IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 6, 2020, accepted August 5, 2020, date of publication August 26, 2020, date of current version September 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3019465

Error-Based Noise Filtering During Neural Network Training

FAHAD ALHARBI, KHALIL EL HINDI, AND SAAD AL-AHMADI
Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
Corresponding author: Fahad Alharbi (fahd888@hotmail.com)

This work was funded and supported by the Deanship of Scientific Research at King Saud University through the initiative of DSR Graduate Students Research Support (GSR).

ABSTRACT The problem of dealing with noisy data in neural network-based models has been receiving more attention by researchers with the aim of mitigating possible consequences on learning. Several methods have been applied by some researchers to enhance data as a pre-process of training while other researchers have attempted to make models of learning aware of noise and thus able to deal with noisy instances. We propose a simple and efficient method that we call Error-Based Filtering (EBF) that is used during training as a filtration technique for supervised learning in neural network-based models. EBF is independent of the model architecture and can therefore be involved in any neural network-based model. Our approach is based on monitoring and analyzing the distribution of values of the loss (error) function for each instance during training. In addition, EBF can be integrated with semi-supervised learning to take advantage of the identified noisy instances and improve classification. An advantage of EBF is to achieve competitive performance compared with other state-of-the-art methods with many fewer additional tasks in a procedure of training. Our evaluation of the efficacy of our method on three well-known benchmark datasets demonstrates an improvement on classification accuracy in the presence of noise.

INDEX TERMS Neural networks, convolutional neural networks, noisy data, semi-supervised learning.

I. INTRODUCTION

During the past few years, deep learning has received wide-ranging attention and development at all levels. This attention has been increasing as a consequence of the ability they offer in making precise decisions and in demonstrating some human-like intelligence in various domains.

In general, models of deep learning depend on the availability of sufficient data. The data are expected to be clean and correct in order to achieve good performance. However, obtaining a clean dataset in the real world is often difficult, expensive, and time-consuming [1], [2], especially in domains with multiple classes. There is a positive correlation between the probability of noise and the quantity of data [3]. In other words, the larger the dataset is, the higher the chance is that it contains noise. Consequently, noisy data are a drawback for many domains. Real-world datasets are susceptible to having noisy instances, leading to an adverse impact on learning algorithms and decreasing the reliability of classifiers.

Noise can appear in a dataset in various forms. The most common forms of noise are feature noise and label noise. Potential consequences of noise affect various aspects of learning, such as the amount of data and time required for training a model in order to achieve an acceptable level of performance; in addition, label noise represents one of the most common causes of overfitting in machine learning and in deep learning methods in particular [4].

The presence of noisy instances in any dataset raises the demand for techniques to address potential negative consequences. For instance, various studies [5]–[11] have been conducted to develop models for discovering noisy instances, while other studies have developed methods for handling these noisy instances to make use of them instead of just eliminating them [4].

In this paper, we propose Error-Based Filtering (EBF), a simple and easy-to-implement method for supervised learning to monitor and analyze the distribution of values of a loss function for each instance during training in neural

The associate editor coordinating the review of this manuscript and approving it for publication was Michael Lyu.

network-based methods. The principal idea of this technique is to analyze the loss (training error) for each instance to determine outliers. During training, the technique attempts to determine a threshold for training losses and eliminates the instances that continue to have a loss above the threshold. Our empirical results show that the proposed EBF is effective at reducing the consequences of noise particularly in domains with a large ratio of noise.

The remainder of this paper is organized as follows. Section two sheds light on some of the related work that has been conducted to mitigate negative impacts of noise. EBF is discussed and explained in Section 3, and the empirical results are presented in Section 4. Section 5 concludes the paper.

II. RELATED WORK

Several studies have been conducted to mitigate the consequences of noise on learning. Some perform filtering as pre-processing for data cleansing [2], [7], [12]. Others develop model-based filtering techniques for mitigating the impact of noise during the training procedure [5], [6], [8], [9], [13]–[16].

In terms of model-based filtering methods, various approaches have achieved state-of-the-art results in mitigating the negative impact of noise. Reed and Lee [17] proposed a bootstrapping technique for fixing noisy labels by using labels predicted by a neural network and the consistency of prediction as a weighted measure for correct labels, then back-propagating the model accordingly. Patrini *et al.* [18] proposed the use of F-correction to correct a network's prediction using a noise transition matrix estimated using a standard network trained initially. Goldberger and Ben-Reuven [19] presented S-model based on the constitution of a noise transition matrix used by an additional softmax layer.

Another promising direction for handling noisy data is training on a selected part of instances that have small loss during training. Technically, neural network-based models tend to learn simple instances first, then learning all samples gradually [20]. Thus, these selected instances with small loss are potentially the clean ones. In other words, this direction is based on investigating whether a training instance is noisy based on tracking its loss (error) values during training. However, only a limited number of studies have been conducted in this direction.

Malach and Shalev-Shwartz [21] proposed a method they called Decoupling based on using two classifiers and updating the parameters based on samples that make different predictions from the two classifiers. Jiang *et al.* [22] proposed MentorNet based on training two networks, a teacher and a student. The teacher network is pre-trained to filter out noisy instances. Then, the filtered data are fed into the student network for training to be used later for classification.

Another method called Co-teaching was proposed by Han *et al.* [23] based on instance selection during training. Two networks are trained simultaneously to identify clean instances and teach each other. This differs from Decoupling through the procedure of updating, in which each network in Co-teaching feeds its peer network with the identified clean instances to be used for training. Clean instances are identified after every mini-batch processing because it is assumed that those instances with small loss represent clean instances.

Wei *et al.* [24] introduced a method they called JoCoR that also uses two networks to identify small-loss instances as clean instances. In particular, two networks are trained simultaneously with one joint loss that involves a regularization term to reduce the diversity between the two networks.

Our approach is motived by the idea of specifying noisy labels during training in an effortless manner. Methods based on the selection of instances during training such as Decoupling, MentorNet, Co-teaching, and JoCoR use two networks. In contrast, EBF does not use any additional network or change the network architecture. In other words, EBF does not rely on particular network architecture, and it can be implemented regardless of a model's peculiarities.

In addition, the methods based on instance selection (Decoupling, MentorNet, Co-teaching, and JoCoR) share the key idea of identifying clean instances based on having small loss during training. In contrast, our method deals with identifying noisy instances and thus is based on having extremely large loss. Technically, neural network models can cope with a limited number of noisy instances in the training data. It is shown in [23] that the standard convolutional neural network (CNN) model can cope with small ratio of noisy data. Consequently, EBF aims to reduce the amount of noise in the training data instead of selecting small-loss instances as clean.

EBF is simple and easy to implement and also offers an opportunity to make use of identified noisy instances to improve learning by using a semi-supervised technique.

III. ERROR-BASED FILTERING

In the early stage of training epochs, noisy instances tend to have larger loss values than clean instances until the classifier overfits those noisy instances [20]. We assume that it is possible to determine outliers statistically by observing the loss values for every instance during training steps (iterations).

Our approach is based on the hypothesis that the values that are normally distributed lie within a number of standard deviations, σ, from the mean μ. In particular, 68% of values are within one σ distance from μ, and 95% of values are within 2σ distances from μ [25] (see Figure 1). Therefore, if a set of data is noise free, then around 2.5% of the values will be greater than $\mu + 2\sigma$ and will fall in the red region in Figure 1. For instance, if we have a set of 200 data points, it is expected that five of them (2.5% of the data) will have values greater than $\mu + 2\sigma$. However, if the count of values that are greater than $\mu+2\sigma$ is greater than 2.5% of the entire data set, then we potentially have outliers.

Similarly, we assume that the average losses for all training instances over a number of training epochs constitutes a normally distributed set of values, and thus 95% of them lie within two standard deviations of the mean. Therefore, if the

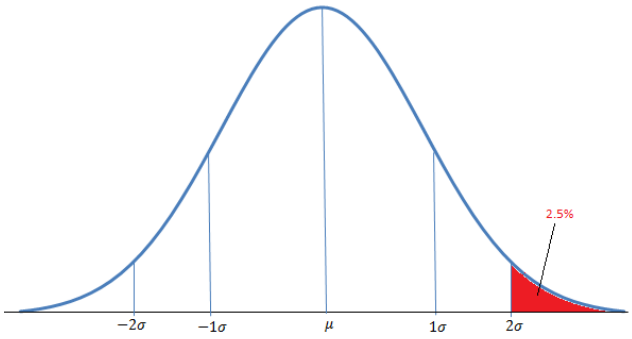

FIGURE 1. Standard normal distribution.

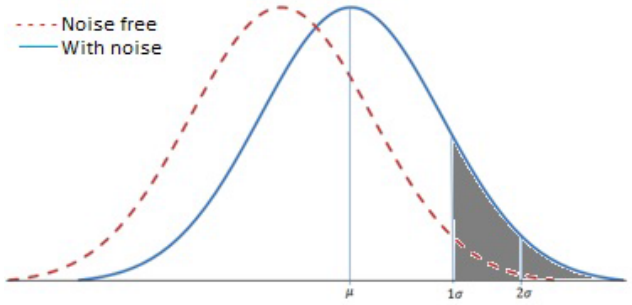

FIGURE 2. Distribution of data with and without outliers.

number of values in the red region exceeds 2.5% of the total number of instances, then those values are assumed to be outliers. It is worth noting that there are some statistical tests can be used to check if a set of values are normally distributed, such as the Shapiro-Wilk test, the Kolmogorov-Smirnov test, and the Anderson-Darling test [26].

However, since a large quantity of noise would cause a growth in the loss values and shifts the curve (distribution of values) to the right, then we consider the 2.5% of the instances with average loss above $\mu + \sigma$ instead of $\mu + 2\sigma$, as shown in Figure 2. In other words, 2.5% of the instances are expected to have average loss values that are greater than $\mu + \sigma$ while any extra instances are expected to be outliers and should, therefore, be eliminated.

Moreover, because the loss value for each instance changes over time as the training process progresses, and to assign more weight to the most recent loss values, we compute the exponential moving average (EMA) [27], [28] for each instance during m steps of training (iterations), as

$$V_0 = 0 \quad V_t = \beta_0 V_{t-1} + \beta_1 \theta_t \tag{1}$$

where V_t is the EMA at time period t (training step number), the parameters β_0 and β_1 are set as 0.9 and 0.1, respectively as was determined empirically in our experiments. θ_t is the loss value of an instance a time period t.

Each instance will have its own EMA value, as shown in Figure 3. Then, we use the EMA of each instance to determine the average EMA value of all instances, μ, and the standard deviation, σ.

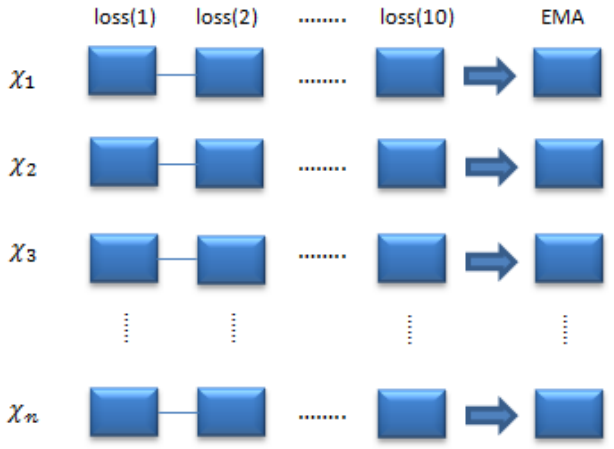

FIGURE 3. Each instance, χ_i, is represented by an EMA value obtained by implementing Equation (1) on ten loss values. Every loss value of (x) is collected after 16 steps of training.

Letting n be the total number of training instances and k the number of instances with an EMA value greater than $\mu + \sigma$, we eliminate d instances, where $d = k - 0.025n$.

Recall that the d instances are determined based on their EMA values. To determine the instances that will be removed as outliers, we sort the elements that are greater than $\mu + \sigma$ (k instances) in ascending order and remove the d instances with greater EMA values.

For efficiency and flexibility, EBF is not performed in every iteration but rather it is performed periodically every round where a round consists of a number of training steps (iterations). The number of rounds in which EBF is performed depends on the type and rate of noise in the training dataset. EBF is performed in the early stages of training; so that, noisy instances are discarded before affecting parameters of the training model. To reduce the overhead on the training time, we determine the d noisy instances during every round of EBF (not every epoch). In particular, EBF is performed for a number of rounds as described in Algorithm 1. During each round, ten loss values for each instance are collected and then the EMA of the losses is calculated, using Equation (1). The loss values are taken every 16 steps of training (iterations), as seen in Figure 3, thereby giving the learning model an opportunity to learn and update its parameters before EBF is performed. As a result, each training instance has an EMA value. Hence, from the values of EMA, the mean and the standard deviation are computed and used to determine the threshold and the number of outliers to eliminate. At the end of each round the outliers are determined and discarded from the training set.

IV. EXPERIMENTS

In this study, we follow the experimental methodology used in [23], thereby enabling a comparison of our results with the methods and results reported there. Han *et al.* [23] provides a comparison of six methods, Bootstrap [17], S-model [19], F-correction [18], Decoupling [21], MentorNet [22], and

Algorithm 1 EBF Algorithm

Input: parameter rounds, losses are collected every 16 training steps, training data X, and batch size is 512 during EBF process;
for each round r **do**:
  Create a two dimensional array 'losses' whose dimensions have lengths 'length of X' and 10
  for $m = 1..160$ **do**: *//m : number of the training step*
    Train network(mini-batch(X))
    if m % 16 = 0 **then**: *// losses are collected every 16 steps of training*
      for each instance i **do:**
        Let $\mathbf{err_i}$ = **the losss value for i**
        $\mathbf{losses_i} = \mathbf{losses_i} \cup \mathbf{err_i}$
      end for
    end if
  end for
  for each instance i do
    let $\mathbf{ema_i}$= **the EMA for** $\mathbf{losses_i}$
  end for
  Let μ be the average of all $\mathbf{ema_i}$
  Let σ be the standard deviation of all $\mathbf{ema_i}$
  Let k *be the number of instances* with EMA above $\mu + \sigma$
  Let $d = k - 0.025n$, where n is the number of training instances
  Remove d instances, with the largest EMA, from X
end for
Output $X_{clean\ labeled}$ **and** $X_{noisy\ labeled}$

Co-teaching [23] (all of these were discussed in Section II). Technically, the methods were implemented using one model architecture and against various rates and types of noise. Hence, to conduct a fair comparison of EBF with the methods reported in [23], we use the same datasets, model architecture, and procedure for corrupting data with noise.

Model Architecture. Following [23], a nine-layer CNN model is used in the implementation, as shown in Table 1. The model is implemented with a Leaky-ReLU (LReLU) activation function [29]. The Adam optimizer [27] is used with an initial learning rate of 0.001. The model is trained for 200 epochs with a batch size of 512 during the rounds of EBF, while the batch size is set to 128 after EBF is accomplished. Dropout [30] and batch normalization [31] are also used. Batch normalization is a technique for stabilizing the learning process as it works on standardizing the inputs to each layer of the model [31]. All of the experiments were conducted on AWS GPU 61GiB RAM.

Datasets used for evaluation. As in [23], three benchmark datasets were used to evaluate the effectiveness of EBF. These datasets include Digits Recognition MNIST [32], object recognition of ten classes CIFAR10 [33], and object recognition of one hundred classes CIFAR100 [33]. A brief description of the datasets is provided in Table 2.

Labels of training sets are corrupted deliberately while testing sets are kept clean to measure the efficacy of the proposed method in mitigating the effect of noise. To add noise, we follow the same procedure implemented in [23] in which two types of noise transition matrices were used, symmetry flipping and pair flipping. In symmetry flipping [34], some of the labels of each class are flipped to the labels of the other classes with the same ratio as defined in the transition matrix Figure 4(b), where in pair flipping [23], some of the labels of each class are flipped to the labels of another specified class as defined in the transition matrix Figure 4(a). As in [23], the ratios of noise were 20% and 50% for symmetry type $\varepsilon = \{0.2, 0.5\}$; and 45% for pair type $\varepsilon = 0.45$. Hence, we conducted several experiments with different noise ratios to measure the actual impact of noise and the potential mitigation of EBF on classification accuracy. The percentages of corrupted labels used were 20% and 50% of noise in the symmetry type and 45% of noise in the pair type.

A. PERFORMANCE OF EBF IN NOISE IDENTIFICATION

In this section, we evaluate EBF in terms of its ability to identify noisy instances, in addition to the precision that measures the true positives in EBF. Precision represents the number of instances that are correctly identified by EBF as noisy (true positive) divided by the number of instances identified by EBF as noisy (true positive + false positive) multiplied by 100.

EBF begins working as the first epoch of training is completed to guarantee that all training instances have been passed to the model at least once. The number of EBF rounds is a hyper-parameter specified in advance; thus, it is changeable according to the type (symmetry or pair) and rate of noise in a dataset. For symmetry noise, EBF was repeated for

$$\begin{bmatrix} 1-\varepsilon & \varepsilon & 0 & 0 & 0 \\ 0 & 1-\varepsilon & \varepsilon & 0 & 0 \\ 0 & 0 & 1-\varepsilon & \varepsilon & 0 \\ 0 & 0 & 0 & 1-\varepsilon & \varepsilon \\ \varepsilon & 0 & 0 & 0 & 1-\varepsilon \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1-\varepsilon & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} \\ \frac{\varepsilon}{C-1} & 1-\varepsilon & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} \\ \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & 1-\varepsilon & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} \\ \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & 1-\varepsilon & \frac{\varepsilon}{C-1} \\ \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & \frac{\varepsilon}{C-1} & 1-\varepsilon \end{bmatrix}$$

(b)

FIGURE 4. Definition of noise transition matrix for five classes as an example. ε stands for noise ratio, and C is the number of classes. (a) Transition matrix of pair flipping. (b) Transition matrix of symmetry flipping.

TABLE 1. Model architecture used in our experiments following [23].

CNN on MNIST	CNN on CIFAR-10	CNN on CIFAR-100
28 × 28 Gray Image	32 × 32 RGB Image	32 × 32 RGB Image
3 × 3 conv, 128 LReLU 3 × 3 conv, 128 LReLU 3 × 3 conv, 128 LReLU		
2 × 2 ma × -pool, stride 2		
dropout = 0.25		
3 × 3 conv, 128 LReLU 3 × 3 conv, 128 LReLU 3 × 3 conv, 128 LReLU		
2 × 2 ma × -pool, stride 2		
dropout = 0.25		
3 × 3 conv, 128 LReLU 3 × 3 conv, 128 LReLU 3 × 3 conv, 128 LReLU		
avg-pool		
Dense 128-> 10	Dense 128-> 10	Dense 128-> 100

three and six rounds for 20% and 50% noise, respectively. However, for 45% pair noise, which turned-out to be the more challenging type of noise, we performed EBF for eight rounds. As shown in Table 3, EBF was extremely effective in identifying noisy instances in MNIST. EBF was able to identify more than 98% of the outliers (the recall value) of the symmetry type with 20% and 50% noise. However, with 45% pair noise, EBF detected 94.01% of the noisy instances. The rate of false positives in noise identification was extremely low for the symmetry type of noise, as can be seen from the high precision values. In the case of pair noise, the precision was 82.1%, indicating that the number of false positives were relatively high.

TABLE 2. Overview of the datasets used in this paper.

Dataset	Dimensionality	Training set	Testing set
MNIST	28 × 28	60K	10K
CIFAR10	32 × 32	50K	10K
CIFAR100	32 × 32	50K	10K

Regarding CIFAR10, which includes RGB images of three levels, EBF showed a remarkable performance in identifying outliers of the symmetry type. In particular, with 20% of symmetry noise, EBF identified outliers with 96.89% recall value; however, with a relatively low precision value 56.85%, indicating a high ratio of false positives. While with 50% of symmetry noise, EBF identified outliers with 94.42% recall value and with a lower ratio of false positives, as reflected by the higher precision value of 79.57%. It stands to reason to have a better precision in identifying outliers when the noise ratio is high, because the ratio of false positives is likely to go down as the noise ratio goes up. However, with the 45% pair noise, outliers were identified at 82.7% accuracy and 59.22% precision, indicating a high ratio of false positives; see Table 3.

EBF works with CIFAR100 almost as with CIFAR10. With 50% symmetry noise, it was able to identify outliers with 90.94% recall value and a relatively high precision (a low ratio of false positives). At 20% symmetry noise, it identified outliers at 94.79% recall but at relatively low precision of 53.79%, indicating that it mistakenly identified a large percentage of genuine instances as outliers. With 45% pair noise, the recall is 73.69% and the precision is 50.10%, which reveals that this type of noise is more challenging than symmetry noise.

In general, we can notice that EBF is able to detect most of the noisy labels in our experiments. However, although the precision values are good in case of MNIST, they degraded in case of CIFAR10 and CIFAR100. This can be attributed to the fact that MNIST includes gray images of one level; while CIFAR10 and CIFAR100 include colored images of three levels, which may make identifying all outliers a more challenging task for EBF.

B. CLASSIFICATION RESULTS

In this section, we present the results of classification accuracy to evaluate the performance of the model before and after implementing EBF. Recall that EBF is performed during the early stages of training which implies that outliers are discarded early before overfitting occurs.

TABLE 3. Results of EBF in noise detection.

Dataset	MNIST			CIFAR10			CIFAR100		
Noise type & rate	20% Symm.	50% Symm.	45% Pair	20% Symm.	50% Symm.	45% Pair	20% Symm.	50% Symm.	45% Pair
Recall (%)	98.45	99.45	94.01	96.89	94.42	82.70	94.79	90.94	73.69
Precision (%)	98.67	98.01	82.1	56.85	79.57	59.22	53.79	78.70	50.10

TABLE 4. Average test accuracy on MNIST over the last ten epochs. The top part presents the results adopted from [23], and the bottom part presents the results achieved by our implementation. The asterisk (∗) indicates the methods based on selected instances during training.

Method	Standard	Bootstrap	S-model	F-correction	Decoupling*	MentorNet*	Co-teaching*	EBF*	EBF+VAT
20% Symmetry Noise Rate	94.05 (±0.16)	94.40 (±0.26)	98.31 (±0.11)	98.80 (±0.12)	95.70 (±0.02)	96.70 (±0.22)	97.25 (±0.03)	98.75 (±0.29)	**99.34** (±0.00)
50% Symmetry Noise Rate	66.05 (±0.61)	67.55 (±0.53)	62.29 (±0.46)	79.61 (±1.96)	81.15 (±0.03)	90.05 (±0.30)	91.32 (±0.06)	98.27 (±0.39)	**98.97** (±0.00)
45% Pair Noise Rate	56.52 (±0.55)	57.23 (±0.73)	56.88 (±0.32)	0.24 (±0.03)	58.03 (±0.07)	80.88 (±4.45)	87.63 (±0.21)	88.91 (±0.62)	**96.19** (±0.00)

TABLE 5. Average test accuracy on CIFAR10 over the last ten epochs. The top part shows the results in [23], and the bottom part shows the results achieved by our implementation. The asterisk (∗) indicates the methods based on selected instances during training.

Method	Standard	Bootstrap	S-model	F-correction	Decoupling*	MentorNet*	Co-teaching*	EBF*	EBF+VAT
20% Symmetry Noise Rate	76.25 (±0.28)	77.01 (±0.29)	76.84 (±0.66)	84.55 (±0.16)	80.44 (±0.05)	80.76 (±0.36)	82.32 (±0.07)	**85.58** (±0.58)	85.27 (±0.61)
50% Symmetry Noise Rate	48.87 (±0.52)	50.66 (±0.56)	46.15 (±0.76)	59.83 (±0.17)	51.49 (±0.08)	71.10 (±0.48)	74.02 (±0.04)	74.30 (±1.26)	**82.66** (±0.40)
45% Pair Noise Rate	49.50 (±0.42)	50.05 (±0.30)	48.21 (±0.55)	6.61 (±1.12)	48.80 (±0.04)	58.14 (±0.38)	**72.62** (±0.15)	59.17 (±1.91)	72.21 (±0.31)

TABLE 6. Average test accuracy for CIFAR100 over the last ten epochs. The top part shows the results in [23], and the bottom part shows the results achieved using our implementation. The asterisk (∗) indicates the methods based on selected instances during training.

Method	Standard	Bootstrap	S-model	F-correction	Decoupling*	MentorNet*	Co-teaching*	EBF*	EBF+VAT
20% Symmetry Noise Rate	47.55 (±0.47)	47.00 (±0.54)	41.51 (±0.60)	**61.87** (±0.21)	44.52 (±0.04)	52.13 (±0.40)	54.23 (±0.08)	59.90 (±0.66)	59.48 (±0.53)
50% Symmetry Noise Rate	25.21 (±0.64)	21.98 (±6.36)	18.93 (±0.39)	41.04 (±0.07)	25.80 (±0.04)	39.00 (±1.00)	41.37 (±0.08)	**48.51** (±0.61)	47.05 (±0.56)
45% Pair Noise Rate	31.99 (±0.64)	32.07 (±0.30)	21.79 (±0.86)	1.60 (±0.04)	26.05 (±0.03)	31.60 (±0.51)	**34.81** (±0.07)	32.65 (±0.60)	33.52 (±0.11)

Since EBF is implemented regardless of the model architecture, those noisy instances identified by EBF can be useful for improving the performance if used through semi-supervised learning (SSL) methods. In other words, SSL can be applied as EBF is finished to take advantage of those identified noisy instances and thus learn from both clean instances (labeled) and noisy instances (unlabeled). To evaluate the efficacy of this approach, we first implemented EBF individually and then implemented it again using a semi-supervised method.

Virtual adversarial training (VAT) [35] is a regularization method for SSL that is used together with EBF. Our choice of VAT was based on the evaluation study [36], which revealed that VAT outperformed various other SSL methods. Furthermore, similar to EBF, VAT can be used regardless of the model architecture, thereby making VAT a compatible method for use with EBF.

1) RESULTS ON MNIST

The MNIST test set, which is totally clean of noise, was used for evaluating the performance of the selected methods. The test accuracy achieved for each method is shown in Table 4. It is clear from the table that with 20% noise, all methods including the standard method, which does not eliminate noisy, gave reasonable results. The standard method was able to achieve 94.05% accuracy in classifying digits. However, EBF gave the second-best result of 98.75%, next to the F-correction method which achieved 98.80% accuracy. Moreover, when EBF was coupled with VAT, it outperformed all of the other methods with 99.34% accuracy.

With 50% noise, the performance of all methods substantially degraded except EBF and EBF+VAT methods, which achieved excellent accuracies of 98.27% and 98.97%, respectively. It is worth noting here that the methods that identify outliers on the basis of selected instances during training (marked with an asterisk *), which include in addition to EBF, Decoupling, MentorNet, and Co-teaching, achieved better results than all other methods, with classification accuracy above 80%. EBF also outperformed all of the other methods by achieving 98.27%, approximately 7% greater than the second best result. EBF coupled with VAT obtained the best accuracy of 98.97%.

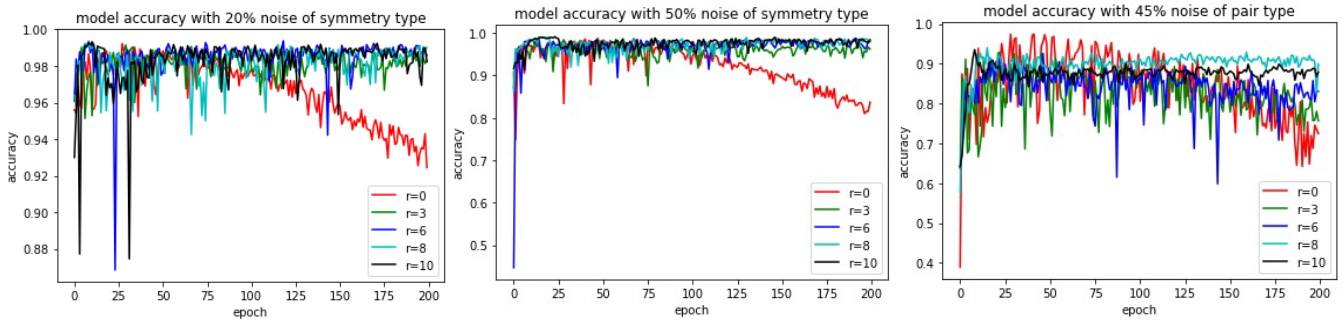

FIGURE 5. Model test accuracy with different number of EBF rounds on MNIST, r denotes to the number of rounds that EBF is implemented.

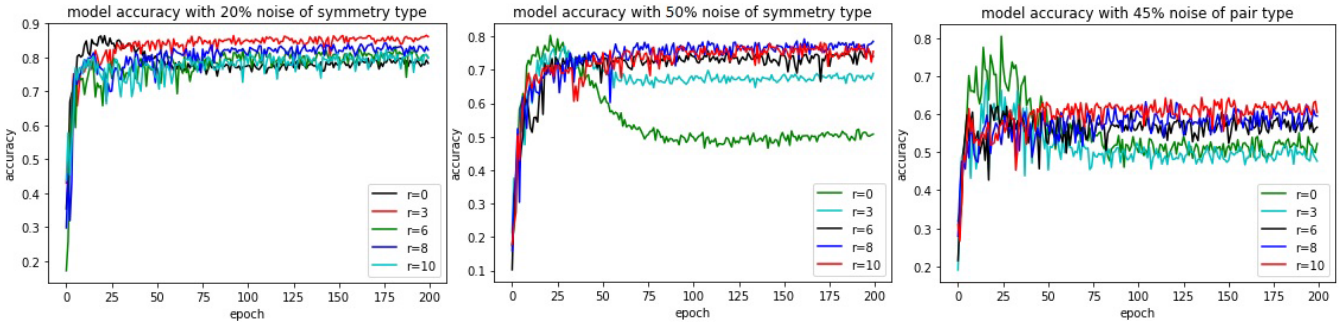

FIGURE 6. Model test accuracy with different number of EBF rounds on CIFAR10, r denotes to the number of rounds that EBF is implemented.

Similarly, in the more challenging case of 45% pair noise EBF and EBF+VAT achieved the best accuracy results of 88.91% and 96.19%, respectively. Once again, the methods that identify noise based on selected instances during training achieved better results than all other methods.

2) RESULTS ON CIFAR10

We applied ZCA whitening on training and testing data of CIFAR10 as a pre-process of the conducted experiment. Whitening is a common technique implemented to normalize inputs such as images, which accelerates training. Technically, it linearly transforms the input variables to equate their covariance with the identity matrix.

Classification accuracies on the test dataset obtained by the different methods are shown in Table 5. The table reveals that EBF and EBF+VAT achieve the best and second-best results of 85.58% and 85.27%, respectively. When the noise rate increases to 50%, performance of all methods degrade as can be expected. However, once again EBF and EBF+VAT achieve the best results of 74.30% and 82.66%, respectively. The large difference between the performance of EBF and EBF+VAT once again indicates that VAT made good use of the outliers. As for the 45% pair noise, the accuracy of most methods, including EBF, fell below 60%. The only exceptions were Co-teaching and EVF+VAT which achieved comparable results of 72.62% and 72.21%, respectively.

3) RESULTS ON CIFAR100

We also applied ZCA whitening on training and testing data of CIFAR100 as a pre-process of the conducted experiment. Since neural network-based models tend to learn to classify simple instances first, then gradually learn the more difficult instances [20] and because CIFAR100 presents a more difficult challenge than MINST and CIFER10 in terms of the number of classes, EBF begins after the second epoch instead of the first, thereby providing the model with a better opportunity to learn the genuine instances before starting to eliminate noise. Table 6 summarizes the accuracy results of all methods.

The table reveals that at 20% symmetry noise F-correction and EBF achieve the best and second best results of 61.87% and 59.90%, respectively. With 50% symmetry noise, EBF and EBF+VAT achieved the best and second-best results of 48.51% and 47.05%, respectively. With 45% pair noise, the hardest case, F-correction totally fails, while Co-teaching and EBF+VAT achieving the best and second-best results of 34.81% and 33.52%, respectively. The poor results achieved by all methods in the case of CIFAR100 can be attributed to the fact that the dataset contains 100 classes which make it a challenging problem and adding a large amount of noise makes it even an extremely difficult problem for all methods.

EBF is performed periodically as multiple rounds during training according to the type (symmetry or pair) and rate of noise in the training dataset. From Figures 5, 6 and 7, when EBF is not performed ($r = 0$), it is clear that after a set of training epochs the classification accuracy decreases dramatically as a result of noise. Hence, rounds of EBF are performed in the early stages of training; therefore, noisy instances are discarded before affecting parameters of the training model.

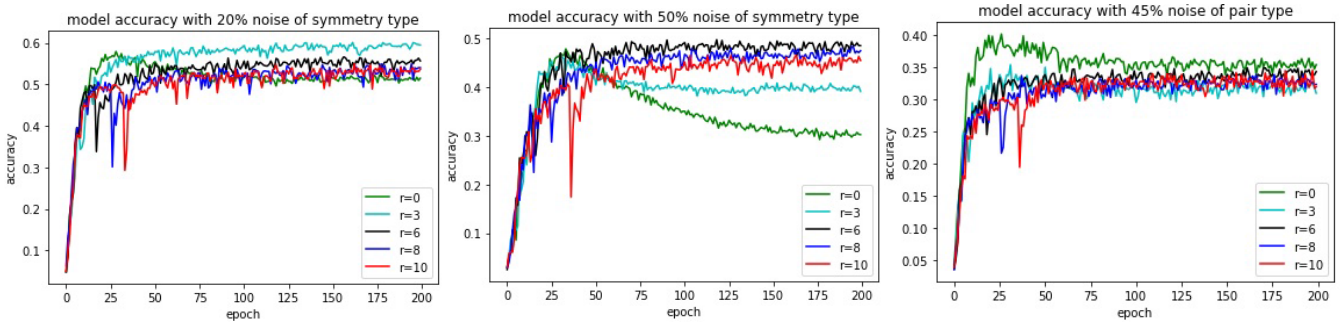

FIGURE 7. Model test accuracy with different number of EBF rounds on CIFAR100, r denotes to the number of rounds that EBF is implemented.

Throughout our experiments, we set the rounds of EBF as 3 rounds with 20% noise, 6 rounds with 50% noise and 8 rounds with 45% noise of pair type. However, Figures 5, 6 and 7 show that the performance with more or less rounds of EBF still show some improvements compared with not using EBF at all.

We can also notice that with 20% of symmetry noise, three rounds of EBF were sufficient for mitigating the effect of noise. Furthermore, although we implemented six rounds of EBF with 50% of symmetry noise throughout our experiments in Table 4, 5 and 6; ten rounds of EBF on MNIST and eight rounds of EBF on CIFAR10 achieved better results. Also with 45% of pair noise on CIFAR10, ten rounds performed slightly better showing a stable performance. This can be attributed to the fact that after a few rounds the EMA of instances get closer to the mean with very few instances with EMA above the threshold. Therefore, very few instances get identified as outliers.

The performance of EBF on CIFAR100 with 45% pair noise was not very good. This the most challenging case because of the large number of classes in CIFAR100 (100 classes) and the type of noise, pair noise, where the classes of randomly selected pairs of instances were swapped.

V. CONCLUSION

Noise in data can appear in various forms resulting in negative consequences on learning such as increasing the required number of training instances, increasing the required number of training epochs, and overfitting. In this paper, we introduced the EBF method for noise detection and elimination during training. EBF eliminates a set of instances that are assumed to be outliers by monitoring the loss values of the training instances and identifies noisy instances statistically during the training procedure. Eliminating those instances revealed that classification accuracy increased, as shown empirically. The empirical results of EBF have shown a positive impact on training models that are based on artificial neural networks, especially with large noise ratios. Furthermore, we also used EBF with SSL which again showed promising results.

In the future, we intend to investigate further techniques that maximize loss values for noisy labels to assist EBF in catching outliers and reducing the ratio of false positives.

ACKNOWLEDGMENT

The authors would like to thank Researchers Support and Services Unit (RSSU) for their technical support.

REFERENCES

[1] X. J. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, 2005.

[2] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *J. Artif. Intell. Res.*, vol. 11, pp. 131–167, Aug. 1999.

[3] C. M. Teng, "A comparison of noise handling techniques," in *Proc. FLAIRS Conf.*, 2001, pp. 269–273.

[4] B. Frenay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, May 2014.

[5] X. Zeng and T. Martinez, "A noise filtering method using neural networks," in *Proc. IEEE Int. Workshop Soft Comput. Techn. Instrum., Meas. Rel. Appl. (SCIMA)*, May 2003, pp. 26–31.

[6] V. Mnih and G. Hinton, "Learning to label aerial images from noisy data," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 567–574.

[7] K. El Hindi and M. Al-Akhras, "Smoothing decision boundaries to avoid overfitting in neural network training," *Neural Netw. World*, vol. 21, no. 4, pp. 311–325, 2011.

[8] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1196–1204.

[9] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Learning from noisy labels with deep neural networks," 2014, *arXiv:1406.2080*. [Online]. Available: https://arxiv.org/abs/1406.2080

[10] M. Pechenizkiy, A. Tsymbal, S. Puuronen, and O. Pechenizkiy, "Class noise and supervised learning in medical domains: The effect of feature extraction," in *Proc. 19th IEEE Symp. Comput.-Based Med. Syst. (CBMS)*, Jun. 2006, pp. 708–713.

[11] K. El Hindi and M. Alakhras, "Eliminating border instance to avoid overfitting," in *Proc. Intell. Syst. Agents*, 2009, pp. 93–99.

[12] P. Jeatrakul, K. W. Wong, and C. C. Fung, "Data cleaning for classification using misclassification analysis," *J. Adv. Comput. Intell. Informat.*, vol. 14, no. 3, pp. 297–302, Apr. 2010.

[13] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2691–2699.

[14] 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.

[15] H. Masnadi-Shirazi and N. Vasconcelos, "On the design of loss functions for classification: Theory, robustness to outliers, and savageboost," in *Proc. NIPS*, 2009, pp. 1049–1056.

[16] A. Vahdat, "Toward robustness against label noise in training deep discriminative neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5596–5605.

[17] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," 2014, pp. 1–11, *arXiv:1412.6596*. [Online]. Available: https://arxiv.org/abs/1412.6596

[18] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. CVPR*, Jul. 2017, pp. 1944–1952.
[19] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *Proc. ICLR*, 2017, pp. 1–9.
[20] D. Arpit, S. Jastrzebski, 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 *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 233–242.
[21] E. Malach and S. Shalev-Shwartz, "Decoupling 'when to update' from 'how to update,'" in *Proc. NIPS*, 2017, pp. 960–970.
[22] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proc. ICML*, 2018, pp. 2304–2313.
[23] 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. (NIPS)*, 2018, pp. 8527–8537.
[24] H. Wei, L. Feng, X. Chen, and B. An, "Combating noisy labels by agreement: A joint training method with co-regularization," 2020, *arXiv:2003.02752*. [Online]. Available: https://arxiv.org/abs/2003.02752
[25] D. J. Wheeler and D. S. Chambers, *Understanding Statistical Process Control*. USPC, 1992.
[26] H. C. Thode, *Testing for Normality*, vol. 164. Boca Raton, FL, USA: CRC Press, 2002.
[27] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, pp. 365–380, vol. 5, no. 2, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[28] M. J. Pring, *Study Guide for Technical Analysis Explained*. New York, NY, USA: McGraw-Hill, 2014.
[29] A. L. Mass, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2003, pp. 1–6.
[30] 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. 1929–1958, 2014.
[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167
[32] *UCI Machine Learning Repository: Optical Recognition of Handwritten Digits Data Set*. Accessed: Feb. 5, 2020. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
[33] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE 12th Int. Conf. Comput. Vis. (ICCV)*, Sep. 2009, pp. 2146–2153.
[34] B. Van Rooyen, A. Menon, and R. C. Williamson, "Learning with symmetric label noise: The importance of being unhinged," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 10–18.
[35] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1979–1993, Aug. 2019.
[36] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3235–3246.

FAHAD ALHARBI received the B.Sc. degree in computer science from Qassim University, Saudi Arabia, and the M.Sc. degree in advanced software engineering from the University of Leicester, U.K. He is currently pursuing the Ph.D. degree with the Department of Computer Science, King Saud University. He is also a Lecturer with the Faculty of Science, Qassim University. His research interest includes outlier detection in the algorithms of deep learning.

KHALIL EL HINDI is currently a Professor with the Department of Computer Science, King Saud University. His main research interests include machine learning, classification algorithms, outlier detection, instance weighing, ensembles of classifiers, similarity distance metrics, and neural networks.

SAAD AL-AHMADI is currently an Assistant Professor of computer science with the College of Computer and Information Sciences, King Saud University. He has published many papers in many journals and conferences. His research interests include cybersecurity, artificial intelligence for computer security, future generation networks, the IoT, and sensors networks.