Deep learning based Image Classification in the Presence of Label Noise

Ziheng Zhang
Student ID: 8386657
School of Information Science and Technology
ShanghaiTech University, Shanghai, China
zhangzh@shanghaitech.edu.cn

Abstract

We consider the problem of training a deep learning based classifier in the presence of noise. Existing works have tried to solve this problem either in implicit ways such as modifying the deep model to make it more robust to label noise, or estimating noise model, using bootstrapping methods to clean the noisy dataset as well as using robust loss. The former methods are often motivated by intuitive thoughts, which are hard to explain and can only claim their effectiveness by empirical results. The later methods, in contract, often inspired by some statistical considerations, hence have better interpretability. However, most explicit methods are scarcely aware of the properties of deep learning based classifier and made inappropriate assumptions about it. In this work, we aim to investigate interpretable methods to solve the problem not only using statistics, but also taking into account the properties of deep learning based classifier. Technically, we make the following contributions: 1) We study the behavior of deep learning based classifier when noise samples exist in training and testing; 2) We formulate the problem with simple statistics; 3) We propose a dual training procedure to decrease the effects of noise.

In-Class and Out-of-Class Noise

While most related works often divide noise into three categories according to dependency considerations, we only use two categories: 1) the true class of noise samples are within the dataset classes, which we called in-class noise and 2) the true class of noise samples are beyond the dataset classes, which we called out-of-class noise. The two kinds of noise effect the training of deep classifier in different ways.

The in-class noise is the most harmful noise to deep classifier. When we train a deep classifier, we are searching for the best θ to maximize the log-likelihood function $\sum_i p(y_i|x_i;\theta)$. However, this target does not necessarily lead to a generalizable solution - a trivial solution which simply maps all x_i in the training set to its label y_i will also maximize the log-likelihood function. In fact, deep classifiers have capabilities to learn such mapping perfectly, and some experiments even show that deep classifiers can easily fit random labels (Zhang et al. 2016). Hence, generalization ability of deep classifiers suggests that instead of learning mapping from input to output, deep classifiers implicitly learn the distribution of images of their belonging classes. However, in-class noise will badly interfere this process and strongly hurt the generalization ability of deep classifiers. According to the experiments in (Rolnick et al. 2017), given the same noise level, in-class noise leads to worse classification accuracy than out-of-class noise.

The out-of-class noise is somehow less harmful. Intuitively, it will make deep classifier to learn larger and perhaps more complex distribution space for images belonging to each class, hence cause classification problems more difficult. Experiments in (Rolnick et al. 2017) show that it does hurt the final accuracy, but not much as in-class noise.

On the other hand, both kinds of noise introduce extra loss terms that are not good for optimization. When noise level is high, noise terms will dominate the overall loss, and make gradient fairly noisy, which consequently cause gradient based optimization algorithm hard to converge to a good optimal.

Image Classification with CNN Classifier

In image classification setting, we have access to a set of N labeled training images. Denote each training image by $x_i \in \mathbf{R}^d$ and class label by $y_i \in \mathcal{C} = \{0\,, 1\,, 2\,, \ldots\,, C-1\}$, where C is the number of classes. The training set $\mathcal{T} = \{(x_1, y_1)\,, (x_2, y_2)\,, \ldots\,, (x_N, y_N)\}$ is the set of all training image-label pairs. Suppose that training set contains noise, we denote the noise subset by $\mathcal{T}_\mathcal{N}$ and the remaining clean subset by $\mathcal{T}_\mathcal{C}$, that is, we have $\mathcal{T}_\mathcal{C} \cup \mathcal{T}_\mathcal{N} = \mathcal{T}, \mathcal{T}_\mathcal{C} \cap \mathcal{T}_\mathcal{N} = \emptyset$. Further, as we mentioned before, we divide noise into two categories: the in-class noise $\mathcal{T}_{\mathcal{N}in}$ and the out-of-class noise $\mathcal{T}_{\mathcal{N}out}$, that is, $\mathcal{T}_{\mathcal{N}in} \cup \mathcal{T}_{\mathcal{N}out} = \mathcal{T}_\mathcal{N}, \mathcal{T}_{\mathcal{N}in} \cap \mathcal{T}_{\mathcal{N}out} = \emptyset$.

We want to estimate the conditional distribution $p(y \mid x)$, which represents the probability of given image x belonging to class $y \in \mathcal{C}$. In our settings, we use convolutional neural network (CNN) classifier $f(x;\theta)$ as the probabilistic model, where θ represents all parameters in the classifier. We will also use $p(y \mid x;\theta)$ to represent the estimated conditional distribution with $f(x;\theta)$ for convenience.

In order to estimate θ in the classifier, we try to find θ that maximizing log-likelihood function

$$\theta = \operatorname*{argmax}_\theta \mathcal{L}(\theta) = \operatorname*{argmax}_\theta \sum_{i=1}^{N} \log p(y_i \mid x_i;\theta)$$

when $\mathcal{T}_{\mathcal{N}} \neq \emptyset$, i.e., there exists noise in the training set, we can divide the above log-likelihood function into two parts

$$\begin{aligned}\theta &= \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}_{\mathcal{C}}(\theta) + \mathcal{L}_{\mathcal{N}}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{C}}}}^{N} \log p(y_i \mid x_i;\theta) + \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}}}}^{N} \log p(y_i \mid x_i;\theta)\end{aligned} \tag{1}$$

When we try to optimize Eq. (1) with stochastic gradient decent (SGD) based algorithm, the first part $\mathcal{L}_{\mathcal{C}}$ will contribute the correct gradient, while the second part $\mathcal{L}_{\mathcal{N}}$ will also influence the optimization procedure by adding wrong direction to the overall gradient. If noise is relatively high, the second part will dominate the overall gradient, consequently leads to sub-optimal solution.

In order to eliminate the influence of $\mathcal{L}_{\mathcal{N}}$, we consider three cases: 1) $\mathcal{T}_{\mathcal{N}} = \mathcal{T}_{\mathcal{N}out}$, i.e., training set only contains out-of-class noise; 2) $\mathcal{T}_{\mathcal{N}} = \mathcal{T}_{\mathcal{N}in}$, i.e., training set only contains in-class noise and 3) both in-class and out-of-class noise exist in the training set.

For case one, we introduce a set of weights $W = \{w_1, w_2, \ldots, w_N\}$ for each (x_i, y_i) pair that represent how much we believe the label is clean, which leads to the weighted version of problem (1)

$$\begin{aligned}\theta &= \underset{\theta}{\operatorname{argmax}} \mathcal{L}_{\mathcal{C}}^{W}(\theta) + \mathcal{L}_{\mathcal{N}}^{W}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{C}}}}^{N} w_i \log p(y_i \mid x_i;\theta) + \\ &\qquad \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}}}}^{N} w_i \log p(y_i \mid x_i;\theta)\end{aligned} \tag{2}$$

In problem (2), if all weights in the noise part $\mathcal{L}_{\mathcal{N}}^{W}$ equal to zero and those in the clean part $\mathcal{L}_{\mathcal{C}}^{W}$ equal to one, then $\mathcal{L}_{\mathcal{N}}^{W}$ will have no effect on the overall loss, and we will get the optimal θ by optimizing

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}_{\mathcal{C}}^{W^*}(\theta) + \mathcal{L}_{\mathcal{N}}^{W^*}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}_{\mathcal{C}}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{C}}}}^{N} \log p(y_i \mid x_i;\theta)\end{aligned}$$

The procedure of estimating weights for each training pair, which we called dataset cleaning procedure, aims to reduce the influence of $\mathcal{L}_{\mathcal{N}}$ by simply remove the noise term from our optimization target, and when we are not sure about the noise term, softly remove it with smaller weight.

For out-of-class noise pairs, removing them is the best thing we can do. However, in case two, where there are only in-class noise pairs, if we can further estimate their true labels, then we will have more clean training pairs and may achieve better results than simply remove them. Hence, we also introduce dataset relabeling procedure, which aims to correct the original false labels to the true ones for in-class noise pairs. Concrete, we use a set of weights $V = \{v_1, v_2, ..., v_N\}$ to represent how sure we believe a given image x_i belong to a new label y_i' other than y_i. Thus, we will get another weighted version of (1)

$$\begin{aligned}\theta &= \underset{\theta}{\operatorname{argmax}} \mathcal{L}_{\mathcal{C}}(\theta) + \mathcal{L}_{\mathcal{N}}^{V}(\theta) + \mathcal{L}_{\mathcal{N}}^{V'}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{C}}}}^{N} \log p(y_i \mid x_i;\theta) + \\ &\qquad \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}}}}^{N} (1 - v_i) \log p(y_i \mid x_i;\theta) + \\ &\qquad \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}}}}^{N} v_i \log p(y_i' \mid x_i;\theta)\end{aligned} \tag{3}$$

If we correctly relabel all noise pairs with $v_i = 1$, then (3) will lead to the optimal θ.

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}_{\mathcal{C}}(\theta) + \mathcal{L}_{\mathcal{N}}^{V^*}(\theta) + \mathcal{L}_{\mathcal{N}}^{V'^*}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}_{\mathcal{C}}(\theta) + \mathcal{L}_{\mathcal{N}}^{V'^*}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{C}}}}^{N} \log p(y_i \mid x_i;\theta) + \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}}}}^{N} \log p(y_i' \mid x_i;\theta)\end{aligned}$$

For case three, if we can distinguish in-class noise and out-of-class noise, then we can apply training set cleaning procedure and training set relabeling procedure to this two kinds of noise respectively, which leads to the mixture weighted version of (1)

$$\begin{aligned}\theta &= \underset{\theta}{\operatorname{argmax}} \mathcal{L}_{\mathcal{C}}^{W}(\theta) + \mathcal{L}_{\mathcal{N}\,out}^{W} + \mathcal{L}_{\mathcal{N}in}^{V}(\theta) + \mathcal{L}_{\mathcal{N}\,in}^{V'}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{C}}}}^{N} w_i \log p(y_i \mid x_i;\theta) + \\ &\qquad \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}_{out}}}}^{N} w_i \log p(y_i \mid x_i;\theta) + \\ &\qquad \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}_{in}}}}^{N} (1 - v_i) \log p(y_i \mid x_i;\theta) + \\ &\qquad \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}_{in}}}}^{N} v_i \log p(y_i' \mid x_i;\theta)\end{aligned} \tag{4}$$

Similarly, the optimal θ can be found when best estimation of w_i and v_i as mentioned in case one and two are applied to (4)

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmax}} \mathcal{L}_{\mathcal{C}}^{W^*}(\theta) + \mathcal{L}_{\mathcal{N}}^{W^*}{}_{out} + \mathcal{L}_{\mathcal{N}}^{V^*}{}_{in}(\theta) + \mathcal{L}_{\mathcal{N}}^{V^{*\prime}}{}_{in}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \mathcal{L}_{\mathcal{C}}(\theta) + \mathcal{L}_{\mathcal{N}}^{V^{*\prime}}{}_{in}(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{C}}}}^{N} \log p(y_i \mid x_i;\theta) + \sum_{\substack{i=1 \\ (x_i,y_i)\in\mathcal{T}_{\mathcal{N}in}}}^{N} \log p(y_i' \mid x_i;\theta)\end{aligned}$$

Noise Estimation with CNN Classifier

Then the problem comes down to how we can estimate training set noise. Concretely, we want to known three things for each training pair: 1) what's the probability that the training pair is a noise pair; 2) if it is a noise pair, which category of the noise, in-class or out-of-class noise, it belongs to and 3) if it is an in-class noise pair, what's the possible true label and what's the probability that estimated true label is right. After that, we can get θ for CNN classifier by solving optimization problem (2), (3) or (4).

The only clues we can rely on are training pairs $\mathcal{T} = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ and a CNN classifier $f(x\,, \theta)$. Hence, a natural idea is to use CNN classifier trained on the noisy training set to estimate noise in the training set. Since from the very beginning, we intend to use CNN classifier to approximate the conditional distribution $p(y \mid x)$, and if our estimation is fairly accurate, the CNN classifier should give us all the informations we need to estimate noise. Even better, experiments in (Rolnick et al. 2017) have shown that CNN classifier still attains good performance even with an essentially arbitrary amount of noise contained in the training set.

However, given the promising properties of CNN classifier, there are still problems

1. CNN classifier is quite powerful, which makes it easily overfit the training set. As we mentioned before, optimization problem (1) does not necessarily ensure generalization ability of CNN classifier. In fact, according to our experiments, the CNN classifier can easily achieve high accuracy on very noisy training set, and we can hardly see the difference between the fitted noise pairs and cleans ones merely from the output of CNN classifier.

2. Though CNN classifier output valid probability values, these values seem fail to reflect real class probabilities for given images. Our experiments have shown that CNN classifier always prefers high probability values whether it predicts the true label or the false one for a given image. In other words, we can not know the probability that a given image belong to a class merely from the output of CNN classifier.

3. CNN classifier will learn the distribution of labels, hence it's hard to distinguish in-class and out-of-class noise. CNN classifier is known to be sensitive to dataset bias, and will prefer to predict the most frequent class appeared in the training set. Actually, in one experiments, we trained a 1000-class CNN classifier on the training set which only contains 10 classes, expecting that the out-of-class image will be predicted as some other classes other than the supervised 10 classes. However, we found that most out-of-class images will still be predicted as one class out of ten. We can not distinguish in-class and out-of-class images merely from the output of CNN classifier.

Many existing works, such as (Reed et al. 2014; Sukhbaatar et al. 2014; Goldberger and Ben-Reuven 2016; Jindal, Nokleby, and Chen 2016), have ignored these specific problems of CNN classifier and just see it as normal probabilistic models. Here, we address these problems and explore ways to solve them.

Consider the CNN classifier trained on a noisy training set $\mathcal{T}$ consisting of the same number of training pairs belonging to C classes, which we denote by $f(x; \theta^{\mathcal{T}})$. We have already known that $f(x; \theta^{\mathcal{T}})$ have good classification performance while at the same time overfit some noise training pairs in $\mathcal{T}$. Assume we know the classification accuracy of the classifier to be p_c, and use this classifier to predict for another set $\mathcal{E}$ that is mutually exclusive with $\mathcal{T}$ and also contains image-label pairs of the same C classes. Then, we can infer probability that the predicted class $\hat{y}_i$ for $(x_i\,, y_i) \in \mathcal{E}_{\mathcal{C}}$ or $(x_i\,, y_i) \in \mathcal{E}_{\mathcal{N}}$ is the same as y_i

$$p\left(f(x_i; \theta^{\mathcal{T}}) = \hat{y}_i = y_i \mid (x_i\,, y_i) \in \mathcal{E}_{\mathcal{C}}\right) = p_c \tag{5}$$

$$p\left(f(x_i; \theta^{\mathcal{T}}) = \hat{y}_i = y_i \mid (x_i\,, y_i) \in \mathcal{E}_{\mathcal{N}}\right) = p_s \leq 1/C \tag{6}$$

respectively. Probability (5) is easy to get, hence we only explain (6) here.

First, let's consider the out-of-class noise part $\mathcal{E}_{\mathcal{N}out} \subseteq \mathcal{E}_{\mathcal{N}}$. As we mentioned before, the CNN classifier learn the class distribution of training set $\mathcal{T}$. Therefore, even though an image x doesn't belong to any class in the C classes, the classifier will also randomly predict one for it. Because $\mathcal{T}$ and $\mathcal{E}$ are mutually exclusive and the number of training pairs for each class in $\mathcal{T}$ is equal, the classifier can't acquire any information from x, thus can only pickup one class in the C classes with equal probability. Hence, the probability that the predicted class equals to the label of the image is $1/C$.

Now consider the in-class noise part $\mathcal{E}_{\mathcal{N}in} \subseteq \mathcal{E}_{\mathcal{N}}$. We have known that the classifier have good performance, which means its classification accuracy is much better than random guesses, that is, $p_c > 1/C$. Given the truth that labeled class of images in $\mathcal{E}_{\mathcal{N}in}$ can not be their true classes (otherwise, they are not noise pairs anymore), the probability that the predicted label for a image $x \in \mathcal{E}_{\mathcal{N}in}$ equals to its true class is certainly less than $1/C$.

Note that if we use classifier $f(x; \theta^{\mathcal{T}})$ to predict classes for images in $\mathcal{T}$, then (5) and (6) do not necessarily hold due to overfit property of CNN classifier.

Assume the proportion of noise pairs in $\mathcal{E}$ is known to be p_n, with Bayes Formula, we can deduce that

$$p\left((x_i, y_i) \in \mathcal{E}_{\mathcal{C}} \mid \hat{y}_i = y_i\right) = \frac{1 - p_n}{1 - p_n + \frac{p_s}{p_c} p_n} \tag{7}$$

$$p\left((x_i, y_i) \in \mathcal{E}_{\mathcal{N}} \mid \hat{y}_i \neq y_i\right) = \frac{p_n}{p_n + \frac{1-p_c}{1-p_s}(1 - p_n)} \tag{8}$$

Eq. (7) and (8) enable us to estimate the probability that a pair (x_i, y_i) is a clean or noise pair by observing whether its labeled class is the same as its predicted class by classifier $f(x_i; \theta^{\mathcal{T}})$.

To better understand how well we can estimate noise with (7) and (8), let's first examine (7). Technically, (7) tells us how sure we can decide a clean pair. From (7), we can see that whenever $p_s/p_c > 1$, (7) will give us better estimation of clean pairs than random guesses, and the better the classifier, the better estimation can be. In particular, when number of classes C is large, noise level p_n is not extremely large and CNN classifier attains good performance, p_c will be far greater than p_s and p_s/p_c will be very closed to zero, which makes (7) approach to one. Actually, in most cases, C is quite large (e.g., $C = 1000$ for ImageNet), and the accuracy of CNN classifier is much greater than random guesses. Hence, we should hardly make mistakes if we use (7) to choose clean pairs for us.

Eq. (8), on the other hand, tells us how sure we can decide a noise pair. Similarly, whenever $(1-p_c)/(1-p_s)$ is less than one, (8) should provide us better estimation of noise pairs than random guesses. However, unlike (7), (8) is much more sensitive to the accuracy of classifier p_c and when noise level p_n is small while p_c is not very large, we will make a lot of mistakes to decide a noise pair with (8).

The Dual Training Procedure

In this section, we introduce the dual training procedure, which enables us to train a CNN classifier on noisy training set. Assume we have a noisy training set $\mathcal{T}$ and a clean validation set $\mathcal{E}$. In general, the dual training procedure is an iterative procedure, which maintains two subsets $\mathcal{T}_1, \mathcal{T}_2$ of training set $\mathcal{T}$ and two CNN classifiers $f(x; \theta^{\mathcal{T}_1}), f(x; \theta^{\mathcal{T}_2})$ trained on each subset. During each iteration, two subsets will grasp the most possible clean training pairs from each other according to some criterion, and the two CNN classifiers will be fine-tuned on the new subsets respectively. Additionally, we will estimate weights using (7) and (8) for every training pairs in $\mathcal{T}$, and train CNN classifier by solving problem (2) or (4).

Initialization

At the beginning, we randomly divide training set $\mathcal{T}$ into two mutually exclusive parts to initialize $\mathcal{T}_1^0$ and $\mathcal{T}_2^0$, arbitrarily use $W^0 = \mathbf{1}$. The initial noise level of $\left(\mathcal{T} \backslash \mathcal{T}_1^0\right)$ and $\left(\mathcal{T} \backslash \mathcal{T}_2^0\right)$, i.e.$\mathcal{T}_2^0$ and $\mathcal{T}_1^0$, should be the same as that of $\mathcal{T}$, i.e., $p_{n1}^0 = p_{n2}^0 = p_n$.

Iteration

During each iteration, we will follow the steps below

1. Update parameters $\theta^{\mathcal{T}_1}$ and $\theta^{\mathcal{T}_2}$ for the two classifiers by training them on the subset $\mathcal{T}_1^i, \mathcal{T}_2^i$ with weights W^i. We will use stochastic gradient decent to solve the optimization problem (2). Note that, for now, we do not distinguish between in-class and out-of-class noise, and only consider to reduce the effects of noise training pairs by solving the optimization problem (2). Later, we will discuss the possible ways to relabel the in-class noise and estimate weight V that needed by (4).
2. Update the classification accuracy p_{c1}^i and p_{c2}^i for $f(x; \theta^{\mathcal{T}_1^i})$ and $f(x; \theta^{\mathcal{T}_2^i})$ with the clean validation set $\mathcal{E}$.
3. Update subset $\mathcal{T}_1^i$ and $\mathcal{T}_2^i$. We use $f(x; \theta^{\mathcal{T}_1^i})$ and $f(x; \theta^{\mathcal{T}_2^i})$ to predict classes of images in subset $(\mathcal{T} \backslash \mathcal{T}_1^i)$ and $(\mathcal{T} \backslash \mathcal{T}_2^i)$, respectively. Then we add pairs whose predicted classes are the same as labeled classes to the subset. That is
$$\mathcal{T}_1^{i+1} = \mathcal{T}_1 \cup \left\{(x, y) \mid (x, y) \in \mathcal{T} \backslash \mathcal{T}_1^i, f(x; \theta^{\mathcal{T}_1^i}) = y\right\}$$
$$\mathcal{T}_2^{i+1} = \mathcal{T}_2 \cup \left\{(x, y) \mid (x, y) \in \mathcal{T} \backslash \mathcal{T}_2^i, f(x; \theta^{\mathcal{T}_2^i}) = y\right\}$$
4. Update noise level p_{n1} and p_{n2} for subset $\left(\mathcal{T} \backslash \mathcal{T}_1^{i+1}\right)$ and $\left(\mathcal{T} \backslash \mathcal{T}_1^{i+1}\right)$. Given $p_{c1}^i, p_{c2}^i, p_{n1}^i$ and p_{n2}^i, we can use (8) to update noise level for subset $\left(\mathcal{T} \backslash \mathcal{T}_1^{i+1}\right)$ and $\left(\mathcal{T} \backslash \mathcal{T}_1^{i+1}\right)$
$$p_{n1}^{i+1} = \frac{p_{n1}}{p_{n1} + \frac{1-p_{c1}}{1-p_s}(1 - p_{n1})}$$
$$p_{n2}^{i+1} = \frac{p_{n2}}{p_{n2} + \frac{1-p_{c2}}{1-p_s}(1 - p_{n2})}$$
Note that we use $1/C$ as the approximation of p_s, which makes the p_{n1}^{i+1} and p_{n2}^{i+1} always be overestimation of true values.
5. Update wights W. For all new training pairs that added to the $\mathcal{T}_1^{i+1}$ and $\mathcal{T}_2^{i+1}$, we assign their new weights w^{i+1} to be p_{c1} and p_{c2} respectively. And for all training pairs in $\left(\mathcal{T} \backslash \mathcal{T}_1^{i+1}\right)$ and $\left(\mathcal{T} \backslash \mathcal{T}_2^{i+1}\right)$, we assign their new weights w^{i+1} to be $(1 - p_{n1}^{i+1})$ and $(1 - p_{n2}^{i+1})$ respectively.

We continue the iteration until classification accuracy of the two classifier on validation set stops growing up. Finally, we can obtain the noise level p_n of $\mathcal{T}$, which should ranges from (9) to (10)

$$p\left(f(x; \theta^{\mathcal{T}_1}) \neq y; (x, y) \in \mathcal{T}_2\right) - (1 - p_{c1}) \tag{9}$$
$$p\left(f(x; \theta^{\mathcal{T}_1}) \neq y; (x, y) \in \mathcal{T}_2\right) \tag{10}$$

Relabeling for in-class noise

So far, we do not distinguish between in-class and out-of-class noise, and merely (softly) remove noise pairs from training set by selecting most possible clean pairs, estimating weights, then solving weighted optimization problem (2).

Generally, we should get better results by estimating true classes for in-class noise pairs, then solving the mixture weighted optimization problem (4). However, it's not easy to recognize in-class noise due to the problems of CNN classifier described in section Noise Estimation with CNN Classifier. Here, we provide a way to recognize in-class noise and to estimate weight V in problem (4).

Our method is quite straight-forward. Optionally, in step 1 of the last iteration, we examine the predicted classes in every iteration for training pairs in $\left(\mathcal{T} \backslash \mathcal{T}_1^+\right)$ and $\left(\mathcal{T} \backslash \mathcal{T}_2^+\right)$,

which should contains all training pairs we believe are noise pairs. We consider a pair (x_i, y_i) an in-class noise pair if at least half of its predicted classes are the same class $y_i{}'$, and assign the weight v_i for our estimated true class $y_i{}'$ to be the average classification accuracy of classifiers that makes the prediction. After that, we can do relabeling as long as dataset cleaning by solving the mixture weighted optimization problem (4).

References

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 248–255. IEEE.

Frénay, B., and Verleysen, M. 2014. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems* 25(5):845–869.

Frénay, B.; Kabán, A.; et al. 2014. A comprehensive introduction to label noise. In *ESANN*.

Goldberger, J., and Ben-Reuven, E. 2016. Training deep neural-networks using a noise adaptation layer.

Jindal, I.; Nokleby, M.; and Chen, X. 2016. Learning deep networks from noisy labels with dropout regularization. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, 967–972. IEEE.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images.

Reed, S.; Lee, H.; Anguelov, D.; Szegedy, C.; Erhan, D.; and Rabinovich, A. 2014. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

Rolnick, D.; Veit, A.; Belongie, S.; and Shavit, N. 2017. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*.

Sukhbaatar, S.; Bruna, J.; Paluri, M.; Bourdev, L.; and Fergus, R. 2014. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhou, B.; Lapedriza, A.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.