Assignment 2

Tutors: Zhuozhuo Tu
Yang Ge, SID:450005028, UniKey: yage9511
Rui Wang, SID:470208162, UniKey: rwan0699
Quan Chen, SID:470199228, UniKey: qche7416

Abstract

This report discusses the performance as well as robustness of two classification methods which are used to classify corrupted images in real world. In this assignment, the corrupted images indicate those images that are affected by label noise. The datasets utilized in this assignment are from two real world image databases, which are Fashion-MINIST and CIFAR, and the datasets are reconstructed with noisy labels for classification. The classification method chosen in this assignment are importance reweighting and a loss correction approach for deep neural networks, which are the basic models in our work for classification. The details of these two methods as well as the reason that deep neural network is chosen in this assignment will be discussed in this report. The report is organized as follows. The introduction will discuss the problem and its importance as well as some applications. Previous work will introduce some relevant methods to address label noise problem and their performance. The methods section will illustrate the details of our applied methods, and the experiment will talk about the experimental processes we performed, followed by conclusions and references.

1 Introduction

Classification is always a central problem in machine learning, and large real-world datasets are the most common used source for training to solve real world problem. Nevertheless, large datasets can be easily corrupted by noises such as label noise, which indicates that some labels of the data are wrongly marked. Therefore, it is important to address the label noise issue of large dataset training so that machine learning can be more helpful in solving real world problems. The application of dealing datasets with label noises can involve many aspects. For instance, social network can have many users, and each user should belong to one or more labels. This research can help to detect mislabeled users and reduce the workload of the website. One hospital can accept many patients with different symptoms, and it is effective to label all the patients according to their symptoms. However, the real cause of disease might not be accurate, making some patients mislabeled. Thus, it is necessary to identify the label noise and correct the wrong label immediately.

There are two methods utilized in this assignment. The first method is importance reweighting, which proves that any surrogate loss function can be applied for classification with label noise as long as they use importance reweighting [1]. Besides, if we assume that true labels are independently changed with a probability, the noise rate can be upper bounded. Another method is called loss correction approach, which focuses on making deep neural network robust to label noise [2]. The method consists of both forward and backward corrections. In addition to that, deep neural network has been proved to be robust to label noise [3].

2 Previous work

The problem of noisy label has always been a hot topic in machine learning classification problem. In 1994, it was showed that linear threshold functions are learnable when classification noises are added using polynomial method [4]. After that, many algorithms that can tolerate the noisy data has been proposed. For instance, binary classification in the presence of label noise can be handled using modified surrogate loss function, by using unbiased estimator and leveraging a decrease of risk minimization under where label noises exist [5]. The implementation of these methods are easy and they can be utilized to deal with high noise rates. However, these two methods only focus on class-depend label noise, which indicates that other types of label noise cannot be effectively detected.

Learning label noise can also be applied in images classification problem. Side information can be utilized in a distillation framework to avoid the risk of meeting noisy labels in image classification, which means that using the assistance of a small clean dataset to correct the noisy labels. This is a novel approach using distillation process and can deal with large size noisy data [6]. One disadvantage of this approach is that it highly relies on a clean dataset. If the dataset is extremely huge, this approach requires a lot of manually work to create a clean dataset. Convolutional neural network has also been proved to be effective in training dataset with label noise [7]. This method can deal with massive amount of noisy data using clean labels as annotations. Apart from the drawback that is mentioned in the previous method, training process can cost a lot of time.

3 Label noise methods

3.1 Pre-Processing

Standardization. In the field of machine learning, different evaluation indicators often have different dimensions and dimension units. Such situations will affect the results of data analysis. Feature standardization can make the value of mean for each feature in dataset equal to 0, and make variance equal to 1. It is a common requirement for many machine learning method. The general method of applying standardization is that we first determine the mean and standard deviation for each feature, then subtract the mean from each feature, in the end, we divide the values by corresponding standard deviation.

$$x' = \frac{x - \bar{x}}{\delta} \tag{1}$$

Where x is the original feature vector, $\bar{x}$ is the mean, δ is the standard deviation correspondingly.

One hot encoding. One hot encoding is a method to represent class the data belongs to. The idea is to assign 0 to all the dimension except 1 for the class the data belongs to. Because of the CNN model's softmax cross-entropy loss, the ground truth numeric or categorical label need to be transformed to One hot encoding format.

3.2 Classification Method

Before introducing the classification methods, the reason why deep neural network is chosen as our classification method should be explained. Deep neural network has been proved to perform well when large size of noises are added to the data, and the accuracy will increase when the size of training data increase [3]. This work used real world corrupted images to train and showed that deep neural network has good robustness to uniform noise, structured label noise and source of noisy labels [3]. The explanation for the robustness of deep neural network is that during back propagation, "gradient updates from randomly sampled noisy labels roughly cancel out, while gradients from correct examples that are marginally more frequent sum together and contribute to learning" [3]. In this case, we choose deep neural network as our classification approach. Next part will introduce the two method utilized in this assignment.

3.3 Importance Reweighting

For this assignment, there are two categories: 0 and 1. Define the noise rates as:

$$\rho_0 = P(\hat{Y} = 1|Y = 0, X), \rho_1 = P(\hat{Y} = 0|Y = 1, X), \tag{2}$$

where $\hat{Y}$ is the observed noisy label and Y is the unobservable true label. X is the feature of the dataset. For Class-Dependent Noise (CCN):

$$\rho_Y(X) = P(\hat{Y}|Y, X) = P(\hat{Y}|Y). \tag{3}$$

The goal is to modify the loss function $\tilde{L}$, which is the loss function with the presence of noisy label, to mach the loss function with real label, so that:

$$arg\min_{f\in \boldsymbol{F}} R_{D,L}(f) = arg\min_{f\in \boldsymbol{F}} R_{D_\rho,\tilde{L}}(f). \tag{4}$$

Importance reweighting is widely used for domain adaptation, here we introduce it to classification in the presence of label noise. If we apply importance reweighting:

$$\begin{aligned} R_{D,L}(f) &= E_{(X,Y)\sim D}[L(f(X), Y)] \\ &= \int P_D(X,Y)L(f(X),Y)dXdY \\ &= \int P_{D_\rho}(X,Y)\frac{P_D(X,Y)}{P_{D_\rho}(X,Y)}L(f(X),Y)dXdY \\ &= E_{(X,Y)\sim D}[\beta(X,Y)L(f(X),Y)], \end{aligned} \tag{5}$$

where

$$\beta(x,y) = \frac{P_D(X=x, Y=y)}{P_{D_\rho}(X=x, \hat{Y}=y)}. \tag{6}$$

For the problem of classification in the presence of label noise, we have $P_D(X) = P_{D_\rho}(X)$. Therefore, we have

$$\begin{aligned} \beta(X,\hat{Y}) &= \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})} \\ &= \frac{P_D(Y|X)P_D(X)}{P_{D_\rho}(\hat{Y}|X)P_{D_\rho}(X)} \\ &= \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)} \end{aligned}$$

Thus classification can still be implemented if only the weight $\beta(X,\hat{Y}) = \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)}$ could be accessed to the loss $l(f(X),\hat{Y})$. We can also get

$$\begin{aligned} \beta(X,\hat{Y}) &= \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)} \\ &= \frac{P_{D_\rho}(\hat{Y}|X) - \rho_{1-\hat{Y}}}{(1-\rho_1-\rho_0)P_{D_\rho}(\hat{Y}|X)} \end{aligned}$$

This is the weight given to the noisy example $(X,\hat{Y}) \sim D$. Then a classifier can deal with label noise by minimizing following reweighted empirical risk

$$\begin{aligned} \hat{f}_n &= arg\min_{f\in \boldsymbol{F}} \hat{R}_{\beta l, D_\rho} \\ &= arg\min_{f\in \boldsymbol{F}} \frac{1}{n}\sum_{i=1}^{n} \beta(X_i,\hat{Y}_i)l(f(X_i),\hat{Y}_i) \end{aligned}$$

where

$$\beta(X_i,\hat{Y}_i) = \frac{P_{D_\rho}(\hat{Y}_i|X_i) - \rho_{1-\hat{Y}_i}}{(1-\rho_1-\rho_0)P_{D_\rho}(\hat{Y}_i|X_i)}$$

In this case, the key process is to calculate $\beta(X,\hat{Y})$ and multiply with the original loss function. To do that, the most important procedure is to calculate the conditional distribution $P_{D_\rho}(\hat{Y}|X)$. The paper provided two approaches to estimate the probability, which are probabilistic classification method and the kernel density estimation method [1].

3.4 Loss Correction

In this method, asymmetric noise is contained, which is the class-dependent label noise. The goal of loss correction is to modify the loss, making it robust to label noise. There are two procedures to correct the loss. For a given dataset, we have

$$p(x, \tilde{y}) = \Sigma_y p(y|\tilde{y}) p(yx) p(x), \tag{7}$$

where is y the correct label, $\tilde{y}$ is the corrupted label and $p(y|\tilde{y})$ is the probability that y turns into $\tilde{y}$. Assume that the noise matrix is T. It contains the probability of one label being flipped into another label, and

$$T_{ij} = p(\tilde{y} = e^j | y = e^i). \tag{8}$$

3.4.1 Backward correction

The the backward corrected loss can be defined as

$$l^{\leftarrow}(\hat{p}(y|x)) = T^{-1} l(\hat{p}(y|x)), \tag{9}$$

where l is the loss we try to modify, $\hat{p}(y|x)$ represents the class-dependent probability and $l^{\leftarrow}$ is the modified loss. After that, the loss correction will become unbiased that for any x:

$$E_{\hat{y}|x} l^{\leftarrow}(y, \hat{p}(y|x)) = E_{y|x} l(y, \hat{p}(y|x)). \tag{10}$$

In this case, the minimizers will also be the same:

$$\underset{\hat{p}(y|x)}{argmin} E_{x,\bar{y}} l^{\leftarrow}(y, \hat{p}(y|x)) = \underset{\hat{p}(y|x)}{argmin} E_{x,y} l(y, \hat{p}(y|x)), \tag{11}$$

which indicates that the two losses have the same minimizer. This corrected loss is a linear combination of the noise matrix and the original loss, making it differentiable and can be optimized backward since it multiply the loss by T^{-1}.

3.4.2 Forward correction

Similar to the backward correction, forward correction procedure multiply the predictions by T:

$$l(\hat{p}(y|x)) = -log T^T \hat{p}(y|x). \tag{12}$$

For both backward and forward correction, if T is known, we can directly aplly the algorithm. If not, we need to estimate the matrix T using a robust two-stage training algorithm [2] the paper provided. This algorithm is shown in section 4.2. [2]

3.5 Cross Validation

If we simply use all training data to train our model, we could get a model that performs well on training data but may perform poorly on test data. What we want is to get a model that has good performance in general data. Cross-validation is a method that can assess how well a model generalizes by training on a portion of the instances and testing on the remaining. In this assignment, we randomly sample 8,000 training samples from the 10,000 training examples and use the sampled 8,000 examples as new training examples to train the model, then repeat this process and check the average performance.

4 Noise rate estimation methods

4.1 Method one

We have following equations

$$P_{D_\rho}(\hat{Y} = 1 | X = \boldsymbol{x}) = (1 - \rho_0 - \rho_1) P_D(Y = 1 | X = \boldsymbol{x}) + \rho_0 \tag{13}$$

$$P_{D_\rho}(\hat{Y} = 0 | X = \boldsymbol{x}) = (1 - \rho_0 - \rho_1) P_D(Y = 0 | X = \boldsymbol{x}) + \rho_1 \tag{14}$$

Where $1-\rho_0-\rho_1$ and $P_D(Y|X)$ are greater than 0, then we have that

$$\rho_0 \leq P_{D_\rho}(\check{Y}=1|X) \tag{15}$$

$$\rho_1 \leq P_{D_\rho}(\check{Y}=0|X) \tag{16}$$

Moreover, if there exists $x_0, x_1 \in \boldsymbol{X}$, such that $P_D(Y=1|x_0) = P_D(Y=0|x_1) = 0$, we have

$$\rho_0 = P_{D_\rho}(\check{Y}=1|x_0) \tag{17}$$

$$\rho_1 = P_{D_\rho}(\check{Y}=0|x_1) \tag{18}$$

which means

$$\rho_0 = \min_{x\in\boldsymbol{X}} P_{D_\rho}(\check{Y}=1|X) \tag{19}$$

$$\rho_1 = \min_{x\in\boldsymbol{X}} P_{D_\rho}(\check{Y}=0|X) \tag{20}$$

4.2 Method two

Here $\boldsymbol{T}$ denote the flip rate matrix, which is illustrated in [2]. We calculate it by using formula below:

$$\overline{\boldsymbol{x}}^i = \arg\max_{x\in X^{'}} \hat{p}(\tilde{\boldsymbol{y}} = \boldsymbol{e}^i|\boldsymbol{x}) \tag{21}$$

$$\hat{T}_{ij} = \hat{p}(\tilde{\boldsymbol{y}} = \boldsymbol{e^j}|\overline{\boldsymbol{x}}^i) \tag{22}$$

Algorithm 1 T estimation algorithm

Require: The noisy training set S, any loss$l$
  if T is unknown: **then**
    Train a network $h(x)$ on S with loss l
    Use data set $X^{'}$ to estimate $\hat{T}$ by equation (15)-(16)
  end if
  Train the network $h(x)$ on S with loss l

Here the $X^{'}$ in our experiment we apply the baldiation

5 Experiments and discussions

In this experiment we set two benchmark for comparison and apply two methods to calculate the flip rate and use it to correct CNN model's loss function.

5.1 Experiments

5.1.1 Compare different classification method benchmark

The data in our experiment is MNIST and Cifar. In each of them we select two classes and set the flip rate $p(S=1|Y=0)=0.2$ and $p(S=0|Y=1)=0.4$. In order to gain a good accuracy and better estimation on flip rate, we apply method two to estimate the flip rate and use this given flip rate in our two CNN models, which including sample reweightting and loss function correction. There are two benchmark models which are SVM and MLP. They are regarded as benchmark in this experiment in order to compare our model and method, which is showed in Table 1. From this table result we choose CNN as our baseline model and implement it in the importance reweighting and loss function correction method. In each experiment we randomly select **80**% of data as training data and the rest are validation data. Validation data play a role of fine tuning and model selection. This procedure repeat **ten times** and then we get the average score as reference for avoiding model comparison bias. Finally we implement the trained model with fixed parameter to predict the test data and evaluate their accuracy score. The structure of our model is showed in Figure 1.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 28, 28, 64)        320
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 14, 14, 64)        16448
_________________________________________________________________
flatten_12 (Flatten)         (None, 12544)             0
_________________________________________________________________
dense_36 (Dense)             (None, 1000)              12545000
_________________________________________________________________
dense_37 (Dense)             (None, 500)               500500
_________________________________________________________________
dense_38 (Dense)             (None, 2)                 1002
=================================================================
Total params: 13,063,270
Trainable params: 13,063,270
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: CNN structure

Data set	Model	Parameter	Traning Accuracy	Validation Accuracy	Test Accuracy
MNIST	SVM	Poly, C=20	0.898±0.004	0.643±0.020	0.797±0.012
MNIST	MLP	(100, 50, 10, 2)	0.758±0.226	0.618±0.046	0.655±0.268
MNIST	CNN	See Figure 1	0.574±0.051	0.599 ±0.014	0.850±0.051
Cifar	SVM	Poly, C=20	0.981±0.001	0.605±0.022	0.67±0.015
Cifar	MLP	(100, 50, 10, 2)	0.823±0.250	0.589±0.040	0.671±0.190
Cifar	CNN	See Figure 1	0.589±0.026	0.619 ±0.045	0.819±0.068

Table 1: Benchmark of baseline model.

5.1.2 Cross Validation

To determine the best CNN parameters for each data set, we do cross validation and select the best parameter according to valdiation data. Firstly we fix the learning rate and only change epochs. After that Table 2 and Table 3 shows the details of different parameter's performance.

Parameter	Traning accuracy	Validation accuracy
lr=5e-5, decay=1e-7, batchsize=1024, **epochs=5**	0.591±0.035	0.6 ±0.020
lr=5e-5, decay=1e-7, batchsize=1024, **epochs=10**	0.585±0.035	0.6 ±0.025
lr=5e-5, decay=1e-7, batchsize=1024, **epochs=15**	0.582±0.03	0.595 ±0.012
lr=1e-5, decay=1e-7, batchsize=1024, epochs=5	0.591±0.015	0.596 ±0.014
lr=1e-5, decay=1e-7, batchsize=1024, epochs=10	0.568±0.068	0.607 ±0.03
lr=1e-5, decay=1e-7, batchsize=1024, epochs=15	0.534±0.128	0.596 ±0.035

Table 2: CNN model comparison on MNIST.

From the result of those two tables we choose the fifth row's parameter for MNIST while third row's parameter for Cifar.

Parameter	Traning accuracy	Validation accuracy
lr=5e-5, decay=1e-7, batchsize=1024, **epochs=5**	0.596±0.018	0.598±0.012
lr=5e-5, decay=1e-7, batchsize=1024, **epochs=10**	0.5871±0.016	0.601 ±0.025
lr=5e-5, decay=1e-7, batchsize=1024, **epochs=15**	0.591±0.024	0.610 ±0.015
lr=1e-5, decay=1e-7, batchsize=1024, epochs=5	0.563±0.037	0.6045 ±0.016
lr=1e-5, decay=1e-7, batchsize=1024, epochs=10	0.581±0.052	0.598 ±0.0052
lr=1e-5, decay=1e-7, batchsize=1024, epochs=15	0.576±0.068	0.6005 ±0.0185

Table 3: CNN model comparison on Cifar.

5.2 Results without dealing with label noise

After cross validation, we choose the appropriate parameter and **fix them** to train our models for each dataset and then evaluate them on the test data. Table 4 show the result with the accuracy on the test dataset.

Data set	Parameters	Accuracy on test dataset
Mnist	lr=1e-5, decay=1e-7, batchsize=1024, epochs=10	0.856±0.0261
Cifar	lr=5e-5, decay=1e-7, batchsize=1024, epochs=10	0.805±0.0299

Table 4: Results without dealing with label noise.

5.2.1 Estimate Flip Rate

We used the second method to estimate flip rate by ten times. Flip rate estimation is based on a specific CNN model's parameter and data set. The result is illustrated in Table 6. We can see that

Data set	Parameter	$p(S=1\|Y=0)=0.2$	$p(S=0\|Y=1)=0.4$
Mnist	lr=1e-5, decay=1e-7, batch-size=1024, epochs=10	0.180±0.011	0.400±0.014
Cifar	lr=5e-5, decay=1e-7, batch-size=1024, epochs=10	0.175±0.027	0.357±0.051

Table 5: Flip rate estimation.

the estimated flip rate is close with the given flip rate.

5.2.2 Importance Reweighting

We use the given flip rates, which are $\rho_0 = 0.2$ and $\rho_1 = 0.4$ to deal with label noise in this experiment.

We use pre-trained models to calculate $P_{D_\rho}(\hat{Y}_i|X_i)$ in $\beta(X_i, \hat{Y}_i)$, then calculate every $\beta(X_i, \hat{Y}_i)$ by using following formula

$$\beta(X_i, \hat{Y}_i) = \frac{P_{D_\rho}(\hat{Y}_i|X_i) - \rho_{1-\hat{Y}_i}}{(1-\rho_1-\rho_0)P_{D_\rho}(\hat{Y}_i|X_i) + 0.00001}$$

and calculate the average $\beta(X_i, \hat{Y}_i)$ for all pre-trained models. We add 0.00001 to avoid 0 at the bottom. Since the weight should be non-negative, if we get $\beta(X_i, \hat{Y}_i) < 0$, we make $\beta(X_i, \hat{Y}_i) = 0$. We use all average $\beta(X_i, \hat{Y}_i)$ as sample weights to train our new CNN models then see the average accuracy performance of the models in test data.

We can get following results:
The accuracy of Cifar is 0.878±0.0066.
The accuracy of Fashion-MNIST is 0.925±0.0030.

5.2.3 Loss Correction

In order to implement the Loss Correction function, we need to fulfill the matrix T with the given flip rate. $T_{estimate}$ is an average of ten times' estimation. The given and estimated T is showed below:

$$T_{estimate\_on\_mnist} = \begin{bmatrix} 0.820\pm0.011 & 0.180\pm0.011 \\ 0.400\pm0.014 & 0.600\pm0.014 \end{bmatrix} \tag{23}$$

$$T_{estimate\_on\_cifar} = \begin{bmatrix} 0.825\pm0.027 & 0.175\pm0.027 \\ 0.357\pm0.051 & 0.643\pm0.050 \end{bmatrix} \tag{24}$$

$$T_{given} = \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix} \tag{25}$$

T_{given} is added in our loss function and retrained by the training data and evaluated on validation data. Then we select the best model with fine-tuning and feed it with test data as finnal evaluation.
We can get following results:
By using forward method, the accuracy of Cifar is 0.866±0.0066.
By using backward method, the accuracy of Cifar is 0.863±0.0007.
By using forward method, the accuracy of Fashion-MNIST is 0.928±0.003.
By using backward method, the accuracy of Fashion-MNIST is 0.924±0.005.

Data set	Method	Accuracy(Previous)	Accuracy(After)
Mnist	Importance Reweighting	0.856±0.0261	0.925±0.0030
Mnist	Loss Correction(forward)	0.856±0.0261	0.928±0.003
Mnist	Loss Correction(backward)	0.856±0.0261	0.924±0.005
Cifar	Importance Reweighting	0.805±0.0299	0.878±0.0066
Cifar	Loss Correction(forward)	0.805±0.0299	0.866±0.017
Cifar	Loss Correction(backward)	0.805±0.0299	0.863±0.007

Table 6: All results.

5.3 Extensive analysis and discussion of results

It is obvious that both two methods gain around 7% in each data set after the implementation of importance reweighting and loss correction methods. To be more specific, loss correction(forward) perform slightly better than importance reweighting on the MNIST data while loss correction(backward) rank third. Importance reweighting gain the best performance in the Cifar data while loss correction(forward) ranke seconda and backward method is the last.

The results showed that the importance reweighting method and the loss correction method are both quite robust on reconstructed Fashion-MNIST and Cifar datasets.

For the loss correction method, it shows that forward correction often performs better than the backward correction do. It may because of numerical or a drastic change of the loss, which may have a detrimental effect on optimization [2].

5.4 Comparison

Both importance reweighting and loss correction methods modify the original loss function by applying the flip rate. However, importance reweighting change the loss function by multiplying β directly to the original one, that is $\beta(X_i, \hat{Y}_i)l(f(X_i), \hat{Y}_i)$, while the loss correction method change $f(X_i)$ or $\hat{Y}_i$ which are both inside loss function $l(f(X_i), \hat{Y}_i)$. In the former method, each instance would have a particular weight, while loss correction method just use T and T^{-1} for all instances. This may be the reason that importance reweithting method is more robust than the loss correction method when it comes to more complicate data.

5.4.1 Relevant personal reflection.

What we learned. In this assignment, we learned two different methods, which are importance reweighting method and loss correction method, to learn with class dependent label noise, and tested the performance of them on reconstructed Fashion-MNIST and CIFAR datasets. The importance reweighting method add weight to the empirical risk, while the loss correction method directly change the $f(X_i)$ or $\hat{Y}$ inside loss function $l(f(X_i), \hat{Y})$.

What we found. We found that both methods are very effective to deal with label noise, and the importance reweighting method are a little bit more robust than the loss correction approach when it comes to more complicate data set.

6 Conclusions and future work

6.1 Conclusions

In this assignment, we used CNN and categorical cross entropy loss function on reconstructed Cifar and Fashion-MNIST datasets and compared two methods which are importance reweighting and loss correction, and analyze the robustness of these two label noise methods.

Based on the results, importance reweighting gain a better performance than the loss correction method. In another word, former one is more robust when it comes to sophisticated data set. Through the whole experiment we know that dealing with real word noise is indispensable procedure in model training. How to let model be robust and

6.2 Future work

In this assignment, we used only categorical cross entropy loss function, we may use different loss function, such as square loss, logistic loss and hinge loss, to compare the robustness of the importance reweighting method and loss correction method.

We only adjusted learning rate and epoch in this experiment. We may change network structure, batch size, optimizer method in the future.

We may use different classification method to compare importance reweighting and loss correction methods. For example, SVM, logistic regression, ResNet, etc.

Different methods to deal with label noise would be used in the future. Unbiased estimator, cost-sensitive loss and rank pruning are both good alternatives.

References

[1] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2016.

[2] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, pages 2233–2241, 2017.

[3] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[4] Tom Bylander. Learning noisy linear threshold functions. *Submitted for journal publication*, 1998.

[5] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[6] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. *arXiv preprint arXiv:1711.07131*, 2017.

[7] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.

Appendix

Code environment

Operation system: Ubuntu 18.0.1
CPU: i7-8700k
GPU: 1080 Ti
RAM: 16GB
SSD: 256GB

Python 3.6.4
Numpy 1.15.1
Tensorflow 1.10
scikit-learn 0.19.1

How to run the code

Place data set files and code files in the same directory, use following commands to run the code in terminal
jupyter notebook

Contribution of group members

The code was written in team work as each person in charge of particular parts of code, also as the work of data collection and analysis. After all coding work finished, we wrote the report together through using Overleaf. All in a word, each group member did the same contribution.