0.846 | 0.881 | 0.988 | 0.887 | 0.987 |
|               | Bird         | 0.647 | 0.674 | 0.980 | 0.783 | 0.966 |
|               | Cat          | 0.549 | 0.572 | 0.974 | 0.699 | 0.955 |
|               | Deer         | 0.816 | 0.850 | 0.964 | 0.717 | 0.984 |
|               | Dog          | 0.650 | 0.677 | 0.978 | 0.770 | 0.966 |
|               | Frog         | 0.861 | 0.897 | 0.967 | 0.747 | 0.989 |
|               | Horse        | 0.803 | 0.836 | 0.983 | 0.842 | 0.982 |
|               | Ship         | 0.848 | 0.883 | 0.984 | 0.855 | 0.987 |
|               | Truck        | 0.816 | 0.850 | 0.986 | 0.864 | 0.984 |
H. EXPERIMENTAL RESULTS ON WEBVISION1000-100 DATASET

After completing tests on the small-scale datasets, we conducted experiments on WebVision1000-100 large-scale dataset, containing real-life noisy images from the web. Since this dataset comprises of significantly large number of images for training, the experiments on the dataset were time-consuming and computationally expensive. Therefore, instead of experimenting with all existing methods, we compared the proposed method only with those models that showed competitive performance on the small-scale datasets, namely, MentorNet, Co-teaching, and Group-teaching. The results of the experiments on the dataset different noise rates are presented in Table 6.

| Model names         | Accuracy (%) |
|---------------------|--------------|
|                     | 20% noise    | 50% noise    |
| Standard            | 78.93±0.23   | 49.72±0.21   |
| MentorNet           | 79.13±0.12   | 73.32±0.14   |
| Co-teaching         | 79.82±0.28   | 75.93±0.26   |
| Group-teaching      | 80.61±0.22   | 77.03±0.19   |
| DCBT-Net            | 81.12±0.16   | 77.71±0.14   |

From Table 6, there was only slight difference in the accuracy of the considered methods when they were trained on 20% incorrectly labeled data. Due to the complexity of the WebVision1000-100 dataset, the existing methods could not obtain high accuracy and converged at approximately 80% accuracy. Surprisingly, standard model’s performance was very similar to those with the state-of-the-art methods. Concerning the proposed model, it exhibited the best performance by achieving 81.12% accuracy and outperforming the existing methods by more than 1%. Regarding the training data with 50% incorrect labels, standard model failed to deal with extreme amount of noise in labels and attained only 49.72% accuracy. The DCBT-Net showed the best performance in the 50% wrongly labeled training data, too. Specifically, it obtained 75.71% accuracy that is 0.68%, 1.08%, and 3.39% better than Group-teaching, Co-teaching, and MentorNet, respectively.

V. DISCUSSION

In this section, we discuss the assessment of the considered models’ performance using various evaluation metrics on noisy versions of the MNIST handwritten digits database, CIFAR-10 dataset, and WebVision1000-100 dataset. Moreover, we measure the generalizability of the proposed method to unseen images. Additionally, we compare the amount of required time for training the existing methods and the proposed scheme. Finally, we discuss the advantages of the proposed method over the existing models.

A. DISCUSSION OF THE EXPERIMENTAL RESULTS

After obtaining the trained model from the experiments, we evaluated its performance on unseen data in the training phase that was eliminated from the data cleaning stage and kept as test data. To compare the results of the DCBT-Net with the state-of-the-art models, we collected the test set results, presented in Table 8.

| Dataset names     | Model names | AS    | TPR   | TNR   | PPV   | NPV   |
|-------------------|-------------|-------|-------|-------|-------|-------|
| MNIST             | Standard    | 0.624 | 0.633 | 0.958 | 0.845 | 0.961 |
|                   | Bootstrap   | 0.672 | 0.714 | 0.961 | 0.897 | 0.924 |
|                   | Decoupling  | 0.817 | 0.861 | 0.973 | 0.904 | 0.969 |
|                   | F-correction| 0.802 | 0.859 | 0.968 | 0.891 | 0.957 |
|                   | MentorNet   | 0.905 | 0.929 | 0.983 | 0.944 | 0.987 |
|                   | Co-teaching | 0.911 | 0.937 | 0.991 | 0.939 | 0.995 |
|                   | Group-teaching | 0.919 | 0.940 | 0.990 | 0.954 | 0.991 |
| DCBT-Net          | 0.934       | 0.943 | 0.993 | 0.946 | 0.993 |
| CIFAR-10          | Standard    | 0.351 | 0.389 | 0.824 | 0.487 | 0.738 |
|                   | Bootstrap   | 0.516 | 0.534 | 0.867 | 0.571 | 0.854 |
|                   | Decoupling  | 0.515 | 0.571 | 0.878 | 0.565 | 0.890 |
|                   | F-correction| 0.602 | 0.679 | 0.914 | 0.603 | 0.871 |
|                   | MentorNet   | 0.697 | 0.724 | 0.967 | 0.771 | 0.869 |
|                   | Co-teaching | 0.734 | 0.751 | 0.978 | 0.779 | 0.933 |
|                   | Group-teaching | 0.739 | 0.757 | 0.972 | 0.781 | 0.941 |
| DCBT-Net          | 0.764       | 0.775 | 0.974 | 0.777 | 0.974 |

From Table 7, the proposed method attained the highest accuracy when trained with 20% incorrect labels. Moreover, the increase of the noise rate negatively impacted on the model’s performance. Specifically, it achieved 92.67% and 91.17% top5 accuracy on the 40% and 50% wrongly labeled data, respectively.
proposed method had more FPs and FNs with respect to TPs and TNs in the MNIST database than the best-performing methods, such as Group-teaching and Co-teaching. Regarding the CIFAR-10 dataset, the DCBT-Net obtained higher number of FPs and FPs with respect to TNs and TPs than Co-teaching and Group-teaching methods, which obtained the best performance in these evaluation metrics.

Still, in the aforementioned cases, the proposed method always obtained the second-best result and the difference between the proposed method and the best-performing model’s result was insignificant. Therefore, in general, the proposed method’s overall performance in five evaluation metrics was considerably better than those of the existing methods. This ensured that the proposed approach achieved the state-of-the-art performance in training DCNN using extremely noisy data.

B. DISCUSSION OF THE DCBT-NET GENERALIZABILITY

To measure generalization ability of the DCBT-Net, we tested the performance of the proposed method on the unseen data. For this purpose, we utilized the training images that were excluded from the training stage in the data cleaning process because the algorithm decided that these training images have incorrect labels. Since these images did not partake either in training or test phase of the proposed method, they can serve as an appropriate tool to measure the generalizability of the model. To conduct the generalizability test, we standardized the testing images, which allowed them to follow the same distribution. The results of the testing are provided in Figure 11.

Figure 11 shows that the DCBT-Net exhibits decent generalization ability on randomly selected images that were not used during training and testing phases. The proposed method predicted correct categories for the vast majority of these images. However, the performance of the DCBT-Net on unseen data was inconsiderable; the model misclassified some images like 1, 4 digits and cat, dog images in MNIST handwritten digits database and CIFAR-10 dataset, respectively. Moreover, the generalizability of the proposed model on large-scale WebVision1000-100 dataset was also remarkable. Despite containing large number of categories, the DCBT-Net could correctly classify the images. Specifically, the model performed well in distinguishing lemons with oranges and hens with cocks. However, it failed to correctly predict the labels for digital clock, acoustic guitar, monitor, African chameleon, and assault gun. Since the input images and predicted labels were completely different, we believe that the misclassification was due to incorrect labels in the dataset. In general, considering the model used 50% of incorrectly labeled data for training, its generalizability is noteworthy.

C. DISCUSSION OF COMPUTATIONAL TIME REQUIRED TO TRAIN THE MODELS

Training time of a DCNN model is one of the important aspects in assessing network’s usability. As stated in the introduction section, the DCBT-Net not only obtains better accuracy when trained using noisy versions of the considered datasets, but it also efficient in terms of computational time. We provide comparison of the existing and proposed methods’ average training time per epoch in Figure 12.

Figure 12 represents average training time (in seconds) required to train a single epoch using different models on noisy labeled MNIST handwritten digits database and CIFAR-10 dataset. We selected MentorNet, Co-teaching, Group-teaching methods since they obtained the highest accuracy scores in the conducted experiments. Also, we provided training time only for these datasets because for training WebVision1000-100 dataset ResNet-18 model was used; therefore, the training time for all models were approximately the same. Regarding the training time for MNIST handwritten digits database, the proposed method significantly outperformed its counterparts by requiring nearly 2.5 and 4.4 times less training time than Co-teaching/Group-teaching and MentorNet methods, respectively. Similar tendency was observed when the models were trained on the noisy version of the CIFAR-10 dataset. The MentorNet was the most time-consuming with 20.90 seconds training time per epoch, while the Co-teaching and Group-teaching required almost 14 seconds for single epoch training. However, the DCBT-Net completed training an epoch in approximately 5 seconds obtaining the best performance in terms of efficiency.
obtained experimental results, the DCBT-Net model has several advantages. First, the proposed scheme uses relatively simple algorithms to separate correctly labeled samples from the noisy data. Second, the DCBT-Net is significantly more efficient in terms of time (see FIGURE) and computation due to its ability to achieve competitive performance by using less computationally intensive 9-layer CNN model in comparison with the existing models [13], [14]. Third, despite requiring fewer trainable parameters and less training time, the DCBT-Net obtains better performance in comparison with the existing models as was shown in the conducted experiments using three real-life datasets.

VI. CONCLUSION AND FUTURE WORK

Conclusively, we conducted research for training the DCNN model using a dataset with extremely noisy labels. We performed extensive literature review on various statistical and DL methods to alleviate the consequences of the issue. Based on knowledge from the related studies, we realized that the existing approaches for training with corrupted labels are either complex or difficult to implement. Therefore, we proposed a method that is relatively simple and effective. The DCBT-Net is based on cleaning the dataset excluding data samples that are incorrectly labeled using the eigenvectors of the similarity Laplacian matrix. Due to the simplicity of the implementation, the proposed method required few engineering skills only to be applied. In the conducted experiments using extremely noisy labeled MNIST handwritten digits database, CIFAR-10 dataset, and WebVision1000-100 datasets, the DCBT-Net showed a notable performance by attaining the state-of-the-art results and outperforming the existing methods. Also, the model obtained high scores in a number of evaluation metrics. Therefore, we concluded that the model addressed the training of DCNN models using poorly labeled data, thereby making the data collection process to be inexpensive and timely.

Although, the DCBT-Net obtained the state-of-the-art performance results using corrupted labels, it can be further enhanced in the future. One way of enhancing is to enhance the data cleaning algorithm that can perfectly separate the correctly labeled samples from the poorly labeled ones. The other method for improving the current scheme is to discover an approach for keeping the maximum number of data points in the training set. In the proposed scheme, the data cleaning process eliminated the incorrectly labeled data points; consequently, it decreased the number of samples for training. We will conduct further studies on the DCBT-Net by creating better algorithms that can further enhance the performance of this model.

ACKNOWLEDGMENT

This study was supported by the BK21 Four project (AI driven Convergence Software Education Research Program) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (4199990214394). This work was also supported by National Research Foundation of Korea. Grant Number: 2020R1A2C1012196. This research was conducted using the assistance of the “HPC Support” Project, supported by the ‘Ministry of Science and ICT’ and NIPA.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1106–1114, doi: 10.1109/CVPR.2014.09.007.
[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. 3rd Int. Conf. Learn. Represent. (ICLR), 2015, pp. 1–14.
[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Comput. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, E. Rhar, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proc. IEEE Comput. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.
[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR), 2017, pp. 1–14.
[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in Proc. IEEE Comput. Conf. Comput. Vis. Pattern Recognit., Jun. 2010, pp. 4700–4708, doi: 10.1109/CVPR.2009.5206848.
[7] C. Zhang, B. Recht, S. Bengio, M. Hardt, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR), 2019, pp. 1–15.
[8] B. Frenay and M. Verleysen, “Classification in the presence of label noise: A survey,” IEEE Trans. Neural Netw. Learning Syst., vol. 25, no. 5, pp. 845–869, May 2014, doi: 10.1109/TNNLS.2013.2292894.

[9] N. Nitajaran, I. S. Dhillon, P. Ravikumar, and A. Tewari, “Learning with noisy labels,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 1196–1204.

[10] N. Nigam, T. Dutta, and H. P. Gupta, “Impact of noisy labels in learning techniques: A survey,” in Advances in Data and Information Sciences (Lecture Notes in Networks and Systems). Singapore: Springer, 2020.

[11] J. Zhang, X. Wu, and V. S. Sheng, “Learning from crowdsourced labeled data: A survey,” Aritif. Intell. Rev., vol. 46, no. 4, pp. 543–576, Dec. 2016, doi: 10.1007/s10462-016-9491-9.

[12] L. Jiang, X. Zhou, T. Leung, J. Li, and L. Fei-Fei, “Montenet: Learning data-driven curriculum for very deep neural networks on corrupted labels,” in Proc. 35th Int. Conf. Mach. Learn. (ICML), 2018, pp. 2304–2313.

[13] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, “Co-teaching: Robust training of deep neural networks with extremely noisy labels,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), Dec. 2018, pp. 8527–8537.

[14] Y. Zheng, Y. Chen, and M. Sarem, “Group-teaching: Learning robust CNNs from extremely noisy labels,” IEEE Access, vol. 8, pp. 34657–34697, 2020, doi: 10.1109/ACCESS.2020.2974779.

[15] J. Goldberger, E. Ben-Reuven, and E. Faculty, “Deep neural—Networks using noise adaptation layer,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR) Conf. Track, no. 2014, 2017, pp. 1–7.

[16] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, “Training deep neural networks on noisy labels with bootstrapping,” in Proc. 3rd Int. Conf. Learn. Represent. (ICLR), 2015, pp. 1–11.

[17] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, “Making deep activations in convolutional network,” 2015, pp. 125–134.

[18] T. Liu and D. Tao, “Classification with noisy labels by importance reweighting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 3, pp. 447–461, Mar. 2016, doi: 10.1109/TPAMI.2015.2456899.

[19] J. Boottiraj and A. Kabán, “Label-noise robust logistic regression and its applications,” in Machine Learning and Knowledge Discovery in Databases (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2012, pp. 134–158, doi: 10.1007/978-3-642-33460-3_15.

[20] A. J. Bekker and J. Goldberger, “Training deep neural-networks based on unreliable labels,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Mar. 2016, pp. 1–10, doi: 10.1109/ICASSP.2016.7472164.

[21] V. Mnih and G. Hinton, “Learning to label aerial images from noisy data,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), Jul. 2017, pp. 1944–1952, doi: 10.1109/CVPR.2017.240.

[22] H. Song, M. Kim, D. Park, and J.-G. Lee, “Learning from noisy labels with deep neural networks: A survey,” 2020, arXiv:2007.08199 [Online]. Available: http://arxiv.org/abs/2007.08199.

[23] A. K. Menon, B. Van Rooyen, C. S. Ong, and R. C. Williamson, “Learning from corrupted binary labels via class-probability estimation,” in Proc. 32nd Int. Conf. Mach. Learn. (ICML), 2015, pp. 125–134.

[24] T. Liu and D. Tao, “Classification with noisy labels by importance reweighting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 3, pp. 447–461, Mar. 2016, doi: 10.1109/TPAMI.2015.2456899.

[25] J. Boottiraj and A. Kabán, “Label-noise robust logistic regression and its applications,” in Machine Learning and Knowledge Discovery in Databases (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2012, pp. 134–158, doi: 10.1007/978-3-642-33460-3_15.

[26] A. J. Bekker and J. Goldberger, “Training deep neural-networks based on unreliable labels,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Mar. 2016, pp. 1–10, doi: 10.1109/ICASSP.2016.7472164.

[27] V. Mnih and G. Hinton, “Learning to label aerial images from noisy data,” in Proc. 29th Int. Conf. Mach. Learn. (ICML), 2012, pp. 567–574.

[28] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, “Training convolutional networks with noisy labels,” in Proc. 3rd Int. Conf. Learn. Represent. (ICLR), 2015, pp. 1–11.

[29] F. Pereira and C. F. Pereira, “Deep learning from crowds,” in Proc. 32nd AAAI Conf. Artif. Intell. (AAAI), 2018, pp. 1–9.

[30] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aihara, “Joint optimization framework for learning with noisy labels,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 5552–5560, doi: 10.1109/CVPR.2018.00582.

[31] L. Yann, C. Corinna, and B. Christopher, “THE MNIST DATABASE of handwritten digits,” Courant Inst. Math. Sci., New York, NY, USA, Tech. Rep., 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[32] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Dept. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[33] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, “WebVision database: Visual learning and understanding from Web data,” 2017, arXiv:1708.02862 [Online]. Available: http://arxiv.org/abs/1708.02862.

[34] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1026–1034, doi: 10.1109/ICCV.2015.123.

[35] D. A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (ELUs),” in Proc. 4th Int. Conf. Learn. Represent. (ICLR), 2016, pp. 1026–1034.

[36] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” 2015, arXiv:1505.00853 [Online]. Available: http://arxiv.org/abs/1505.00853.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–2958, 2014.

[38] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in Proc. 3rd Int. Conf. Learn. Represent. (ICLR), 2015, pp. 1–15.