Assignment 2

Group members: Arjun Sathasivam (470367366 - asat9100), Chau Ngo (470157446 - bngo5284), Dhanu Vardhan (480563862 - djha5117)

Abstract

Image processing plays a crucial role in many fields such as robotics, autonomous vehicles and healthcare, yet it remains a massive challenge in machine learning study. One reason is due to the presence of label noise, which may cause serious negative consequences. In this paper, we implement three algorithms, namely Importance Reweighting, Unbiased Estimator and Rank Pruning, to address label noise problems. Our experiments also test the robustness of these methods on the given corrupted datasets. Importance Reweighting modifies the loss function by reweighting the importance given to each data label. Unbiased Estimator method also modifies the loss function by constructing an α reweighted loss function such that this function performs well for traditional learning on noisy label data. Rank Pruning algorithm removes the non-confident examples with respect to their labels from the dataset and trains only examples that are likely to be correct. Furthermore, we also implement a method for estimating the flip rates as part of Rank Pruning methodology.

1 Introduction

Classification is one of the most widely studied problems in machine learning. The digitisation of processes is leading to increasing application of classification in real world scenarios. Traditional machine learning methodologies solve the problem of classification by learning from training data and evaluating on real time test data. In the supervised setting the training data consists of features and their associated labels. Classification exploits this relationship between the features and their labels to learn a classifier. However, in real world problems these datasets can be contaminated with noise. The noise can be of two types: in the features and in the labels. The noise in features can corrupt the observations whereas the noise in the labels alters the class to which the observation is assigned. [1] and [2] show that label noise can be more harmful than feature noise. [3] shows that the traditional learning rate decreases monotonically with label noise. Therefore dealing with label noise is of great practical importance. Example of noisy labelled datasets include medical, human labeled and sensor datasets. Correcting these labels manually is costly and time consuming. Furthermore, it is not an easy task to distinguish data manually especially if it lacks visually discriminative information.

Label noise can be broadly classified into three categories. First, Random Classification Noise (RCN) where the noise is assumed to be a constant and each sample is assumed to be flipped randomly with a probability $\rho \in [0, 0.5)$. Second, label noise can be assumed to be class dependent where the probability of every sample being mislabeled is only dependent on its class. The probability of mislabeling a positive class instance is given by $P(S = 0|Y = 1)$ and the probability of negative labels by $P(S = 1|Y = 0)$. They are denoted as ρ_1 and ρ_0, respectively. Third, the label noise can be a function of both the class and the observation point.

In this paper we assume the label noise to be class dependent. The probability of mislabeling is denoted as flip rates. We present three methods to handle the noisy labels problem, which are importance reweighting, unbiased estimator and rank pruning.

Importance reweighting, as proposed by Liu et al. [4], assumes the class labels are flipped independently with constant probabilities ρ_1 and ρ_0. By applying a classification algorithm to the unmodified noisy training set, it estimates ρ_1 and ρ_0, and calculates a factor β for each data point. The factor β ultimately acts to reweight each data point when the actual classifier is trained later.

The unbiased estimator method has been proposed by Nagarajan et al. [5]. It formulates a label dependent cost function such that the expected loss over the noisy data using the weighted surrogate loss function is of the same sign as the expected loss obtained from minimising the expected loss by a Bayes classifier using 0-1 loss on clean data.

Rank pruning method, proposed by Northcutt et al. [6], consists of 2 parts: estimation of the flip rates and removal of likely mislabeled samples from the corrupted dataset. Instead of removing the samples simply by predicted probability, Rank Pruning estimates the number of mislabeled samples in each class and prunes such number of samples by ranking order. Thus, one advantage of Rank Pruning is the ability to mitigate the sensitivity of the probability estimation produced by the chosen classifier [7]. Furthermore, it also proves the robustness and consistency to both mislabeled and added noise.

The main purpose of this paper is to show, compare and analyse the robustness of the above three methods in handling label noise. In section 2 we give a review on some related approaches to tackle this problem. Section 3 presents in-depth knowledge about our chosen label noise methods, including how to estimate the values of flip rates. In section 4, we do some experiments on the two given datasets using the given flip rate values and evaluate the robustness of these methods by comparing their average accuracies on test data. The last section is to summarize our findings and future work suggestions.

2 Related work

Frénay et al. [8] provided a comprehensive survey on the sources, consequences of label noise and some techniques to address this problem. They classified those approaches into 3 categories: label noise-robust models such as bagging; data cleansing methods such as outlier detection and label noise-tolerant learning algorithms. Label noise-robust models rely on the ability to prevent overfitting. Data cleansing methods completely remove the likely mislabeled samples data that are above a given threshold from the training set. This approach may also lead to overcleansing that reduces the performance of the classifiers, due to the lack of training data. However, some research work showed that the act of overcleasing played less harmful than keeping those mislabeled instances [11]. Moreover, some papers also concluded that totally removing the mislabled samples yielded better performance than re-labeling them [9] [10]. Label noise-tolerant learning algorithms consist of probabilistic models, neural networks and SVM, etc, which allow us to gain knowledge from the analysis of label noise consequences, with the cost of model complexity as trade-off.

In the case of Random Classification Noise, Angluin et al. [12] proved that Random Classification Noise is PAC-learnable. Ghosh et al. [13] proved a sufficient condition on a loss function to make it robust to label noise. These functions include of 0 - 1 loss, sigmoid loss, ramp loss and probit loss. Furthermore, under certain conditions, the 3 later functions can be tolerant to non-uniform label noise.

With class-dependent label noise, several attempts to solve this problem have been done. Class-dependent label noise problems can be divided into 2 sequential processes: estimating the flip rates and using the estimated flip rate values for prediction. Liu et al. [4], Natarajan [5] and Northcutt et al. [6] methods can be used to any probability classifier and are time-efficient. Both methods provide theory support that the expected risk function for the modified loss function is equivalent to the expected risk on the original clean dataset. Except Natarajan [5] method, the other two methods also rely on the probability estimation robustness of the chosen classifier. However, Natarajan [5] did not provide any method for noise rates estimation. On the other hand, Northcutt et al. [6] provided a method to estimate flip rates values which can be opened to added noise. To be best of our knowledge, this is most time-efficient and consistent method to measure noise rates up to date.

For Instance and Class - Dependent label noise, assuming the noise rates are upper bounded, Cheng et al. [14] proposed to tackle this problem in 2 cases. When the marginal distributions have non-overlapping supports, their method can correctly identify and re-assign true labels to mislabeled examples and use that cleansed dataset for training. With overlapping case, their method is able to boost up the performance of the classifier with only a few noisy samples need to be corrected. Yet how to effectively measure the noise rates in this situation remains an open question.

3 Methods

3.1 Label noise methods

In this paper we define $\rho_0 = P(S = 1|Y = 0)$ and $\rho_1 = P(S = 0|Y = 1)$ as flip rates, under the assumption that the label noise is created by independently flip a certain number of correct labels such that $\rho_0 + \rho_1 < 1$.

3.1.1 Importance Reweighting

Importance reweighting is a method for learning from data with noisy labels, described in [4]. Given that the data labels have been corrupted with constant, class-dependent probabilities (ρ_1 and ρ_0), the method attempts to reweight each data point according to its likelihood of being mislabelled. This technique has also found use in the emerging area of domain adaptation.

Importance reweighting makes use of the observation that

$$\begin{aligned} R_{l,D}(f) &= R[D, f, l] = E_{(X,Y)\sim D}[l(f(X), Y)] \\ &= E_{(X,\hat{Y})\sim D_\rho}[\frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})} l(f(X), \hat{Y})] \\ &= R[D_\rho, f, \beta(X,\hat{Y})l(f(X), \hat{Y})] \\ &= R_{\beta l, D_\rho}(f), \end{aligned}$$

where R represents the empirical risk under surrogate loss function l and decision function f, and distributions D and D_ρ are the underlying distributions of the clean and noisy datasets respectively.

The challenge, therefore, is in estimating the reweighting factor β for each point in the noisy dataset. When the label flip rates ρ_1 and ρ_0 are known, this is a straightforward exercise. We can rewrite $\beta(X, \hat{Y})$ as

$$\beta(X,\hat{Y}) = \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})} \tag{1}$$

$$= \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)} \tag{2}$$

$$= \frac{P_{D_\rho}(\hat{Y}|X) - \rho_{1-\hat{Y}}}{(1-\rho_1-\rho_0)P_{D_\rho}(\hat{Y}|X)}. \tag{3}$$

$$\tag{4}$$

In order to evaluate $\beta(X, \hat{Y})$, we need to find $P_{D\rho}(\tilde{Y}|X)$. In this paper $P_{D\rho}(\tilde{Y}|X)$ is estimated by building a probabilistic classifier on noisy labelled data. We use a convolutional neural network and a logistic regression to estimate the posterior probability of the noisy labelled data. In the case of the logistic regression classifier, the posterior probability over the noisy labelled dataset becomes:

$$P_{D\rho}(\hat{Y}|X, f) = \frac{1}{1 + \exp(-\hat{Y}f(X))} \tag{5}$$

The algorithm for Importance Reweighting is summarised in Algorithm 1:

Algorithm 1 Importance Reweighting

Input: training data *X*, noisy labels $\hat{Y}$, classifier clf, flip rates ρ_1 and ρ_0
Output: clf

1: Prepare dataset
2: Build a probabilistic classifier on the noisy labelled dataset to find $P(\tilde{Y}|X)$
3: Compute $\beta(X, \tilde{Y})$ using 1 for every noisy label $\tilde{Y}$
4: Form a weight vector W using the $\beta(X, \tilde{Y})$ computed for every label in the above step.
5: Modify the loss function by fitting the classifier clf using X, $\tilde{Y}$, and weight vector W)
6: Return the model learned i.e. $f(X)$

3.1.2 Unbiased Estimator

Developed by Nagarajan et al. [5], the Unbiased Estimator technique for learning from noisy labels modifies the loss function. The loss function is thus termed as a label dependent loss function. It constructs the modified loss function by reweighting the loss such that the following holds true:

$$E_{\tilde{y}}[\tilde{l}(t, \tilde{y})] = l(t, y) \tag{6}$$

where y is the clean label, $\tilde{y}$ is the noisy label, t is the prediction and $\tilde{l}(f(x), \tilde{y})$ is the modified loss function. Such a modification allows us to get performance bounds over empirical risk minimisation. While developing this method, Nagarajan et al. [5] assume a class conditional random label noise. This method takes into consideration that the probability of mislabelling a data point ρ_0 and ρ_1 are given. They also assume that the data is independent and identically distributed, sampled from a distribution $\mathcal{D}$ and the noisy data is sampled from the distribution $\mathcal{D}_\rho$. The empirical risk over the noisy labeled data is defined as:

$$\hat{R}_{\tilde{l}}(f) = \frac{1}{n} \sum_{i=1}^{n} \tilde{l}(f(X_i), \tilde{Y}_i), \tag{7}$$

where n is the total number of observations.

The expected risk over the noisy data is defined as

$$R_{\tilde{l}, D_\rho}(f) = E_{(X,\tilde{Y})\sim D_\rho}[\tilde{l}(f(X), \tilde{Y})]. \tag{8}$$

The expected risk under clean distribution is defined as

$$R_{l,D}(f) = E_{(X,Y)\sim D}[l(f(X), Y)]. \tag{9}$$

For a given f, as n increases, the unbiased estimator of loss function (6) helps in converging of (7) to (9), despite the fact that (7) is computed on noisy data and (9) on true data. However one limitation of the above method is that the modified loss function may become non-convex even if the original loss function is convex.

Another result mentioned in Nagarajan et al. [5] is the method of label dependent cost. Under this approach they state that in case of any surrogate loss function, there exists an α weighted surrogate loss function for $\alpha \in [0, 1]$ such that the expected loss on the noisy data has the same sign as Bayes classifier built on 0-1 loss on clean data. The empirical risk under noisy labels is given as:

$$\hat{f}_\alpha = \arg\min_{f\in\mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} l_\alpha(f(X_i), \tilde{Y}_i), \tag{10}$$

where l_α is α weighted loss function, which is defined as

$$l_\alpha(t, y) = (1 - \alpha)1_{y=1} l(t) + \alpha 1_{y=-1} l(-t). \tag{11}$$

Nagarajan et al. [5] provide the best value of α denoted as α^* in their results, which is given by

$$\alpha^* = \frac{1 - \rho_1 + \rho_0}{2}. \tag{12}$$

In this paper we implement the label dependent cost method of Nagarajan et al. [5]. The algorithm for the implementation is given in Algorithm 2

Algorithm 2 Label Dependent Cost

Input: training data *X*, noisy labels *S*, classifier *clf*, flip rates ρ_0 and ρ_1
Output: clf

1: Prepare dataset
2: Calculate α^* as per (12)
3: Prepare a weight array W such that W[i] = α^* if S[i]= 0 else W[i] = 1- α^*
4: Modify the loss function by calling clf.fit(X, S, sample_weight = W)
5: Return the model learned i.e. $f(X)$

3.1.3 Rank Pruning

Developed by Northcutt et al. [6], the idea of Rank Pruning is to identify "confident" examples, which are likely to be correctly labeled and remove the "unconfident" examples from the original corrupted set, leaving a subset containing only correctly labeled data for the training process. The number of "unconfident" examples is obtained by estimating the inverse flip rates $\pi_0 = P(Y = 1|S = 0)$, which is the fraction of mislabeled examples in the observed negative set (denoted by $\tilde{N}, \tilde{N} = \{X|S = 0\}$) and $\pi_1 = P(Y = 0|S = 1)$, which is the fraction of mislabeled examples in the noisy positive set (denoted by $\tilde{P}, \tilde{P} = \{X|S = 1\}$).

By Bayes Rule we have

$$\pi_1 = P(Y = 0|S = 1) = \frac{P(S = 1|Y = 0)P(Y = 0)}{P(S = 1)} = \frac{\rho_0(1 - P(Y = 1))}{P(S = 1)} \tag{13}$$

On the other hand,

$$\begin{aligned} P(S = 1) &= P(Y = 0, S = 1) + P(Y = 1, S = 1) \\ &= P(S = 1|Y = 0)P(Y = 1) + P(S = 1|Y = 1)P(Y = 1) \\ &= \rho_0(1 - P(Y = 1)) + (1 - \rho_1)P(Y = 1) \\ &= P(Y = 1)(1 - \rho_0 - \rho_1) + \rho_0 \end{aligned}$$

Thus

$$P(Y = 1) = \frac{P(S = 1) - \rho_0}{1 - \rho_0 - \rho_1} \tag{14}$$

Substituting (14) to (13) we have

$$\pi_1 = \frac{\rho_0}{P(S = 1)} \frac{1 - \rho_1 - P(S = 1)}{1 - \rho_0 - \rho_1} \tag{15}$$

Similarly,

$$\pi_0 = \frac{\rho_1}{1 - P(S = 1)} \frac{P(S = 1) - \rho_0}{1 - \rho_0 - \rho_1} \tag{16}$$

Next, we remove $\pi_1|\tilde{P}|$ examples in $\tilde{P}$ having the lowest predicted probabilities $g(X) = P(\hat{S} = 1|X)$ obtained from a classifier such as Logistic Regression, SVM or Neural Networks on noisy data and $\pi_0|\tilde{N}|$ examples in $\tilde{N}$ having the highest $g(X)$. The remaining subsets $\tilde{P}_{conf}$ and $\tilde{N}_{conf}$ are then combined to X_{conf} as training examples and refitted to the chosen classifier by reweighting

the loss function for samples in $\tilde{P}_{conf}$ with weight $\frac{1}{1-\rho_1}$ and samples in $\tilde{N}_{conf}$ with weight $\frac{1}{1-\rho_0}$ in order to recover the estimated balance of positive and negative samples [6]. The predicted labels from such classifier $\hat{f}$ is denoted as $\hat{y}_i$ for sample i.

Let $l(\hat{y}_i, s_i)$ be the original loss function for $x_i \in \mathcal{D}_\rho$. The loss function for Rank Pruning method is thus defined as

$$\tilde{l}(\hat{y}_i, s_i) = \frac{1}{1-\rho_1} l(\hat{y}_i, s_i) \cdot \mathbf{1}(x_i \in \tilde{P}_{conf}) + \frac{1}{1-\rho_0} l(\hat{y}_i, s_i) \cdot \mathbf{1}(x_i \in \tilde{N}_{conf}) \tag{17}$$

with 0 weight for the removed samples.

In their paper, Northcutt et al. [6] proved that the classifier $\hat{f}$ using Rank Pruning method from the corrupted dataset could be learned for the hidden dataset $\mathcal{D}$ by minimizing the empirical risk function

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}_{\tilde{l}, \mathcal{D}_\rho}(f) = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \tilde{l}(f(x_i), s_i) \tag{18}$$

Furthermore, for any classifier f and any bounded loss function $l(\hat{y}_i, y_i)$,

$$R_{\tilde{l}, \mathcal{D}_\rho}(f) = R_{l, \mathcal{D}}(f) \tag{19}$$

From (19), Rank Pruning method can yield equivalent expected risk as learning with the original clean dataset. Therefore, Rank Pruning can exactly uncover the classification f fit to the correct Y labels, given the exact flip rates.

The Rank Pruning method is illustrated in Algorithm 3

Algorithm 3 Rank Pruning algorithm

Input: training data *X*, noisy labels *S*, classifier *clf*, flip rates ρ_0 and ρ_1
Output: clf

1: clf.fit(X, S)
2: $g(X) \leftarrow clf$.predict_proba$(\hat{S} = 1|X)$
3: $\pi_1 = \frac{\rho_0}{P(S=1)} \frac{1-\rho_1-P(S=1)}{1-\rho_0-\rho_1}$, $\pi_0 = \frac{\rho_1}{1-P(S=1)} \frac{P(S=1)-\rho_0}{1-\rho_0-\rho_1}$
4: Remove $\pi_1|\tilde{P}|$ examples in $\tilde{P}$ having the lowest predicted probabilities $g(X)$, and $\pi_0|\tilde{N}|$ examples in $\tilde{N}$ having the highest $g(X)$. The remaining dataset is denoted as X_{conf}, S_{conf}.
5: clf.fit(X_{conf}, S_{conf}, sample weight = $\frac{1}{1-\rho_1} \cdot \mathbf{1}(S_{conf} = 1) + \frac{1}{1-\rho_0} \cdot \mathbf{1}(S_{conf} = 0)$)

3.2 Estimation of flip rates

Both our three methods require the knowledge of flip rates ρ_0 and ρ_1, which is normally not given in real scenarios. Liu et al. [4] proposed using minimum predicted probabilities as flip rates.

$$\rho_{-y} = \min_{x \in \mathcal{X}} P(\tilde{S} = y|X) \tag{20}$$

where $\mathcal{X}$ is a set of x_{+1}, x_{-1} such that $P_{\mathcal{D}}(y = 1|x_{-1}) \approx 0$ and $P_{\mathcal{D}}(y = -1|x_{+1}) \approx 0$.

Thus, the values of flip rates heavily depend on the robustness of the chosen classifier and often result in $\min P(\hat{Y} = y|x) = 0$.

In this subsection, we present another method developed by Northcutt et al. [6] to estimate the flip rates.

Denote the fraction of negative label examples ($S = 0$) that "confidently" have positive label ($Y = 1$) to the total confident positive label examples as $\hat{\rho_1}^{conf}$. Similarly, $\hat{\rho_0}^{conf}$ is the fraction of positive label examples ($S = 1$) that are confidently have negative label ($Y = 0$) to the total confident positive label examples. The confident counts $\hat{\rho}_1^{conf}$ and $\hat{\rho}_0^{conf}$ are then used to estimate ρ_1 and ρ_0 respectively.

$$\hat{\rho}_1^{conf} \coloneqq \frac{|\tilde{N}_{y=1}|}{|\tilde{N}_{y=1}| + |\tilde{P}_{y=1}|}; \qquad \hat{\rho}_0^{conf} \coloneqq \frac{|\tilde{P}_{y=0}|}{|\tilde{P}_{y=0}| + |\tilde{N}_{y=0}|} \tag{21}$$

such that

$$\begin{cases} \tilde{P}_{y=1} = \{X \in \tilde{P} |\ g(X) \geq LB_{y=1}\} \\ \tilde{N}_{y=1} = \{X \in \tilde{N} |\ g(X) \geq LB_{y=1}\} \\ \tilde{P}_{y=0} = \{X \in \tilde{P} |\ g(X) \leq UB_{y=0}\} \\ \tilde{N}_{y=0} = \{X \in \tilde{N} |\ g(X) \leq UB_{y=0}\} \end{cases}, \tag{22}$$

where $g(X) = P(\hat{S} = 1|X)$ obtained from a classifier on corrupted data $\mathcal{D}_\rho$. $LB_{y=1}$ can be seen as a threshold above which a sample X most likely has true label $Y = 1$. Similarly, $UB_{y=0}$ is also an upper threshold for samples having hidden label $Y = 0$. These thresholds are defined as

$$\begin{cases} LB_{y=1} \coloneqq P(\hat{S} = 1|\ S = 1) = E_{X \in \tilde{P}}[g(X)] \\ UB_{y=0} \coloneqq P(\hat{S} = 1|\ S = 0) = E_{X \in \tilde{N}}[g(X)] \end{cases} \tag{23}$$

In their paper, Northcutt et al. [6] also prove that ρ_0 and ρ_1 are upper bounded by $\hat{\rho}_0^{conf}$ and $\hat{\rho}_1^{conf}$ respectively. Furthermore, under certain conditions, $\rho_0 = \hat{\rho}_0^{conf}$ and $\rho_1 = \hat{\rho}_1^{conf}$, making $\hat{\rho}_0^{conf}$ and $\hat{\rho}_1^{conf}$ a consistent and robust estimation for the asymmetric flip rates.

A step by step estimation of flip rates is illustrated in Algorithm 4:

Algorithm 4 Estimation of flip rates

Input: training data *X*, noisy labels *S*, classifier *clf*
Output: $\hat{\rho}_1$ and $\hat{\rho}_0$

1: clf.fit(X, S)

2: $g(X) \leftarrow clf$.predict_proba$(\hat{S} = 1|X)$

3: $P(S = 1) = \frac{\text{count}(S=1)}{\text{len}(S)}$

4: $LB_{y=1} = E_{X \in \tilde{P}}[g(X)],\ UB_{y=0} = E_{X \in \tilde{N}}[g(X)]$

5: $\hat{\rho}_1 = \hat{\rho}_1^{conf} \coloneqq \frac{|\tilde{N}_{y=1}|}{|\tilde{N}_{y=1}| + |\tilde{P}_{y=1}|},\ \hat{\rho}_0 = \hat{\rho}_0^{conf} \coloneqq \frac{|\tilde{P}_{y=0}|}{|\tilde{P}_{y=0}| + |\tilde{N}_{y=0}|}$

4 Experiments

4.1 Experimental setup

4.1.1 Datasets

The datasets used for comparing the robustness of learning against noisy labels are the MNIST fashion and CIFAR datasets. Both datasets are widely used among the research fraternity due to their simplicity and practicality.

MNIST fashion dataset is prepared by Zalando, an electronic commerce company based in Berlin [15]. The dataset originally consists of 60,000 greyscale images belonging to 10 classes of fashion

products as training set and 10,000 greyscale images as test dataset, each of size 28×28. In this paper we are given 10,000 images for training and a test set of 2,000 images for binary classification.

CIFAR-10 dataset is prepared by Alex Krizhevsky, Vinod Nair and Geofferey Hinton from Canadian Institute for Advanced Research [16]. The dataset consists of 50,000 coloured training images and 10,000 test images in 10 classes. The size of each image is 32×32. Each pixel also contains red, green and blue channel values. For the experiments in the paper, we are given 10,000 training images and 2,000 test images for binary classification.

4.1.2 Data Preprocessing

As part of data preprocessing, both datasets were normalised by dividing the pixel values by 255. Furthermore, in case of MNIST fashion dataset, each image had size 28×28 which was then converted to a vector of size 784×1. The CIFAR dataset has images of size $32 \times 32 \times 3$ which were then reshaped into a vector of size 3072×1.

4.1.3 Experimental design

Estimation of flip rates

We experiment the two methods for estimating the flip rates proposed by Liu et al. [4] using minimum predicted probabilities (denoted as minP) and Northcutt et al. [6] in Rank Pruning method (denoted as RP). For each dataset, we randomly select 8,000 samples from the training data and use Logistic Regression and Convolutional Neural Network (CNN) classifiers to predict the probabilities, as described in section 3. In order to avoid overfitting, we use k-fold cross-validation when estimating the flip rates. We then repeat the process 10 times and evaluate the average over 10 times of each rate ρ_0 and ρ_1 by comparing to the exact given values $\rho_0 = 0.2$ and $\rho_1 = 0.4$.

Label noise methods

For each dataset, we randomly select 8,000 samples from 10,000 training samples as training data. Given flip rates, we combine each label noise method with simple (Logistic Regression) and complex (CNN) classifiers. A model using only Logistic Regression or CNN on the noisy data is also implemented as Baseline for evaluation. For fair comparison, the same simple CNN model of 2 layers with Max Polling and Dropout, epochs = 20 is applied to each method. We repeat this process 10 times for different random sets of training data.

Hardware and software specifications

The experiments are run on Google Colaboratory (2-core Intel(R) Xeon(R) CPU @ 2.30GHz, 13GB RAM) with free Tesla K80 GPU. The external libraries used in our codes include sklearn and keras using Tensorflow backend.

4.1.4 Evaluation Metric

The metric used for evaluating different algorithms is Accuracy. Accuracy is defined as follows:

$$\text{Accuracy} = \frac{\text{number of correctly classified examples}}{\text{total number of examples}} \times 100\% \tag{24}$$

In the results section, we use the metric of average accuracy. Average accuracy is calculated on the test data over 10 models that are being built by randomly selecting 8,000 samples from training data.

4.2 Experimental results and Discussion

4.2.1 Estimation of flip rates

The results of two methods using two different classifiers are presented in Table 1 for MNIST dataset and Table 2 for CIFAR dataset. The better estimation is highlighted in bold.

Table 1: Estimation of flip rates for MNIST dataset

MNIST		True	minP		RP	
			Estimation	Deviation from true value	Estimation	Deviation from true value
Logistic Regression	ρ_1	0.4	0.0109 ± 0.0061	-97.28%	**0.4294** $\pm$ **0.0048**	7.35%
	ρ_0	0.2	0.0029 ± 0.0023	-98.55%	**0.2423** $\pm$ **0.0040**	21.15%
CNN	ρ_1	0.4	0.2292 ± 0.0179	-42.7%	**0.4077** $\pm$ **0.0087**	1.93%
	ρ_0	0.2	0.1388 ± 0.0175	-30.6%	**0.2035** $\pm$ **0.0074**	1.75%

Table 2: Estimation of flip rates for CIFAR dataset

CIFAR		True	minP		RP	
			Estimation	Deviation from true value	Estimation	Deviation from true value
Logistic Regression	ρ_1	0.4	0.0019 ± 0.0016	-99.53%	**0.5151** $\pm$ **0.0042**	28.78%
	ρ_0	0.2	0.0013 ± 0.0006	-99.35%	**0.3348** $\pm$ **0.0036**	67.4%
CNN	ρ_1	0.4	0.1708 ± 0.0172	-57.3%	**0.4280** $\pm$ **0.0114**	7%
	ρ_0	0.2	0.1518 ± 0.202	-24.1%	**0.2306** $\pm$ **0.0094**	15.3%

As can be seen from Table 1 and 2, RP method outperforms minP in all experiments of flip rate estimation. 6/8 of the experiments with RP method have errors of less than 0.05 and small standard deviations (highlighted in red). This is because RP method is less sensitive to the predicted ability of the chosen classifier. Since ρ_0 and ρ_1 are estimated by the upper bounds $\hat{\rho}_0^{conf}$ and $\hat{\rho}_1^{conf}$ respectively, RP method tends to overestimate the flip rates, which accounts for the difficulty of more complex problems. But at the same time, it may lead to more samples being pruned when combined with Rank Pruning method, which might contain some correctly labeled samples [6].

As expected, the complex CNN classifier yields significantly better results than a simple classifier such as Logistic Regression, but on the other hand results in larger standard deviations. One thing to note is that in this exercise, we randomly select 8,000 samples from the set of 10,000 training data for each running time, therefore, the given values $\rho_1 = 0.4$ and $\rho_0 = 0.2$ may not reflect the true values of flip rates in each subset.

4.2.2 Label noise experiments

Running time

The total execution time (10 times running) of our label noise experiments is reported in Table 3 and 4.

Table 3: Running time (by seconds) of 3 label noise methods for MNIST dataset

Classifier	Baseline	Importance Reweighting	Unbiased Estimator	Rank Pruning
Logistic Regression	54	92	51	494
CNN	636	1111	674	1886

Unbiased Estimator achieves the lowest execution time because this method only calls for 1 fitting time in total, whereas the other two methods have to call at least 2 fitting times. Precisely, Unbiased Estimator takes $\mathcal{O}(n)$ time prior to the final fitting stage, on the other hand, both Importance

Table 4: Running time (by seconds) of 3 label noise methods for CIFAR dataset

Classifier	Baseline	Importance Reweighting	Unbiased Estimator	Rank Pruning
Logistic Regression	830	1628	698	1905
CNN	628	1211	640	2087

Reweighting and Rank Pruning take $\mathcal{O}(T) + \mathcal{O}(n)$ where *T* is the fitting time of a specific classifier. Furthermore, in order to avoid overfitting for Rank Pruning method, we use k-fold cross validation (k = 3) in the process of estimating the number of samples to be pruned, thus the execution time of Rank Pruning method is on the high side.

Classification Accuracy

The comparison of the accuracy of three label noise methods using different classifiers on the two given datasets are shown in Table 5 and 6. The best performances are highlighted in bold for each category.

Table 5: Classification accuracies (percentage) of 3 label noise methods for MNIST dataset

Classifier	Baseline	Importance Reweighting	Unbiased Estimator	Rank Pruning
Logistic Regression	79.18 ± 0.52	$\mathbf{88.09 \pm 0.61}$	87.41 ± 0.57	85.61 ± 0.37
CNN	89.86 ± 2.81	$\mathbf{95.01 \pm 0.58}$	94.68 ± 0.55	93.06 ± 1.02

Table 6: Classification accuracies (percentage) of 3 label noise methods for CIFAR dataset

Classifier	Baseline	Importance Reweighting	Unbiased Estimator	Rank Pruning
Logistic Regression	66.65 ± 0.61	$\mathbf{70.81 \pm 0.49}$	70.48 ± 0.59	70.29 ± 0.58
CNN	86.72 ± 1.54	80.18 ± 3.1	82.64 ± 5.09	$\mathbf{88.98 \pm 0.68}$

From table 5 and 6, in general, we can observe the significant improvements of both three methods compared to the baseline. Some methods even result in almost 10% improvement. Thus, this result proves the robustness of our chosen methods in addressing label noise problems. Both three methods have very small standard deviations. CNN models achieve much better results than Logistic Regression, since the normal Logistic Regression classifier is heavily affected by label noise. Even without the presence of label noise, Neural Networks has long been seen as the state-of-the-art method for image classification problems.

In the case of the simpler MNIST dataset, the Importance Reweighting method achieves the highest average accuracies with both Logistic Regression and CNN classifiers. The other two methods perform slightly worse than Importance Reweighting, although they perform significantly better than the baseline classifier.

For the more complex CIFAR dataset, both three methods achieve the same performance when using a Logistic Regression classifier. However, when combining with CNN classifier, Importance Reweighting and Unbiased Estimator do not perform well compared to the baseline with standalone CNN model. We suspect this is due to the robustness of the CNN method in handling noisy data by itself. To explain this, Rolnick et al. [17] show the capability of Deep Neural Networks in generalising after training on massive data, not only by merely memorising the noise. With a sufficient

amount of training data, Deep Neural Networks can cover a wide range of noise levels, even when the data is massively dominated by incorrect labels. Drory et al. [18] also examines the resistance of CNN to label noise that is randomly spread across the training data. On the other hand, Rank Pruning method shows the consistent robustness to label noise.

Furthermore, we can clearly see that the accuracies of our chosen methods also tie back to the robustness of the chosen classifier, because the last step of both three methods involves the classifier fitting process in order to make final prediction on test data. Hence, it would be more efficient if we are able to test the performance of each classifier on the uncorrupted datasets, so that we could have a clearer analysis on the robustness of our chosen label noise methods.

5 Conclusion and future work

In this paper we have attempted to test the robustness of 3 methods in handling label noise on 2 real-life and commonly used datasets, CIFAR and MNIST-fashion. The performance of all three methods shows a significant improvement compared to a standalone classifier on noisy data. In our experiments, Unbiased Estimator appears to be the most time-efficient method. Importance Reweighting achieves highest performance results compared to other 2 methods, except for the case of more complex dataset (CIFAR) and more complex classifier (CNN). In that sense, Rank Pruning performance appears to be more consistent. It is able to mitigate the dependency on probability estimation ability of the chosen classifier, compared to Importance Reweighting. Therefore, in practice, depending on the nature of the dataset, noise levels, how fast the algorithm should run and other criteria, we can choose an appropriate approach to tackle the given problem.

Rank Pruning methodology also provides an effective way to measure flip rates, which is crucial in real-life problems. The consistency of this method even with imperfect probability estimation is also shown in our experiments.

To better examine the robustness of our chosen methods in dealing with label noise, in future work, we would attempt to experiment on different settings of flip rates, even with the extreme values. Furthermore, we can also try these methods with different classifiers such as kNN, as well as different settings of CNN model. The Importance Reweighting method can be tried with kernel density estimation method and density ratio estimation method to estimate posterior probability of noisy data. These techniques can also be implemented for the estimation of flip rates, as suggested by Liu et al. [4]. One of ambitious future work would be to establish a relationship of observations with the noisy labels which is dependent not only the class of the observation but also on the observed feature.

References

[1] X. Zhu and X. Wu, Class noise vs. attribute noise: A quantitative study, Artif. Intell. Rev., vol. 22, no. 3, pp. 177210, 2004.

[2] J. Sez, M. Galar, J. Luengo, and F. Herrera, Analyzing the presence of noise in multi-class problems: Alleviating its influence with the one-vs-one decomposition, Knowl. Inf. Syst., pp. 128, Nov. 2012, in press

[3] Aha, D. W., Kibler, D., and Albert, M. K. Instance-based learning algorithms. Mach. Learn., 6(1):3766, 1991.

[4] Liu, T., and Dacheng T. "Classification with noisy labels by importance reweighting." IEEE Transactions on pattern analysis and machine intelligence 38.3 (2016): 447-461.

[5] Natarajan, N., Dhillon, I., Ravikumar, P. and Tewari, A. "Learning with Noisy Labels." NIPS Proceedings, Part of: Advances in Neural Information Processing Systems 26 (NIPS 2013)

[6] Northcutt, C., Wu, T. and Chuang, I. Learning with Confident Examples: Rank Pruning for Robust Classification with Noisy Labels. Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2017.

[7] Menon, A. K., Jiang, X., Vembu, S., Elkan, C., and Ohno-Machado, L. "Predicting accurate probabilities with a ranking loss". CoRR, abs/1206.4661, 2012.

[8] Frnay, B., Michel V. "Classification in the presence of label noise: A survey." IEEE Transactions on Neural Networks and Learning Systems 25.5 (2014): 845-869.

[9] Miranda, A., Garcia, L., Carvalho, A., and Lorena, A., Use of classification algorithms in noise detection and elimination, in Proc. 4th Int. Conf. Hybrid Artif. Intell. Syst., Salamanca, Spain, Jun. 2009, pp. 417424.

[10] Joseph, L. and Gyorkos, T., Inferences for likelihood ratios in the absence of a gold standard, Med. Decis. Making, vol. 16, no. 4, pp. 412417, 1996.

[11] Brodley, C. and Friedl, M. Identifying mislabeled training data, J. Artif. Intell. Res., vol. 11, pp. 131167, Jun. 1999.

[12] Angluin, D., and Laird, P. 1988. Learning from noisy examples. Machine Learning 2(4):343 370.

[13] Ghosh, Aritra, Manwani N., Sastry, P. "Making risk minimization tolerant to label noise." Neurocomputing 160 (2015): 93-107

[14] Cheng, J., Liu, T., Ramamohanarao, K., Tao, D. Learning with Bounded Instance- and Label-dependent Label Noise, arXiv:1709.03768, 2017.

[15] Zalando's fashion-MNIST dataset. Available at https://github.com/zalandoresearch/fashion-mnist

[16] CIFAR dataset prepared by Canadian Institute for Advanced Research. Available at https://www.cs.toronto.edu/ kriz/cifar.html

[17] Rolnick, D.; Veit, A.; Belongie, S.; Shavit, N. "Deep Learning is Robust to Massive Label Noise". CoRR, arXiv:1705.10694, 2017.

[18] Drory, A., Avidan, S., Giryes R. "On the Resistance of Neural Nets to Label Noise", arXiv:1803.11410v1, 2018.

APPENDIX - Instruction for running the code

The code to run the analysis is provided as a set of Jupyter Python notebooks in the code folder. There are seven notebooks provided. 6 of them are for label noises methods (1 for each method/dataset or classifier choice) and one for flip rates estimation. The code is executed by running cell by cell.

The experiments are best run on Google Colab with free GPU setting. Using this method, the file should be uploaded directly to the notebook. We also experienced some difficulties when running CNN model on Jupyter Notebook due to the incompatibility of the latest python version and keras's latest version.