Label Cleaning with a Likelihood Ratio Test

Anonymous authors
Paper under double-blind review

Abstract

To collect large scale annotated data, it is inevitable to introduce label noise, i.e., incorrect class labels. A major challenge is to develop robust deep learning models that achieve high test performance despite training set label noise. We introduce a novel approach that directly cleans labels in order to train a high quality model. Our method leverages statistical principles to correct data labels and has a theoretical guarantee of the correctness. In particular, we use a likelihood ratio test (LRT) to flip the labels of training data. We prove that our LRT label correction algorithm is guaranteed to flip the label so it is consistent with the true Bayesian optimal classifier with high probability. We incorporate our label correction algorithm into the training of deep neural networks and train models that achieve superior testing performance on multiple public datasets.

1 Introduction

Label noise is ubiquitous in real world data. It may be caused by unintentional mistakes of manual or automatic annotators (Yan et al., 2014; Veit et al., 2017). It may also be introduced by malicious attackers (Steinhardt et al., 2017). Noisy labels impair the performance of a model (Smyth et al., 1994; Brodley & Friedl, 1999), especially a deep neural network, which tends to have strong memorization power (Ernay & Verleysen, 2014; Zhang et al., 2017). Improving the robustness of a model to label noise is a crucial yet challenging task in many applications (Mnih & Hinton, 2012; Wu et al., 2018). Existing methods mainly follow two directions, probabilistic reasoning and data selecting.

Probabilistic methods explicitly model a noise transition matrix, namely, the probability of one label being corrupted into another (Goldberger & Ben-Reuven, 2017; Patrini et al., 2017). The transition matrix is often estimated from the data, and is used to re-calibrate the training loss or to correct the prediction. Explicit estimation of the transition matrix can be problematic due to the large variation of noise patterns, e.g., uniform noise, asymmetric noise, or mixtures. Furthermore, the transition matrix size is quadratic to the number of classes, making the estimation task prohibitive when the data has hundreds or even thousands of classes.

Data-selecting methods are agnostic of the underlying noise pattern. These methods gradually collect clean data whose labels are trustworthy (Malach & Shalev-Shwartz, 2017; Jiang et al., 2018; Han et al., 2018). As more clean data are collected, the quality of the trained models improves. The major issue of these methods is the lack of a quantitative control of the quality of the collected clean data. Without a principled guideline, it is hard to find the correct data collection pace. An aggressive selection can unknowingly accumulate irreversible errors. On the other hand, an overly-conservative strategy can be very slow in training, or stops with insufficient clean data and mediocre models.

We propose a novel method with the benefit from both the probabilistic and the data-selecting approaches. Similar to data-selecting methods, our method continuously improves the purity of the data labels by correcting the noise-corrupted ones. Meanwhile, we improve the classifier using the updated labels. Our label correction algorithm is based on statistical principles and is theoretically guaranteed to deliver a high quality label set. Instead of explicitly estimating the transition matrix, the correction algorithm only depends on the prediction of the current model, denoted as \( f \). Using an \( f \)-based likelihood ratio test, we determine whether the current label of each data should be corrected. Our main theorem proves that the label correction algorithm will clean a majority of noisy labels with high probability.

In practice, we incorporate the label correction algorithm into the training of deep neural networks. Our method iteratively updates the labels of the data while continuously training a deep neural
network. To ensure the deep neural network does not overfit with noise labels that are yet to be corrected, we introduce a new retroactive loss term that regulates the model by enforcing its consistency with models in previous epochs. The rationale is that the model in an earlier training stage tends to fit the true signal rather than noise, although its overall performance is sub-optimal. Through experiments on various datasets with various noise patterns and levels, we show that our method produces robust neural network models with superior performance.

To the best of our knowledge, our method is the first to correct labels with theoretical guarantees. It has advantages over both probabilistic methods and data-selecting methods. Compared with other data-selecting methods, it has a better quantitative control of the label quality and thus is less brittle when generalizing to different datasets and different noise patterns. Also note that we are not selecting clean data. Instead, we correct labels and always use the whole training set to train. This brings an additional advantage of fully leveraging the data. Compared with other probabilistic methods, our correction algorithm assumes a rather general family of underlying noise patterns and avoids an explicit estimation of the transition matrix.

1.1 Related Work

Recent works could be classified into three categories. One is to model and employ noise transition matrix to correct the loss. For example, [Patrini et al. (2017)] proposes to correct the loss function with estimated noise pattern. The resulting loss is an unbiased estimator of the ground truth loss, and enables the trained model to achieve better performance. However, such an estimator relies on strong assumptions and could be inaccurate in certain scenarios. [Reed et al. (2014)] considers modeling the noise pattern with a hidden layer. The learning of this hidden layer is regularized with a feature reconstruction loss, yet without a guarantee that the true label distribution is learned. Another method mentioned in their work is to minimize the entropy of neural network output; however, this method tends to predict a single class. To address this weakness, [Hendrycks et al. (2019)] proposes to utilize a small number of kosher data to pre-train a network and estimate the noise pattern. However, such clean data may not always be available in practice.

Another strategy to handle noisy label problem is to design models that are intrinsically robust to noisy data. [Crammer et al. (2009)] introduces a regularized confidence weighting learning algorithm (AROW), where parameters of classifiers are assumed normally distributed and the mean and covariance of this distribution is updated during training. The idea here is to preserve the weight distribution as much as possible while requiring the model to maintain predictive ability. In the follow-up work [Crammer & Lee (2010)] proposes to improve this algorithm by herding the updating direction via specific velocity field (NHERD), achieving better performance. Both of these works impose parametric constraint on parameters, which could prevent classifiers from adapting to complex data set. [Arpit et al. (2017)] shows that deep neural networks tend to learn meaningful patterns before they over-fit to noisy ones. Based on this observation, they propose to add Gaussian or adversarial noise to input when training with noisy labels, and empirically show that such data perturbation is able to make the resulting model more robust. Other commonly adopted techniques, such as weight decay and dropout, are also shown to be effective in increasing the robustness of trained classifier ([Arpit et al. 2017] [Zhang et al. 2017]). However, the intrinsic reasons for this phenomenon still remains unclear and overfitting to noisy label is still inevitable.

Apart from the above mentioned strategies, one recent work proposes to correct the corrupted labels during training. In particular, [Tanaka et al. (2018)] propose to jointly train the deep network and estimate the underlying true labels. While achieving improved performance, their method largely relies on the prior distribution and is difficult to deploy under cases where there is a large number of classes.

Finally, beyond deep learning framework, there are several theoretic works that demonstrate the robustness of a variety of losses to label noise ([Long & Servedio 2010] [Natarajan et al. 2013] [Ghosh et al. 2015] [van Rooyen et al. 2015]). Following the work of [Wang & Chaudhuri 2018], [Gao et al. (2016)] proposes an algorithm that can converge to the Bayesian optimal classifier under different noisy settings. Moreover, they discuss the performance of k-nearest neighbor (KNN) classifiers however, the problem with KNN is that it is computationally intensive and thus difficult to be incorporated into a learning context. Within the framework of deep learning, there are more efforts that need to be made to bridge theory and practice.
2 Method

Our method has two synchronized modules, the training module and the label correction module. The training module continues to learn a classifier based on the current labels. Meanwhile, the label correction module uses the prediction of the classifier to correct labels.

We start with some preliminaries necessary for the exposition (Section 2.1). In Section 2.2, we explain our correction algorithm. It uses the prediction of the classifier (trained on noisy labels) for a likelihood ratio test. Based on the test result, it decides whether to correct the label of a data. Theoretically, we prove that under certain assumptions of the prediction, \( f \), the algorithm will change the labels to the correct ones, i.e., the ones consistent with the Bayes optimal classifier (Theorem 1).

In Section 2.3, we present the overall training method. We incrementally train a deep neural network based on the corrected labels, \( \tilde{y} \). Meanwhile, the network’s prediction is used for label correction. To improve the quality of the labels to the correct ones, i.e., the ones consistent with the Bayes optimal classifier (Theorem 1), we introduce a new loss term, called the retroactive loss. The goal is to regulate the model using models trained in earlier epochs, as they may be less overfitting with corrupted labels.

For the label correction module, we focus on a binary classifier. But the algorithm and the theoretical results can easily be generalized to the multiclass setting (Corollary 1).

2.1 Preliminaries

Let \( \mathcal{X} \) be the feature space, \( \mathcal{Y} = \{0, 1\} \) be the label space, and \( D \) be an unknown distribution on \( \mathcal{X} \times \mathcal{Y} \). The joint probability can be factored as \( D(x, y) = \Pr(y|x)\Pr(x) \). We denote by \( \eta(x) = \Pr(y = 1|x) \) the true conditional probability. The Bayes risk of a classifier \( h : \mathcal{X} \to \mathcal{Y} \) is \( R(D, h) = \Pr(x, y) \sim D(h(x) \neq y) \). A Bayes optimal classifier is the minimizer of the Bayes risk, i.e., \( h^* = \arg \min_h R(D, h) \). It can be calculated using the true conditional probability, \( \eta \),

\[
    h^*(x) = 1_{\{\eta(x) > \frac{1}{2}\}}(x) = \begin{cases} 1 & \text{if } \eta(x) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}
\]

We assume the true conditional probability, \( \eta \), satisfies the Tsybakov condition (also called the TNC condition) [Tsybakov et al. 2004]. This condition stipulates that the uncertainty of \( \eta \) and thus the Bayes optimal classifier is bounded. This assumption in general helps to bound the margin of Bayesian decision rule such that the proposed classifier’s risk can be bounded accordingly.

**Definition 1** (Tsybakov Condition). There exist \( C > 0, \lambda > 0, \) and \( t_0 \in (0, \frac{1}{2}] \), such that for all \( t \leq t_0 \),

\[
    \Pr \left[ \eta(x) - \frac{1}{2} < t \right] \leq C e^{\lambda t}.
\]

The noisy label setting. Instead of samples from \( D \), we are given a sample set with noisy labels \( \mathcal{S} = \{(x, \tilde{y})\} \) where \( \tilde{y} \) is the possibly corrupted label based on the true label \( y \). We assume a transition probability \( \tau_{i \to j} = \Pr(\tilde{y} = j|y = i) \), i.e., the chance a ground truth label \( y \) is flipped from class \( i \) to class \( j \). For simplicity, we denote \( \tau_{ij} = \tau_{i \to j} \). The transition probabilities \( \tau_{01} \) and \( \tau_{10} \) are independent of the true joint distribution \( D \) and the feature \( x \). We denote the conditional probability of the noisy labels as \( \tilde{\eta}(x) = \Pr(\tilde{y} = 1|x) \). In short, we call \( \tilde{\eta} \) the noisy conditional probability. It is linearly to the true conditional probability, \( \eta \):

\[
    \tilde{\eta}(x) = (1 - \tau_{10})\eta(x) + \tau_{01}\eta(1 - \eta(x)) = (1 - \tau_{01} - \tau_{10})\eta(x) + \tau_{01}.
\]

2.2 The Label Correction Algorithm and its Theoretical Guarantees

Our label correction algorithm takes in a current neural network prediction \( f : \mathcal{X} \to [0, 1] \) (i.e., an estimation of \( \eta \) based on the noisy labels). For all training data and their current noisy label \( (x, \tilde{y}) \),
the correction algorithm uses $f$ to run a likelihood ratio test and to decide whether to flip the label according to the result. The goal of the likelihood test is to decide whether the null hypothesis, $H_0: \tilde{y} = y$, is true. If yes, $\tilde{y}$ is accepted as it is. Otherwise, we flip $\tilde{y}$, so that hopefully it becomes $y$. Formally, the likelihood ratio is defined as

$$LR(f, x, \tilde{y}) = \frac{f(x)^{\tilde{y}} (1 - f(x))^{1 - \tilde{y}}}{f(x)^{\tilde{y}} (1 - f(x))^{1 - \tilde{y}}} \quad (3)$$

We compare this likelihood ratio with a predetermined value $\Delta$. If $LR(f, x, y) \leq \Delta$, we reject the null hypothesis and flip the label $\tilde{y}_{new} = 1 - \tilde{y}$. Otherwise, the label remains unchanged, $\tilde{y}_{new} = \tilde{y}$.

See Figure 2 for an illustration of the algorithm.

Note that the constant $\Delta$ depends on the underlying noise pattern, $f$ and $\tilde{y}$. Below we show that if we choose $\Delta$ carefully, the label correction algorithm is guaranteed to make proper correction and clean most of the corrupted labels. However, in practice, $\Delta$ is unknown and needs to be tuned.

**Intuition.** In the likelihood ratio (Eq. 3), the numerator is the likelihood that the prediction $f$ is consistent with the noisy label $\tilde{y}$. The denominator is the likelihood of the opposite case. When this ratio is smaller than 1, we know that the prediction of $f$ is more likely to be inconsistent with $\tilde{y}$. But whether $f$ agrees with $\tilde{y}$ is not the hypothesis to test. To test the intended null hypothesis ($\tilde{y} = y$), we need to check whether $\tilde{y}$ is consistent with the true conditional distribution $\eta$, namely, the Bayes optimal classifier prediction $h^*(x)$. To this end, we assume $f$ is a close enough approximate of $\tilde{y}$ as it is trained on the noisy labels. This way, testing whether $f$ agrees with $\tilde{y}$ is close to testing whether $\tilde{y}$ agrees with $\tilde{y}$, except that the threshold $\Delta$ needs to be carefully chosen. Another issue we need to consider is that the $\Delta$ is unknown. Our main theorem will bound the chance of failed correction by how close $f$ approximates $\eta$ and how close we can set $\Delta$ to the perfect one.

**Remark 1.** Our likelihood testing is the uniformly most powerful one for the intended hypothesis, based on the Neyman-Pearson Lemma. In other words, this test has the strongest statistical power in rejecting a false null hypothesis.

### 2.2.1 Formal Statement of the Algorithm and the Theorem

We start by assuming $f$ depends linearly on $\eta$, i.e., $f(x) = a\eta(x) + b$, in which $a, b > 0$ are known constants. But this constraint will be relaxed later (Remark 2). Consider three different conditions based on the noise patterns: $\tau_{10} < \tau_{01}, \tau_{10} = \tau_{01}$, or $\tau_{10} > \tau_{01}$. Let $u$ be $-1, 0$ or $+1$ corresponding...
to which one of these three conditions holds. Based on different \( u, \tilde{y} \) and \( f(x) \), we choose different \( \Delta \) as in Table \ref{tab:delta}. The label correction algorithm is given in Procedure \ref{proc:lrc}. It checks the likelihood ratio with regard to the chosen \( \Delta \). If LR is no greater than \( \Delta \), we flip the label. Otherwise, \( \tilde{y}_{new} \) is the same as \( \tilde{y} \). In practice, \( \Delta \) is unknown and needs to be decided empirically.

**Procedure 1 LRT-Correction**

*Input:* \( f, (x, y), u, a, b \).

*Output:* \( \tilde{y}_{new} \)

1: if \( LR(f; x, \tilde{y}) \leq \Delta(f, \tilde{y}, u, a, b) \),
   where \( \Delta \) is as in Table \ref{tab:delta} then
2: \( \tilde{y}_{new} = 1 - \tilde{y} \)
3: else
4: \( \tilde{y}_{new} = \tilde{y} \)
5: end if

<table>
<thead>
<tr>
<th>( y )</th>
<th>( f &lt; b + a/2 )</th>
<th>( f &gt; b + a/2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tilde{y} = 1 )</td>
<td>( \frac{\tilde{a} + \tilde{b}}{2a + 2b} )</td>
<td>( 0 )</td>
</tr>
<tr>
<td>( \tilde{y} = 0 )</td>
<td>( \frac{2 - a - 2b}{a + 2b} )</td>
<td>( \frac{2 - a - 2b}{a + 2b} )</td>
</tr>
</tbody>
</table>

Our main theorem states that suppose the classifier prediction, \( f \), is a close approximation of the noisy conditional probability, \( \tilde{\eta} \). And suppose we can find a good enough \( \Delta^* \) that is close enough to the ideal \( \Delta \). Then there is a very good chance that our algorithm corrects most labels to the correct ones, i.e., the same as the Bayes optimal classifier prediction. Please note that here “proper correction” means that the new label, \( \tilde{y}_{new} \), is the same as the Bayes optimal classifier prediction, \( h^*(x) \), instead of \( y \). This is well justified as it means that the correction will give us a classifier as good as the Bayes optimal one.

**Theorem 1.** Assume \( \eta \) satisfies the Tsybakov condition with constants \( C > 0 \) and \( \lambda > 0 \). Recall \( h^* \) denote the Bayes optimal classifier. The noisy conditional probability \( \tilde{\eta}(x) = (1 - \tau_0 - \tau_{10})\eta(x) + \tau_{10} \). Assume \( f(x) = ax(x) + b \) with a and b unknown, such that \( f(x) \in [\tilde{\eta}(x) - \epsilon, \tilde{\eta}(x) + \epsilon] \) for some \( \epsilon > 0 \). Let \( \Delta(\tilde{y}, f(x), u, a, b) \) be as in Table \ref{tab:delta}. Let \( \Delta^* > 0 \) be a constant such that \( \Delta^* \in [\Delta - \epsilon, \Delta + \epsilon] \). If \( \tilde{y}_{new} \) denotes the output of the LRT-Correction with \( \tilde{y}, x, f, \) and the give \( \Delta^* \), then

\[
Pr_{(x,y)\sim D}[\tilde{y}_{new} \neq h^*] \leq 8C \left( \frac{\tau_{10} - \tau_0}{2(1 - \tau_{10} - \tau_0)} + O(\epsilon) \right)^\lambda.
\]

If \( \tau_0 = \tau_{10} \), then \( Pr_{(x,y)\sim D}[\tilde{y}_{new} \neq h^*] = 8C (O(\epsilon))^\lambda \).

**Remark 2.** The condition that \( f \) is linear to \( \eta \) is not necessary for the theorem to hold. We only require \( f \) to be pointwise close to \( \tilde{\eta} \).

**Intuition of the proof.** We will prove two lemmas. The first lemma shows that when strictly assuming \( f \) is linear to \( \eta \), with known coefficients \( a, b \), and we set \( \Delta \) according to Table \ref{tab:delta} then the correction algorithm can be correct everywhere. The second lemma proves a more relaxed version. If \( f \) is exactly \( \tilde{\eta} \) and if we only know the difference between the transition probabilities, \( \tau_{01} - \tau_{10} \), then we can bound the chance of mistakes of the correction algorithm. Finally, based on these two lemmas, careful case analysis, and the Tsybakov condition of the true conditional probability, \( \eta \), we can prove the theorem. The complete proof can be found in Appendix \ref{app:proof}.

So far, all the description and theoretical results are based on a binary classification setting. However, the results can be generalized to a multiclass setting without any technical difficulties. Informally, we state the following corollary (proof omitted).

**Corollary 1.** LRT-Correction can be generalized to multiclass classification tasks, by flipping \( \tilde{y} \) to the best prediction of \( f \) when the null hypothesis is rejected. Theorem \ref{thm:main} can be generalized to multiclass classification tasks, by considering all pairs of class values.

\[\text{Here } u \text{ is irrelevant for choosing } \Delta \text{ value here. But we keep it as a parameter for } \Delta \text{ as it will be necessary in the proof (Appendix \ref{app:proof}).}\]
2.3 Training Deep Neural Networks with LRT-Label-Correction

Our training algorithm continuously trains a deep neural network while correcting the noisy labels. Procedure 2 is the pseudocode of the training method, called AdaCorr. It trains a neural network model iteratively. Each iteration includes both label correction and model training steps. In label correction step, the prediction of the current neural network, \( f \), is used to run LRT test on all training data, and to correct their labels according to the test result. Since \( f \) is used to approximate the conditional probability \( \tilde{y} \), we use the softmax layer output of the neural network as \( f \). After the labels of all training data are updated, we use them to train the neural network incrementally. We continue this iterative procedure until the whole training converges.

We also have a burn-in stage in which we train the network using the original noisy labels for \( m \) epochs. During the burn-in stage, we use the original cross-entropy loss, \( L_{CE} \). Afterwards, we add an additional retroactive loss which will be explained below.

Training with retroactive loss. After the burn-in stage, we want to avoid overfitting of the neural network, so that its output better approximates \( \tilde{y} \). To achieve this goal, we introduce a retroactive loss term \( L_{retro}(f(x), \tilde{y}) \). The idea is to enforce the consistency between \( f \) and the prediction of the model at a previous epoch, \( f' \). It has been observed that a neural network at earlier training stage tends to learn the true pattern rather than to overfit the noise \( [\text{Arpit et al.} 2017] \). Formally, the loss can be written as \( \sum_{c=1}^{N_c} f'_c(x) \log f_c(x) \), in which \( N_c \) is the number of possible label classes. The training loss is the sum of the retroactive loss and the cross-entropy loss

\[
L(f(x), \tilde{y}) = L_{retro}(f(x), \tilde{y}) + L_{CE}(f(x), \tilde{y}) = \sum_{c=1}^{N_c} f'_c(x) \log f_c(x) + \sum_{c=1}^{N_c} \tilde{y}_c \log f_c(x).
\]

In practice, we set \( f' \) to be the prediction of the model at the \( m \)-th epoch. In other words, once the burn-in stage is finished, the training switches from \( L_{CE} \) to \( L_{CE} + L_{retro} \). And the model at the end of the burn-in stage is used for the retroactive loss. We also set the label correction to start slightly after the burn-in stage, say \( m + 10 \). The key hyperparameter is the starting epoch \( m \). Another hyperparameter is \( \Delta \). We select both \( m \) and \( \Delta \) empirically. Ablation study in Section 4 shows that our method is robust to these hyperparameters.

3 Experiments

In this section we empirically evaluate our proposed method with several datasets, where noisy labels are injected according to specified noise transition matrices.

Datasets. We use the following datasets: MNIST \( [\text{LeCun \\& Cortes} 2010] \), CIFAR10 \( [\text{Krizhevsky et al.} 2009] \), CIFAR100 \( [\text{Krizhevsky et al.} 2010] \) and ModelNet40 \( [\text{Wu \\& Xiao} 2015] \). MNIST consists of \( 28 \times 28 \) grayscale images with 10 categories. It contains 60,000 images, and we use 45,000 for training, 5,000 for validation and 10,000 for testing. CIFAR10 and CIFAR100 share the same 60,000 \( 32 \times 32 \times 3 \) image data, with CIFAR10 having 10 categories while CIFAR100 having 100 categories. Similar to MNIST, we split 90% and 10% data from the official training set for the training and validation respectively, and use the official test set for testing. ModelNet40 contains 12,311 CAD models from 40 categories, where 8,859 are used for training, 984 for validation and the remaining 2,468 for testing. We follow the protocol of \( [\text{Qi et al.} 2017] \) to convert the CAD models into point clouds by uniformly sampling 1,024 points from the triangular mesh and normalizing them within a
unit ball. In all experiments, we use early stopping on validation set to tune hyperparameters and to report the performance on test set.

**Noise patterns.** Following (Reed et al. 2014, Patrini et al. 2017), we corrupt our labels artificially using a noise transition matrix $T$, where $T_{ij} = \tau_{ij} = P(y = j | y = i)$ is the probability that category $i$ is flipped to category $j$. In our work we focus on two types of $T$: (1) uniform, where the true label $i$ is flipped to other classes with equal probabilities, i.e., $T_{ij} = p/(N_c - 1)$ for $i \neq j$ and $T_{ii} = 1 - p$, where $p$ is the noise level and $N_c$ is the class number; (2) pair flipping, where the true label $i$ is flipped to $j$ with $T_{ij} = p$ for $i \neq j$ and $T_{ii} = 1 - p$. Examples of these two types of transition matrices are as follows:

$$T_1 = \begin{pmatrix} 0.7 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.7 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.7 \end{pmatrix} \quad T_2 = \begin{pmatrix} 0.7 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.7 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.7 & 0.3 \\ 0.3 & 0.0 & 0.0 & 0.7 \end{pmatrix}$$

in which $T_1$ is for uniform noise pattern with noise level 0.3 and $T_2$ is for pair flipping with noise level 0.3.

**Baselines.** We compare the proposed method with the following methods: (1) Standard, which trains the network in a standard manner, without any label resistance technique; (2) Forward correction (Patrini et al. 2017), which explicitly estimates the noise transition matrix to correct the training loss; (3) Decoupling (Malach & Shalev-Shwartz 2017), which trains two networks simultaneously and updates the parameters on selected data whose labels are possibly clean; (4) Coteaching (Han et al. 2018), which also trains two networks but exchanges their error information for network update; (5) MentorNet (Jiang et al. 2018), which learns a curriculum to filter out noisy data; (6) Forgetting (Arpit et al. 2017b), which uses dropout to help deep models resist label noise.

**Experimental Setup.** For the classification of MNIST, CIFAR10 and CIFAR100, we use preactive resnet34 (He et al. 2016) as the backbone for all the methods. On ModelNet40, we use PointNet (Qi et al. 2017). We train the models for 180 epochs to ensure that all the methods have converged. We utilize RAdam (Liu et al. 2019) for the network optimization, and adopt batch size 128 for all the datasets. The experimental results are listed in Table 2. As is shown, our method outperforms the competing methods across the datasets under different noise settings.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Method</th>
<th>Noise Level of Uniform Flipping</th>
<th>Noise Level of Pair Flipping</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>MNIST</td>
<td>Standard</td>
<td>99.0 ± 0.2</td>
<td>98.7 ± 0.4</td>
</tr>
<tr>
<td></td>
<td>Forget</td>
<td>99.0 ± 0.1</td>
<td>98.8 ± 0.1</td>
</tr>
<tr>
<td></td>
<td>Forward</td>
<td>99.1 ± 0.1</td>
<td>98.7 ± 0.2</td>
</tr>
<tr>
<td></td>
<td>Decouple</td>
<td>99.3 ± 0.1</td>
<td>99.0 ± 0.1</td>
</tr>
<tr>
<td></td>
<td>MentorNet</td>
<td>99.2 ± 0.2</td>
<td>98.7 ± 0.1</td>
</tr>
<tr>
<td></td>
<td>Coteach</td>
<td>99.1 ± 0.2</td>
<td>98.7 ± 0.3</td>
</tr>
<tr>
<td></td>
<td>AdaCorr</td>
<td>99.5 ± 0.0</td>
<td>99.4 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>AdaCorr</td>
<td>99.5 ± 0.0</td>
<td>99.4 ± 0.0</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>Standard</td>
<td>87.5 ± 0.2</td>
<td>83.1 ± 0.4</td>
</tr>
<tr>
<td></td>
<td>Forget</td>
<td>87.1 ± 0.2</td>
<td>83.4 ± 0.2</td>
</tr>
<tr>
<td></td>
<td>Forward</td>
<td>87.4 ± 0.8</td>
<td>83.1 ± 0.8</td>
</tr>
<tr>
<td></td>
<td>Decouple</td>
<td>87.6 ± 0.4</td>
<td>84.2 ± 0.5</td>
</tr>
<tr>
<td></td>
<td>MentorNet</td>
<td>90.3 ± 0.3</td>
<td>83.2 ± 0.5</td>
</tr>
<tr>
<td></td>
<td>Coteach</td>
<td>90.1 ± 0.4</td>
<td>87.3 ± 0.5</td>
</tr>
<tr>
<td></td>
<td>AdaCorr</td>
<td>91.0 ± 0.5</td>
<td>88.7 ± 0.5</td>
</tr>
<tr>
<td>CIFAR100</td>
<td>Standard</td>
<td>58.9 ± 0.8</td>
<td>52.1 ± 1.0</td>
</tr>
<tr>
<td></td>
<td>Forget</td>
<td>59.3 ± 0.8</td>
<td>53.0 ± 0.2</td>
</tr>
<tr>
<td></td>
<td>Forward</td>
<td>58.4 ± 0.5</td>
<td>52.2 ± 0.3</td>
</tr>
<tr>
<td></td>
<td>Decouple</td>
<td>59.0 ± 0.7</td>
<td>52.2 ± 0.7</td>
</tr>
<tr>
<td></td>
<td>MentorNet</td>
<td>63.6 ± 0.5</td>
<td>51.4 ± 1.4</td>
</tr>
<tr>
<td></td>
<td>Coteach</td>
<td>66.1 ± 0.5</td>
<td>60.0 ± 0.6</td>
</tr>
<tr>
<td></td>
<td>AdaCorr</td>
<td>67.8 ± 0.1</td>
<td>60.2 ± 0.8</td>
</tr>
<tr>
<td>ModelNet40</td>
<td>Standard</td>
<td>79.1 ± 2.6</td>
<td>75.3 ± 3.3</td>
</tr>
<tr>
<td></td>
<td>Forget</td>
<td>80.1 ± 1.8</td>
<td>73.9 ± 0.6</td>
</tr>
<tr>
<td></td>
<td>Forward</td>
<td>52.3 ± 5.1</td>
<td>49.4 ± 6.8</td>
</tr>
<tr>
<td></td>
<td>Decouple</td>
<td>82.5 ± 2.2</td>
<td>80.7 ± 0.7</td>
</tr>
<tr>
<td></td>
<td>MentorNet</td>
<td>57.8 ± 0.5</td>
<td>52.7 ± 3.1</td>
</tr>
<tr>
<td></td>
<td>Coteach</td>
<td>85.6 ± 0.9</td>
<td>84.2 ± 0.8</td>
</tr>
<tr>
<td></td>
<td>AdaCorr</td>
<td>86.9 ± 0.3</td>
<td>85.1 ± 0.6</td>
</tr>
</tbody>
</table>

Under review as a conference paper at ICLR 2020
4 Ablation Study and Discussion

We conduct ablation study to see the significance of our contributions. We compare our method (LRT + L_{ce} + L_{retro}) with two baselines: our method without the retroactive loss (LRT + L_{ce}) and using cross-entropy loss only without LRT-Correction (L_{ce} Only). We report the test accuracy on CIFAR10 in Table 3. We observe that adding LRT Label Correction to L_{ce} alone helps improve the performance significantly. The numbers inside the parenthesis are the percentage of correct labels after the training and label correction. We observe that the LRT Correction corrected a large portion of noise labels. Also we observe adding the retroactive loss improves the method further in both test accuracy and label correction rates.

<table>
<thead>
<tr>
<th>Method</th>
<th>uniform 0.2</th>
<th>uniform 0.4</th>
<th>uniform 0.6</th>
<th>uniform 0.8</th>
<th>pair 0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>L_{ce} Only</td>
<td>87.5(80.0)</td>
<td>83.1(60.0)</td>
<td>76.4(40.0)</td>
<td>47.6(20.0)</td>
<td>84.5(60.0)</td>
</tr>
<tr>
<td>LRT + L_{ce}</td>
<td>91.1(95.7)</td>
<td>87.9(91.2)</td>
<td>80.7(81.7)</td>
<td>47.3(47.1)</td>
<td>87.5(91.9)</td>
</tr>
<tr>
<td>LRT + L_{ce} + L_{retro}</td>
<td>91.0(95.7)</td>
<td>88.7(90.5)</td>
<td>81.2(82.5)</td>
<td>49.2(49.0)</td>
<td>89.2(91.3)</td>
</tr>
</tbody>
</table>

We evaluate how different hyperparameters affect the performance of our method. We compare our method with different \( m \), the length of the burn-in stage. We start introducing the retroactive loss after \( m \) epochs, and start label corrections after \( m + 10 \) epochs. The final testing accuracies are shown in Table 4. We observe the performance of our method is rather robust w.r.t. different \( m \)'s. We choose \( m = 20 \) in this data set (CIFAR10) and similarly in other datasets.

<table>
<thead>
<tr>
<th>Noisy Type</th>
<th>Epoch 15</th>
<th>Epoch 20</th>
<th>Epoch 25</th>
<th>Epoch 30</th>
<th>Epoch 35</th>
<th>Epoch 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniform 0.4</td>
<td>87.6(90.1)</td>
<td>88.7(90.5)</td>
<td>87.4(90.7)</td>
<td>86.7(90.6)</td>
<td>84.8(88.7)</td>
<td>84.1(87.2)</td>
</tr>
<tr>
<td>uniform 0.6</td>
<td>79.4(81.0)</td>
<td>81.2(82.5)</td>
<td>80.9(81.9)</td>
<td>79.3(81.9)</td>
<td>79.1(81.8)</td>
<td>78.1(81.7)</td>
</tr>
<tr>
<td>pair 0.4</td>
<td>75.8(80.0)</td>
<td>89.2(80.1)</td>
<td>90.8(87.0)</td>
<td>89.2(89.3)</td>
<td>88.2(90.1)</td>
<td>86.7(90.1)</td>
</tr>
</tbody>
</table>

We also evaluate the performance on different \( \Delta \)'s. \( \Delta \) is the unknown value for our likelihood ratio test. It controls how aggressive we would like to correct the labels. From Table 5, we observe a bigger \( \Delta \) tends to give better results (as it is less aggressive in correcting labels). We observe 1/1.2 is the best one for CIFAR10. Similar values of optimal \( \Delta \) are found in other data sets.

<table>
<thead>
<tr>
<th>Noisy Type</th>
<th>1/1.0</th>
<th>1/1.2</th>
<th>1/1.5</th>
<th>1/2.0</th>
<th>1/2.5</th>
<th>1/3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniform 0.4</td>
<td>88.3(91.7)</td>
<td>88.7(90.5)</td>
<td>83.0(89.3)</td>
<td>77.1(86.7)</td>
<td>75.2(85.1)</td>
<td>75.5(84.0)</td>
</tr>
<tr>
<td>uniform 0.6</td>
<td>79.9(81.0)</td>
<td>81.2(82.5)</td>
<td>80.9(81.9)</td>
<td>79.3(81.9)</td>
<td>79.1(81.8)</td>
<td>78.1(81.7)</td>
</tr>
<tr>
<td>pair 0.4</td>
<td>88.2(92.2)</td>
<td>89.2(80.1)</td>
<td>84.4(89.8)</td>
<td>77.0(86.8)</td>
<td>76.4(85.1)</td>
<td>77.0(84.1)</td>
</tr>
</tbody>
</table>

In general, we observe our hyperparameters are rather consistent across different datasets. This reveals a better generalization power of our method over other datasets and noise patterns. We believe this is due to the principled approach we take in label cleaning.

5 Conclusion

We propose a label correction algorithm to combat label noise. We perform a likelihood ratio test for each input label such that if it is rejected, this label is flipped to the class that has the highest likelihood. Theoretically, we prove that our method corrects noisy labels with high probability. Experiments on various datasets show that our method outperforms state-of-the-arts and is robust to hyperparameters.
REFERENCES


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


Eran Malach and Shai Shalev-Shwartz. Decoupling “when to update” from “how to update”. In NeurIPS, 2017.


A P R O O F O F T H E O R E M 1

Recall \( u \in \{-1, 0, 1\} \) is a number indicating whether \( \tau_{10} < \tau_{01}, \tau_{10} = \tau_{01}, \) or \( \tau_{10} > \tau_{01}, \) respectively. We restate the theorem as follows for completeness.

**Theorem 1.** Assume \( \eta \) satisfies the Tsybakov condition with constants \( C > 0 \) and \( \lambda > 0. \) Let \( h^* \) denote the Bayes classifier. Let \( \tilde{\eta}(x) = (1 - \tau_{01} - \tau_{10})\eta(x) + \tau_{01} \) denote the corrupted Bayesian classifier. Let \( f(x) = a\eta(x) + b \) with \( a \) and \( b \) unknown, such that \( f(x) \in [\tilde{\eta}(x) - \epsilon, \tilde{\eta}(x) + \epsilon] \) for some \( \epsilon > 0. \) Let \( u \in \{-1, 0, +1\} \) be known. Let \( \Delta(\tilde{y}, f(x), u, a, b) \) be as in Table 1. Let \( \Delta' > 0 \) be a constant such that \( \Delta' \in [\Delta - \epsilon, \Delta + \epsilon]. \) If \( \tilde{y}_{\text{new}} \) denotes the output of the LRT-algorithm with input \((\tilde{y}, x, f, \Delta')\), then

\[
\Pr_{(x,y) \sim D}(\tilde{y}_{\text{new}} \neq h^*) \leq 8C \left( \frac{\tau_{10} - \tau_{01}}{2(1 - \tau_{10} - \tau_{01})} + O(\epsilon) \right)^{\lambda}.
\]
If \( \tau_{01} = \tau_{10} \), then \( \Pr_{(x,y) \sim D}(\tilde{y}_{new} \neq h^*) = 8C (O(\epsilon))^{\lambda} \).

To prove this theorem, we first prove two lemmas. In the following we will state and prove the lemmas, and then continue to finish the proof of Theorem 1.

**Lemma 2.** Assume \( \eta \) satisfies the Tsybakov condition with constants \( C > 0 \) and \( \lambda > 0 \). Let \( h^* \) denote the Bayes classifier. If \( f \) depends linearly on \( \eta \), i.e., \( f(x) = a\eta(x) + b \), \( a, b > 0 \), and \( a, b \) and \( u \) are known, and \( \Delta \) is chosen as in Table 1 then

- If LRT-Correction\((f,(x,y),u,a,b)\) flips the label of \( y \), then \( \tilde{y}_{new} = h^*(x) \), and
- \( \Pr_{(x,y) \sim D}(\tilde{y}_{new},y) \neq h^*(x) \) = 0.

**Lemma 3.** Assume \( \eta \) satisfies the Tsybakov condition with constants \( C > 0 \) and \( \lambda > 0 \). Let \( h^* \) denote the Bayes classifier. Let \( \bar{\eta}(x) = (1 - \tau_{01} - \tau_{10})\eta(x) + \tau_{01} \) denote the corrupted Bayesian classifier. If \( \Delta \) is chosen as in Table 6, then

- If LRT-Correction\((f,(x,y),u,a,b)\) flips the label of \( y \), then \( \tilde{y}_{new} = h^*(x) \),
- \( \Pr_{(x,y) \sim D}(\tilde{y}_{new},y) \neq h^*(x) \) \leq C \left( \frac{\tau_{01} - \tau_{10}}{2(1 - \tau_{01} - \tau_{10})} \right)^{\lambda} \), and
- \( \Pr_{\tau_{01} = \tau_{10}}(f(x,y) \sim D(\tilde{y}_{new},y) \neq h^*(x)) = 0. \)

**Table 6: Values of \( \Delta \) given \( \tilde{y} \)**

<table>
<thead>
<tr>
<th>( y )</th>
<th>( \tilde{y} &lt; 1/2 )</th>
<th>( \tilde{y} &gt; 1/2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y = 1 )</td>
<td>( \frac{1+\tau_{01} - \tau_{10}}{1+\tau_{10} - \tau_{01}} )</td>
<td>0</td>
</tr>
<tr>
<td>( y = 0 )</td>
<td>0</td>
<td>( \frac{1+\tau_{01} - \tau_{10}}{1+\tau_{01} - \tau_{10}} )</td>
</tr>
</tbody>
</table>

**Proof of Lemma 2**

We begin with some observations:

**Observation 1:** If \( \tilde{y} = 1(f > b + a/2) \), the algorithm never flips. This is because \( \Delta = 0 \) in all such cases, and since the likelihood ratios \( f/(1 - f) \) or \( (1 - f)/f \) are always positive, \( \tilde{y} \) is not flipped.

**Observation 2:** \( f(x) < b + a/2 \) if, and only if, \( \eta(x) < 1/2 \). This follows by a straightforward calculation using \( f(x) = a\eta(x) + b \).

In the remainder of this proof, we assume that \( u = 1 \), i.e., \( \tau_{10} > \tau_{01} \). The proof when \( u = -1 \) follows by symmetry.

**Observation 3:** Assume \( u = 1 \). If \( \tilde{y}(x) = 1 \) and \( f(x) < b + a/2 \) (so a non-trivial test (unlike ones in Observation 1) is performed), then \( f(x) / (1 - f(x)) < \Delta(\tilde{y} = 1, f(x) < b + a/2) \) if, and only if, \( \eta(x) = 1/2 \). Similarly if \( \tilde{y}(x) = 0 \) and \( f(x) > b + a/2 \) then \( (1 - f(x))/f(x) < \Delta(\tilde{y} = 0, f(x) > b + a/2) \) if, and only if, \( \eta(x) > 1/2 \).

**Proof of Observation 3:** First notice from the table that \( \Delta(\tilde{y} = 1, f(x) < 1/2) \) is the reciprocal of \( \Delta(\tilde{y} = 0, f(x) > 1/2) \). Since the likelihood test ratio is also the reciprocal in the two situations, proving one statement suffices. We prove the first statement. For convenience we denote \( \Delta(\tilde{y} = 1, f(x) < 1/2) \) by \( \Delta \).

\[
\begin{align*}
\frac{f(x)}{1 - f(x)} &< \Delta \\
\iff f(x) &< \Delta/(1 + \Delta) \\
\iff f(x) &< \frac{a + 2b}{2 - a - 2b} \\
\iff f(x) &< \frac{a + 2b}{2 - a - 2b} \\
\iff \eta(x) &< \frac{a + 2b}{2} - b \\
\iff \eta(x) &< 1/2.
\end{align*}
\]
We can now prove the first assertion of Lemma 4 for the case $u = 1$. That is, if $LRT-Correction(f, (\tilde{y}, \tilde{y}), u, a, b)$ flips the label of $\tilde{y}$, then $\tilde{y}_{new} = h^*(x)$. Let us first consider the case $\tilde{y}(x) = 1$. If $f(x) > b + a/2$, then it implies that $\eta(x) > 1/2$ (Observation 2). The corresponding $\Delta$ in the table is 0, meaning that no test is performed, and $\tilde{y}_{new}(x, \tilde{y}) = \tilde{y}(x) = h^*(x)$. On the other hand, if $f(x) < b + a/2$, we perform a likelihood ratio test on $f/(1 - f)$, which is less than the corresponding value of $\Delta$ if $\eta(x) < 1/2$ (Observation 3), so again $\tilde{y}_{new}(x, \tilde{y}) = \tilde{y}(x) = h^*(x)$. The case $\tilde{y}(x) = 0$ can be analyzed analogously.

Thus the only points $x$ where we may not match $h^*(x)$ are ones where the flipping algorithms do not flip, and the result mismatches with $h$. By the above observations, the following is where this happens.

**Observation 4:** Assume $u = 1$. Then
\[
\{x : \tilde{y}_{new}(x, \tilde{y}) \neq h^*(x)\} = \{x : \tilde{y}(x) = 0, f(x) < b + a/2, \eta(x) > 1/2\}.
\]

However, according to Observation 2, $f(x) < b + a/2$ implies $\eta(x) < 1/2$. Thus this case cannot happen. Hence $Pr((x, y) \sim D(\tilde{y}_{new}(x, \tilde{y}) \neq h^*(x))) = 0$, and Lemma 2 is proved.

**Proof of Lemma 3**

This proof is an extension of the proof of Lemma 2. All the steps are analogous, except in Observation 4, we get the following

**Observation 5:** Assume $u = 1$. Then
\[
\{x : \tilde{y}_{new}(x, \tilde{y}) \neq h^*(x)\} = \{x : \tilde{y}(x) = 0, \tilde{y}(x) < 1/2, \eta(x) > 1/2\}.
\]

This measure is non-zero. However, it is bounded above by the measure of the set $\{\tilde{y}(x) < 1/2, \eta(x) > 1/2\}$. Since $\tilde{y}(x) = (1 - \tau_{01} - \tau_{10})\eta(x) + \tau_{01}$, this is the same as the measure of the set $\{x : 1/2 < \eta(x) < 1/2 - \tau_{01} - \tau_{10}\}$. By the Tsybakov condition, this measure is at most $\left(\frac{\tau_{10} - \tau_{01}}{2(1 - \tau_{10} - \tau_{01})}\right)^{\lambda}$. This was for the case $u = 1$, and when $u = -1$ we can check by analogous calculation that the numerator is reversed. Hence we have proved
\[
Pr((x, y) \sim D(\tilde{y}_{new}(x, \tilde{y}) \neq h^*(x))) \leq C \left(\frac{\tau_{01} - \tau_{10}}{2(1 - \tau_{10} - \tau_{01})}\right)^{\lambda}.
\]

The last assertion of Lemma 3 follows by just substituting $\tau_{01} = \tau_{10}$ in the above formula. Hence Lemma 3 is proved.

**Proof of Theorem 1**

We give here a sketch of the proof. Let $A$ denote the event $(f < \lambda < \Delta)'$, and $B$ denote the event $(\frac{1 - \tau}{f} < \Delta')'$. In the following we describe the cases when our algorithm results (flips or does not flip) in a label inconsistent with $h^*$.

- $(\tilde{y} = 1, f < 1/2, \eta > 1/2, LRT flips).$ According to the LRT scheme, this happens with probability at most $Pr(A \cap (\eta > 1/2))$.
- $(\tilde{y} = 1, f < 1/2, \eta < 1/2, LRT does not flip).$ According to the LRT scheme, this happens with probability at most $Pr(A^c \cap (\eta < 1/2))$. Here $A^c$ denotes the complement of the event $A$.
- $(\tilde{y} = 1, f > 1/2, \eta > 1/2, LRT flips).$ According to the LRT scheme, this happens with probability at most $Pr(A \cap (\eta > 1/2))$.
- $(\tilde{y} = 1, f > 1/2, \eta < 1/2, LRT does not flip).$ According to the LRT scheme, this happens with probability at most $Pr(A^c \cap (\eta < 1/2))$.
- $(\tilde{y} = 0, f < 1/2, \eta > 1/2, LRT does not flip).$ According to the LRT scheme, this happens with probability at most $Pr(B \cap (\eta > 1/2))$.
- $(\tilde{y} = 0, f > 1/2, \eta < 1/2, LRT does not flip).$ According to the LRT scheme, this happens with probability at most $Pr(B^c \cap (\eta < 1/2))$.
- $(\tilde{y} = 0, f < 1/2, \eta > 1/2, LRT does not flip).$ According to the LRT scheme, this happens with probability at most $Pr(B \cap (\eta > 1/2))$. 

12
In all of these cases, one observes that the probability of these 8 events is either of the form $\Pr\left(\frac{1}{2} < \eta(x) < \frac{1/2 - \tau_{01}}{1 - \tau_{01} - \tau_{10}} + O(\epsilon)\right)$ or $\Pr\left(\frac{1/2 - \tau_{01}}{1 - \tau_{01} - \tau_{10}} - O(\epsilon) < \eta(x) < \frac{1}{2}\right)$. Assuming Tsybakov condition, one can bound each of these probabilities (and hence the sum by) $8\left(\frac{\tau_{10} - \tau_{01}}{2(1 - \tau_{10} - \tau_{01})} + O(\epsilon)\right)^\lambda$. Thus we have proved

$$\Pr_{(x,y)\sim D}(\tilde{y}_{new} \neq h^*) \leq 8C\left(\frac{\tau_{10} - \tau_{01}}{2(1 - \tau_{10} - \tau_{01})} + O(\epsilon)\right)^\lambda,$$

which finishes the proof of Theorem 1.