A Comprehensive Analysis for Learning with Class-Conditional Multi-label Noise

Tutors: Rafiul Nakib, Daniel Friedrich
Group members:Kane Wang(530098966, qwan2318), Shiwen Xu (520569045, shxu4542), Junjie Guo (500627547, jguo7670)

Abstract

Incorrectly labeled examples within datasets, a ubiquitous problem, significantly impacts model performance. In this study, we aim to investigate and assess the effectiveness of robust techniques in addressing class-conditional noise (CCN) within the context of multiclass classification problems. To validate the robustness of these techniques, we conducted experiments involving three distinct algorithms: an adaptation layer with forward learning, importance reweighting, and sample selection, integrated into a base Convolutional Neural Network (CNN). Experiments were carried out on three widely recognized datasets. Our findings revealed the challenge of enhancing overall performance through these robustness methods. This challenge may be attributed to the inherent objective of these methods, which aim to assign greater weights to ”clean” samples, thereby achieving a degree of resilience against label noise. Yet, it appears impractical to compel CNN to learn noisy conditional probabilities when their natural inclination is to learn the pristine conditional distribution directly. This study underscores the need for further exploration of more robust techniques, particularly when applied to label-noise-resistant algorithms like large neural networks.

1 Introduction

The label-noise problem has been a prominent subject of research, primarily because the presence of incorrectly labeled examples in datasets is virtually unavoidable. This issue can arise from a variety of factors, including personal subjectivity, non-expert labeling, or a lack of sufficiently discriminative information available to the annotators. It remains a crucial and open research question, primarily because studies have demonstrated that label noise exerts a more severe adverse effect on traditional machine learning classification model performance in comparison to attribute noise[29, 18]. The prevailing large neural network models are also more prone to fit falsely labeled noise due to their inherent capacity to learn highly complex functions[1]. In the context of deep learning methods, current label-noise robust models frequently enhance their robustness through various approaches, including the design of Architectures[23, 11] the modification of Loss functions [15, 19], and Sample Selection method [6]. To assess the practicality of robust classifiers in real-world contexts, we choose to implement three algorithms from each category, including Adaptive Layer with Forward Learning [23, 19], Importance Reweighting[15], and Adaptive Sample Selection[6] to analyze and compare the performance in terms of their robustness against Class-Conditional Noise(CCN).

The combination of Adaptive Layer with Forward Learning [23, 19], when appropriate regularization is applied, offers the advantage of enabling the model to learn both its weights and the transition matrix in an end-to-end manner, consequently reducing the computational source. In contrast, importance reweighting[15] draws inspiration from domain adaptation principles, where the noisy data is regarded as the source domain and the unobservable clean data as the target domain. Unlike the first method, importance reweighting demands more computational resources as the designed loss function reweight each training example by assigning smaller weights to those with false labels and greater weights to those with true labels. To address problem of memorization nature of the large neural networks, We also explored sample selection method [6] to make the neural networks less vulnerable to overfitting by selecting true-labeled examples from the noisy training set. Experiments are carried out with the above three algorithms on FashionMNIST05, FashionMNIST06, and CIFAR datasets.

The following section provides a detailed overview of relevant prior research work in the field of label-noise robust classifiers. In Section 3, we provide a comprehensive description of three selected algorithms, addressing their problem formulation, cost function, optimization methods, and the estimation of the transition matrix. Section 4 elaborates on the details of experiment setups, evaluation metrics, experimental outcomes, performance comparisons, and analysis, followed by the key conclusion summarized in section 5. Implementation details and code run instruction is provided in the appendix.

2 Related Work

Numerous variants of noise-label robust classifiers have been proposed. Since all of our methods are based on deep neural networks, we will not explicitly discuss conventional machine learning approaches, such as [17, 4]. Instead, our focus will be on approaches related to deep learning, considering the aspects of architecture and objective function.

2.1 Architecture

When dealing with an unknown noise distribution, a common approach is to introduce an additional adaptation layer into deep neural networks. Incorporating this adaptation layer allows the whole model to predict the correct labels as well as learn the transition matrix in an end-to-end manner.

To address the issue of class-dependent classification noise (CCN), one of the pioneering works by [23] integrated deep networks with a noisy adaptive layer. They introduced a multi-class neural network with an additional linear noise layer and applied a regularizer to the estimated noise distribution. This regularization constraint was shown to encourage diffusion, subsequently promoting the base model to learn the correct labels. Instead of using a linear layer to model the transition matrix, as only under certain strong assumptions can the linear layer be interpreted as such, [11] proposed an alternative approach. They added another softmax layer and used the Expectation-Maximization (EM) algorithm to train the network. It is important to note that the EM algorithm may sometimes get stuck in local optima, and it can be computationally expensive. Other similar approaches have been proposed in [2, 3], further exploring methods to address CCN.

In summary, this modification to the neural network requires a straightforward adjustment and is relatively easy to implement, as it allows for the joint learning of the noise distribution. However, [26] highlights a common drawback of this approach, which is the inability to identify falsely labeled examples and the practice of treating all examples equally. This limitation can lead to inaccurate estimation of the transition matrix, especially when only noisy data is available or when the noise rate is high.

2.2 Objective Function

Certain methods address the problem by modifying the objective function. This can be achieved through the addition of implicit or explicit regularization terms in the objective function. Alternatively, the loss function can be improved using techniques such as loss correction or loss reweighting.

Adding an extra regularization term to the objective function aims to allow the model to achieve better generalization by avoiding or alleviating overfitting noisy labels [12]. [25] suggests that conventional regularization methods, such as data augmentation[20], weight decay[14], dropout[22], etc., tend to perform well when the noise in the data is not too heavy. Some methods [25, 13] add an additional regularization term to the objective function explicitly. However, explicitly adding an extra regularization term introduces an additional hyperparameter to the model and therefore needs to be fine-tuned to get the optimal result. Other implicit regularization method includes label smoothing [16], or mini-batch training [28]. Compared to explicit regularization terms, it does not introduce any new hyperparameters as it is applied directly to the training data, whereas the extended features or label space may slow down the training process [21].

Apart from regularization, methods modify the loss function through techniques like loss correction [19] or loss reweighting [15]. Unlike the adaptive layer method that directly maps the difference from noisy data to clean data using the estimated transition matrix, this approach enables a more extensive exploration of the training data by adjusting the loss for each individual sample. However, [21] suggests that this approach can lead to the accumulation of false corrections, especially when dealing with a large number of classes or a substantial number of mislabeled instances.

We have explored the Adaptation Layer method with forward learning, and importance reweighting as well as the sample selection methods in this study. Detailed explanations of these methods along with experiment results are provided below.

3 Methods

3.1 Base CNN Model

We built a simple convolutional neural network(CNN) model from sketch as the baseline method, because CNNs have remarkable performance in image processing tasks.

CNNs are also known to be robust against label noise on extensive datasets. According to [9], CNNs are especially more robust against class-dependant label noise, which is discussed in section 4.4.

For a fair comparison, all label noise robustness methods are implemented based on this same CNN base model.

3.2 Robustness Method 1: Adaptation Layer + Forward Learning(bottom-up learning)

We implemented the Adaptation Layer from [23] and combined it with forward loss correction from[19]. Unlike methods that explicitly utilize the known transition matrix to predict the correct label for each instance, this approach includes another noise layer into the neural network to adapt the network outputs to match the noise label distribution. Specifically, the neural network learns the parameters to predict the label as well as learning the noise distribution jointly. In the following sections, we will provide a more in-depth exploration of the problem formulation and the objective function. We will also demonstrate how this approach learns the noise distribution directly from the noisy data. Please note that all equations presented in this context are from [23].

3.2.1 Problem Formulation

Given that the clean labels Y are unobservable, and assuming the label noise is conditioned on true classes but independent from the feature vectors X, then the distribution of the noise data can be parameterized by a matrix $Q = q_{ji}$, where $q_{ji} := p(\widetilde{y} = j|y^* = i)$. Here, $\widetilde{y}$ indicates the corrupted noise labels from some noisy distribution $\widetilde{D}$, y^* indicates the unobserved true labels from clean distribution D. Q is the probability matrix and is constrained that each element is positive and each column sums to 1. Then the probability of input x being labeled as j in noise distribution $\widetilde{D}$ can be formulated as:

$$p(\widetilde{y} = j|x, \theta) = \sum_i p(\widetilde{y} = j|y^* = i)p(y^* = i|x) = \sum_i q_{ji} p(y^* = i|x, \theta) \tag{1}$$

where $p(y^* = i|x, \theta)$ is the probability output of the base model with estimated parameter θ. Essentially, Q is the weights of the adaptive layer to learn the noise distribution from the noisy data. Therefore, the noise layer is a simple linear layer without bias, where its weights are constrained between 0 to 1 and each column of the weights has to be summed to 1 because $\sum_j q_{ji} = 1$.

3.2.2 Optimisation Problem

Therefore, the learning objective is to learn the weights of the neural network w as well as the noise distribution Q by maximizing the log-likelihood over N samples, which can be formulated as:

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N} logp(\widetilde{y} = \widetilde{y}_n|x_n,\theta) = \frac{1}{N}\sum_{n=1}^{N} log(\sum_i q_{\widetilde{y_n}i}p(y^* = i|x_n,\theta)) \tag{2}$$

According to [19], the above loss is called "forward" correction when it compares the noise label $\widetilde{y}$ to the averaged noisy prediction corrupted by Q (which is the transition matrix T in the original paper).

When only the noise data are available, then the weights of noise layer $\hat{Q}$ can be learned through backpropagation to approximate the Q (true noise distribution). The authors point out that there is no guarantee that $\hat{Q}$ would necessarily converge to true Q. To address this uncertainty, the authors introduced a regularizer term, $tr(\hat{Q})$, to the objective, ensuring that the base model to be less affected by the noise and consequently ensure $\hat{Q}$ to converge to true Q. The optimization method can be derived as the following theorem.

Theorem 1. *In the optimization problem, the only global minimum is $\hat{Q} = Q$ and $C = I$. (where Q, $\hat{Q}$, and C are the probability matrices)*

$$\underset{\widetilde{Q},C}{minimize} \quad tr(\hat{Q}) \quad subject\ to \quad \hat{Q}C = Q, \hat{q}_{ii} > \hat{q}_{ij}, q_{ii} > q_{ij}, for \forall i, j \neq i. \tag{3}$$

Proof. Let Q^* be the global solution. Then

$$tr(Q) = tr(Q^*C) = \sum_i(\sum_j q^*_{ij}c_{ji}) \le \sum_i(\sum_j q^*_{ii}c_{ji}) = \sum_i q^*_{ii}(\sum_j c_{ji}) = \sum_i q^*_{ii} = tr(Q^*) \tag{4}$$

Note that this equation will hold true only when $C = I, Q = Q^*$. □

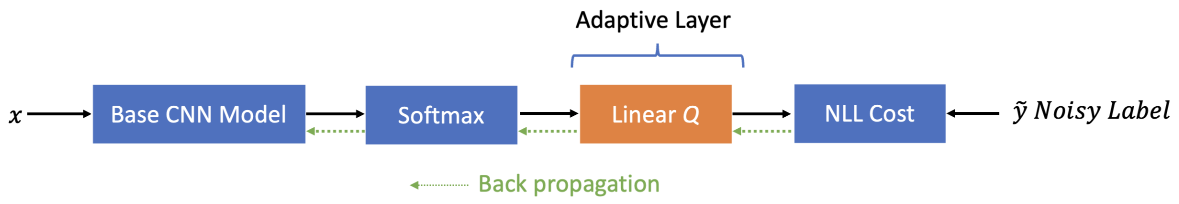


Figure 1: Architecture of base model with the Adaptive Layer

3.3 Robustness Method 2: Importance Reweighting

We implement the importance reweighting based on [15] which integrates importance into the loss function. In specific, this approach obtains robustness to label noise by adding a reweighting mechanism to the existing loss function. It is worth noting that while the original article operates under the assumption of random categorical noise (RCN), the authors also present the applicability of their approach to category-conditional noise (CCN). For the purpose of rigour, in the following sections we will still base on the assumptions of the original article. Please note that all equations presented in this context are from [15].

3.3.1 Problem Formulation

Given the data distribution D of the random variable pair (X, Y), where $X \in \mathbb{R}^m$ and $Y \in \{+1, -1\}$. Samples $(X_1, Y_1), \ldots, (X_n, Y_n)$ are all independently drawn form D, each sample is independent of the others, and all come from the same probability distribution (i.i.d.). Next, considering a asymmetric RCN model [17], we get the new distribution D_ρ which donate the label noised distribution with samples $(X_1, \hat{Y}_1), \ldots, (X_n, \hat{Y}_n)$. The label noise rate is given by

$$P(\hat{Y}=+1|Y=-1)=\rho_{-1},\; P(\hat{Y}=-1|Y=+1)=\rho_{+1} \tag{5}$$

with $\rho_{+1}, \rho_{-1} \in [0,1)$ and $\rho_{+1}+\rho_{-1}<1$.

Under this content, assume the only the noise samples $(X_1,\hat{Y}_1),\ldots,(X_n,\hat{Y}_n)$ which from D_ρ are available, so that the classifier and noise rate will be learned form the noise samples.

3.3.2 Optimisation Problem

Initially the loss function with the noise data are defined as: For the classification with the label noise data, it still have $P_D(X)=P_{D_\rho}(X)$, therefore the equation of $\beta(X,\hat{Y})$ is:

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

Then the importance reweighting parameter $\beta(X,\hat{Y})$ will be applied on the empirical loss function as following:

$$R_{\beta,D_n}(\hat{f})=\frac{1}{n}\sum_{i=1}^{n}\beta(X_i,\hat{Y}_i)\ell(\hat{f}(X_i),\hat{Y}_i) \tag{7}$$

Eventually the objective function with importance reweighting loss function is defined as:

$$f_n=\arg\min_{f\in F}\hat{R}_{\beta,D_n}(f) \tag{8}$$

The optimization process will be the same as the original model. The only changes need to make is to employ the reweighting parameter in the original loss function. Thus once the noise rate ρ_{+1} and ρ_{-1} are calculated, the beta could be calculated by the theorem:

Theorem 2. *Reweighting the surrogate loss function of the traditional classification problem could used for solving the asymmetric RCN problem. Thus the weight of* $(X,\hat{Y})\sim D_\rho$ *is given by:*

$$\beta(X,\hat{Y})=\frac{P_{D_\rho}(\hat{Y}|X)-\rho_{-\hat{Y}}}{(1-\rho_{+1}-\rho_{-1})P_{D_\rho}(\hat{Y}|X)} \tag{9}$$

Where $\beta(X,\hat{Y})\geq 0$ *for* $P_{D_\rho}(\hat{Y}|X)\neq 0$. *If* $P_{D_\rho}(\hat{Y}|X)=0$ *then set* $\beta(X,\hat{Y})=0$.

Hence, for the data in the presence of noise , Classifier can be learned by minimising the empirical risk of reweighting by the following equation:

$$\hat{f}_n=\arg\min_{f\in F}\frac{1}{n}\sum_{i=1}^{n}\beta(X_i,\hat{Y}_i)\ell(f(X_i),\hat{Y}_i) \tag{10}$$

3.4 Robustness Method 3: Adaptive Sample Selection

The adaptive sample selection strategy relies only on batch statistics of a given mini-batch to provide robustness against label noise. This algorithm does not have any additional hyper-parameters for sample selection, including noise rates and clean labels [6]. This algorithm is motivated by curriculum learning and can be thought of as a way to design an adaptive curriculum [6]. Curriculum learning is a way of training a machine learning model where more difficult aspects of a problem are gradually introduced in such a way that the model is always optimally challenged [10].

3.4.1 Problem Formulation

Consider a K-class problem, with $\mathcal{X}$ as the feature space and $\mathcal{Y} = \{0,1\}_K$ as the label space. We assume all labels are one-hot vectors and denote by e_k the one-hot vector corresponding to class k. Let $S^c = \{(x_i, y_i^c)\}$ be iid samples drawn from a distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$ [6].

we have a training set $S = \{(x_i, y_i)\}$ drawn from a distribution $\mathcal{D}_\eta$, y_i are corrupted labels related to y_i^c through

$$P[y_i = e_{k'} | y_i^c = e_k] = \eta_{kk'} \tag{11}$$

where $\eta_{kk'}$ are noise rates which can be represented as a transition matrix. We want to learn a classifier for the distribution $\mathcal{D}$ but given training data drawn from $\mathcal{D}_\eta$.

3.4.2 Optimization Problem

$$\begin{aligned}
\arg \min_{\sigma, \omega \in [0,1]^m} \mathcal{L}_{wtd}(\theta, \omega) &= \sum_{i=1}^{n} (w_i l_i - \lambda(y_i) w_i) \\
&= \sum_{j=1}^{K} \sum_{i=1, i:y_i=e_i}^{K} (w_i l_i + (1 - w_i)\lambda_j) - \sum_{j=1}^{K} \sum_{i=1, i:y_i=e_i}^{K} \lambda_j
\end{aligned} \tag{12}$$

where the class-label-dependent thresholds e_j equal to epoch loss values $\lambda_j = \lambda(e_j)$ and l_i is the loss of step i.

3.5 Optimization Method

For fair comparison, apart from applying additional constraints to the the adaptive layer in the bottom-up learning method, the base CNN and all three robustness methods use Adam [7] as the optimisation method as shown below.

```
**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0,1)$: Exponential decay rates for the moment estimates
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Require:** $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  **while** $\theta_t$ not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  **end while**
  **return** $\theta_t$ (Resulting parameters)
```

Figure 2: Adam optimisation algorithm

3.6 Noise Rate Estimation

The method we used to estimate the noise rate is a simple extension from the binary method proposed by Jiacheng et al [8]. Let Y be the clean label, and $\tilde{Y}$ be the noisy label, we can obtain 13.

$$\begin{bmatrix} P(\tilde{Y}=0|X) \\ P(\tilde{Y}=1|X) \\ P(\tilde{Y}=2|X) \end{bmatrix} = \begin{bmatrix} P(\tilde{Y}=0|Y=0) & P(\tilde{Y}=0|Y=1) & P(\tilde{Y}=0|Y=2) \\ P(\tilde{Y}=1|Y=0) & P(\tilde{Y}=1|Y=1) & P(\tilde{Y}=1|Y=2) \\ P(\tilde{Y}=2|Y=0) & P(\tilde{Y}=2|Y=1) & P(\tilde{Y}=2|Y=2) \end{bmatrix} \begin{bmatrix} P(Y=0|X) \\ P(Y=1|X) \\ P(Y=2|X) \end{bmatrix} \tag{13}$$

To estimate the first column of the transition matrix, we need to find an anchor point x^0, such that $P(Y=0|X=x^0)=1$, $P(Y=1|X=x^0)=0$ and $P(Y=2|X=x^0)=0$. Then we can obtain the first column of the transition matrix as follows,

$$\begin{bmatrix} P(\tilde{Y}=0|X=x^0) \\ P(\tilde{Y}=1|X=x^0) \\ P(\tilde{Y}=2|X=x^0) \end{bmatrix} = \begin{bmatrix} P(\tilde{Y}=0|Y=0) & P(\tilde{Y}=0|Y=1) & P(\tilde{Y}=0|Y=2) \\ P(\tilde{Y}=1|Y=0) & P(\tilde{Y}=1|Y=1) & P(\tilde{Y}=1|Y=2) \\ P(\tilde{Y}=2|Y=0) & P(\tilde{Y}=2|Y=1) & P(\tilde{Y}=2|Y=2) \end{bmatrix} \begin{bmatrix} P(Y=0|X=x^0) \\ P(Y=1|X=x^0) \\ P(Y=2|X=x^0) \end{bmatrix} \tag{14}$$

$$= \begin{bmatrix} P(\tilde{Y}=0|Y=0) & P(\tilde{Y}=0|Y=1) & P(\tilde{Y}=0|Y=2) \\ P(\tilde{Y}=1|Y=0) & P(\tilde{Y}=1|Y=1) & P(\tilde{Y}=1|Y=2) \\ P(\tilde{Y}=2|Y=0) & P(\tilde{Y}=2|Y=1) & P(\tilde{Y}=2|Y=2) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{15}$$

$$= \begin{bmatrix} P(\tilde{Y}=0|Y=0) \\ P(\tilde{Y}=1|Y=0) \\ P(\tilde{Y}=2|Y=0) \end{bmatrix} \tag{16}$$

Similarly, we can find another two anchor points to estimate the second and third columns of the transition matrix. When the noise rate is upper bounded by a constant, then the anchor points can be found as follows 24,

$$x^i = argmax_{x \in \mathcal{X}} P(\tilde{Y}=i|X=x). \tag{17}$$

4 Experiments

4.1 Evaluation Metrics

4.1.1 Recall

Recall is defined as the ratio of the number of true positive predictions to the total number of actual positive cases which is critical in situations where the consequences of missing a true positive result are severe [24]. It is calculated using the following formula:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{18}$$

4.1.2 Precision

Precision measures the proportion of positive identifications that were actually correct. It ensures that the model's positive predictions are truly relevant, thus minimising the cost of false positives [24].

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{19}$$

4.1.3 F1-score

F1-score is the harmonic mean of precision and recall which is a single metric that combines both precision and recall into one number, providing a balance between the two. It is particularly valuable when the costs of both false alarms and omissions are high and need to be carefully weighed [24].

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

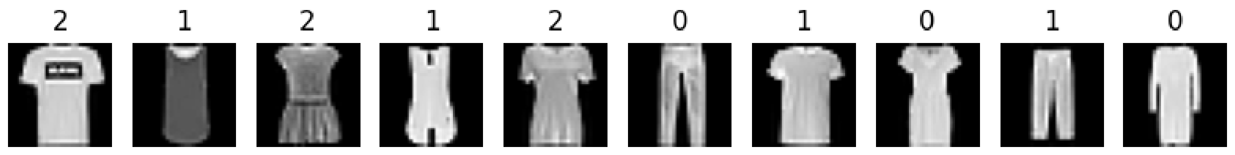
4.1.4 Top 1 Accuracy

Top-1 accuracy is the percentage of times the highest probability prediction exactly matches the true category in a single classification task. For tasks that allow only one attempt at correct classification, top 1 accuracy is a useful metric [24].

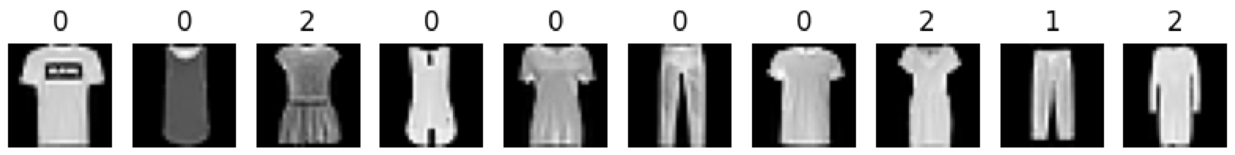
$$\text{Top-1 Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{21}$$

4.2 Datasets

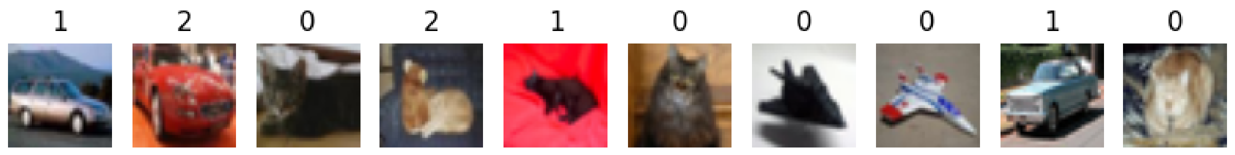
We experiment with the models on three datasets. FashionMNIST05 and FashionMNIST06 are gray-scaled 28 $\times$ 28 image datasets. Both of them have 18000 training examples and 3000 testing examples.CARFI is a RGB 32 $\times$ 32 image dataset, with 15000 training examples and 3000 testing examples.



(a) FashionMNIST05



(b) FashionMNIST06



(c) CIFAR

Figure 3: Noisy training data

All the training sets are contaminated by class-dependent label noise while the testing sets are clean 3. For FashionMNIST05 and FashionMNIST06, the transition matrices are known 22.

$$T_{FashionMNIST05} = \begin{bmatrix} 0.5 & 0.2 & 0.3) \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}, T_{FashionMNIST06} = \begin{bmatrix} 0.4 & 0.3 & 0.3) \\ 0.3 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} \tag{22}$$

We firstly normalised the train data and test data and then split the train data to the training set(90%) and the validation set(10%) during cross validation.

4.3 Results

We train the adaptive sampling selection model 200 epochs and other methods 40 epochs. We repeated the process 10 times and randomly split training set and validation set every time.

	Acc	Std	Precision	Std	Recall	Std	F1	Std
Base Model	0.9364	0.0026	0.9372	0.0027	0.9364	0.0026	0.9366	0.0026
Importance Reweighting	0.9102	0.0022	0.9161	0.0027	0.9102	0.0022	0.9112	0.0021
Bottom Up Learning	0.9368	0.0028	0.9375	0.0030	0.9368	0.0028	0.9370	0.0029
Adaptive Sample Selection	0.8364	0.1718	0.8339	0.1917	0.8364	0.1718	0.8337	0.1825

Table 1: Results on FashionMNIST05

	Acc	Std	Precision	Std	Recall	Std	F1	Std
Base Model	0.8915	0.0031	0.8996	0.0024	0.8915	0.0031	0.8927	0.0031
Importance Reweighting	0.8776	0.0041	0.8880	0.0034	0.8776	0.0041	0.8791	0.0039
Bottom Up Learning	0.8937	0.0048	0.9011	0.0033	0.8937	0.0048	0.8948	0.0047
Adaptive Sample Selection	0.6112	0.2559	0.5761	0.2934	0.6112	0.2559	0.5873	0.2790

Table 2: Results on FashionMNIST06

Both the importance reweighting and the adaptive sample selection seem not as efficient as they claim. While the base model converges after about 15 epochs, the adaptive sample selection may not have converged after 100 epochs. The importance reweighting takes 8 times longer time as the base CNN when they are trained with the same numbers of epochs.

For all the models, their acc, precision, recall and F1 are almost the same as shown in Table 1, 2 and 3.

For better comparison, the epoch numbers are aligned to the same in figure 4, 5, 6, by sample the adaptive sampling selection training history with fixed step. Be aware of that the actual training process of the adaptive sampling selection model is five times long as shown in the figures.

Also note that the error bounds in the figures are 50% shrunk to avoid massive overlaps. The real bounds are twice large as that shown in these figures.

4.3.1 Performance on FashionMNIST05

On FashionMNIST05, none of the robustness methods outperform the base CNN as shown in figure 4. The bottom up learning has very close results to the base CNN. Their accuracy is around 0.93. The importance reweighting is slightly worse with 0.91 accuracy. The adaptive sample selection has the worst performance with 0.83 accuracy.

The base CNN, importance reweighting and the bottom up learning have stable performance during cross-validation, while adaptive sample selection is extremely unstable with large variation.

While the base CNN, bottom-up learning and importance reweighting achieve high accuracy after 1 epoch, the adaptive sample selection starts with a much lower accuracy and grows slowly.

4.3.2 Performance on FashionMNIST06

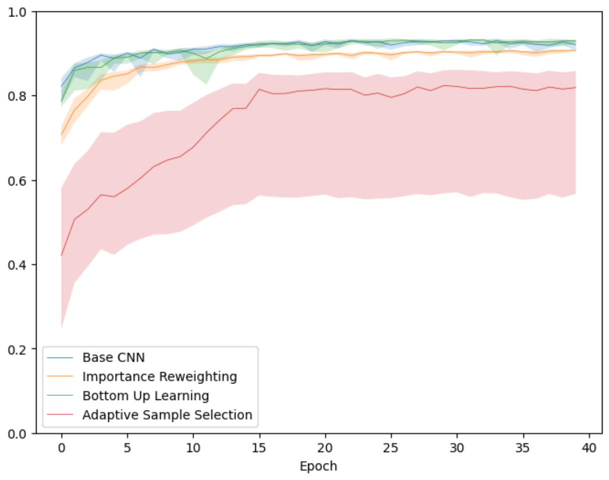
On FashionMNIST06, the performance of the base CNN, bottom up learning and importance reweighting is very close as shown in figure 5. Their accuracy is around 0.89. The importance reweighting is slightly worse with 0.91 accuracy. The adaptive sample selection still has the worst performance with 0.61 accuracy.

Similar to the performace on FashionMNIST05, the base CNN, importance reweighting and the bottom up learning are stable on FashionMNIST06, while adaptive sample selection often performs badly. However, the best performance of all the models are almost the same.

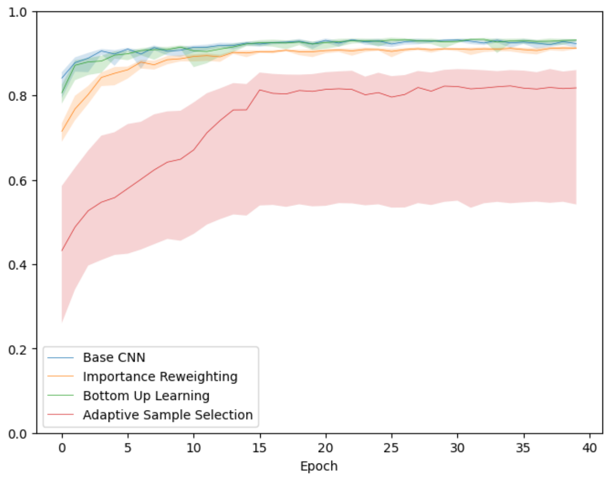
Compared to FashionMNIST05, all methods are less stable on FashionMNIST06.

4.3.3 Performance on CIFAR

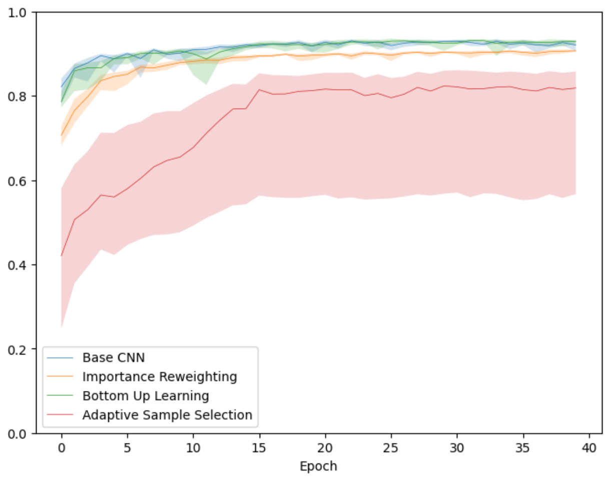
Because the transition matrix of CIFAR is unknown, We trained the base CNN on CIFAR training set, and estimated the noise rates as follow by using the method explained in section 3.5



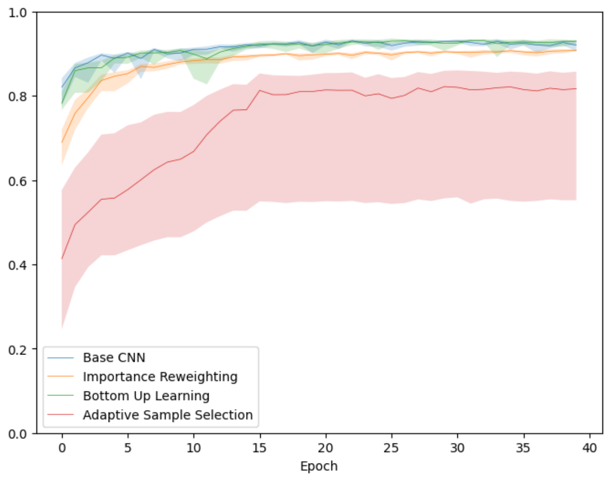
(a) Top 1 Accuracy



(b) Precision



(c) Recall



(d) F1

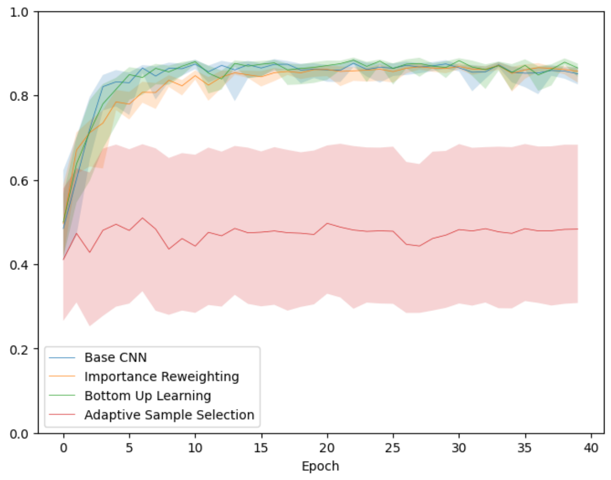
Figure 4: Performance on FashionMNIST05

	Acc	Std	Precision	Std	Recall	Std	F1	Std
Base Model	0.6181	0.0129	0.6265	0.0126	0.6181	0.0129	0.6133	0.0151
Importance Reweighting	0.5759	0.0069	0.5829	0.0063	0.5759	0.0069	0.5702	0.0080
Bottom Up Learning	0.6153	0.0182	0.6264	0.0231	0.6153	0.0182	0.6097	0.0182
Adaptive Sample Selection	0.5200	0.0607	0.5187	0.0664	0.5200	0.0607	0.5083	0.0620

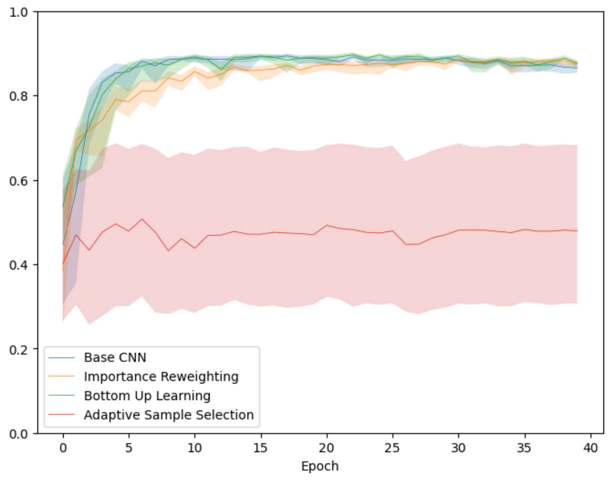
Table 3: Results on CIFAR

$$T_{CIFAR} = \begin{bmatrix} 0.5761 & 0.2119 & 0.2119) \\ 0.2119 & 0.5761 & 0.2119 \\ 0.2119 & 0.2119 & 0.5761 \end{bmatrix} \tag{23}$$

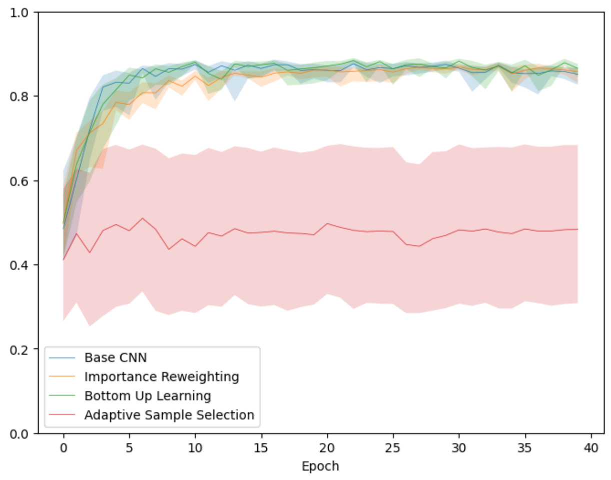
The performance of these methods on CIFAR is much worse than on previous datasets and more unstable. The base CNN and bottom-up learning have the highest accuracy 0.61 as show in Figure 6 and Table 3. Importance reweighting achieves 0.57 accuracy while adaptive sample selection remain the worst with 0.52 accuracy.



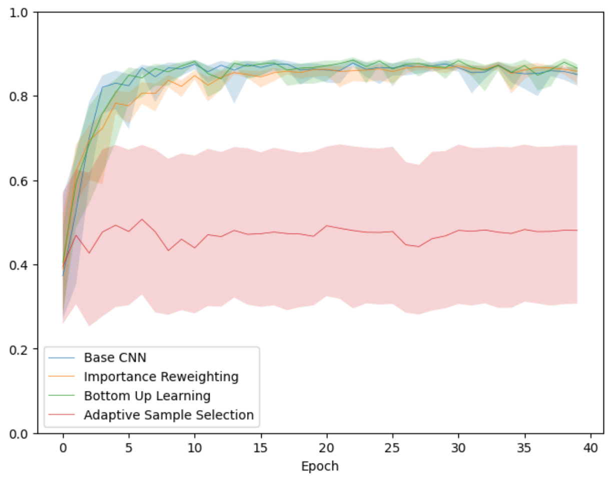
(a) Top 1 Accuracy



(b) Precision



(c) Recall



(d) F1

Figure 5: Performance on FashionMNIST06

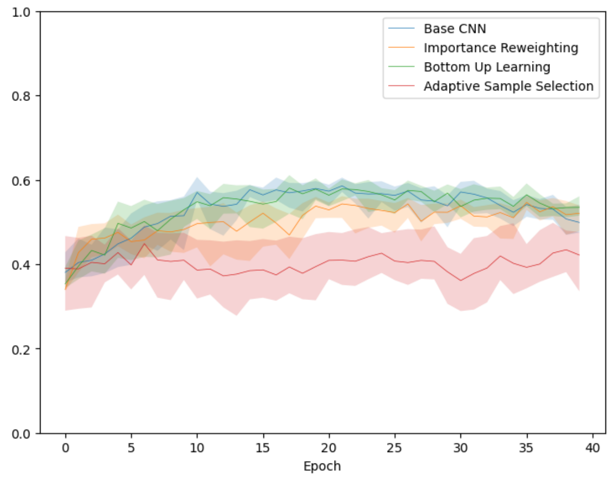
4.4 Discussions

We found that the transition estimation method extended from the binary classification is not rigorous. It seems not naturally easy to extend it to multiple-class classification. As discussed in section 3.6, we find the anchor points as follows

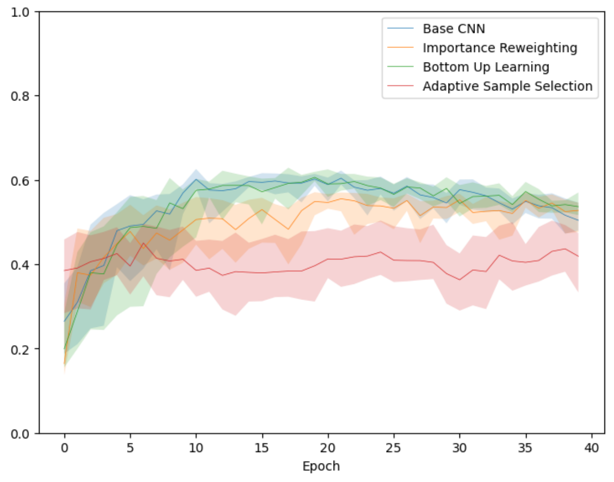
$$x^i = argmax_{x \in \mathcal{X}} P(\tilde{Y} = i | X = x). \tag{24}$$

Taking the first column of the transition matrices for example, for binary classification, $P(\tilde{Y} = 0|Y = 0)$ can be estimated by $P(Y = 0|X = x^0)$. Then we can derive $P(\tilde{Y} = 1|Y = 0)$ by $1 - P(Y = 0|X = x^0)$.

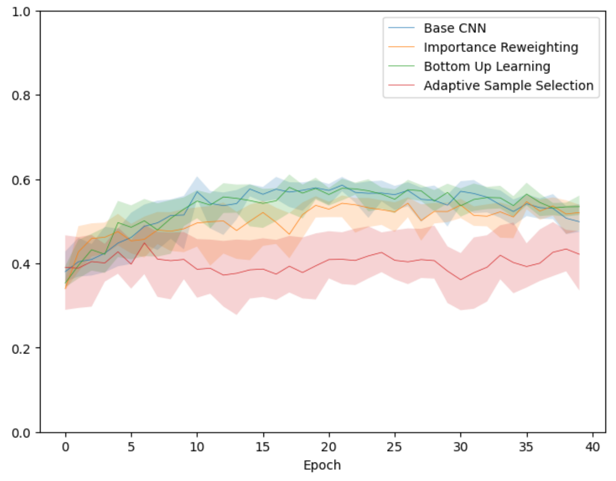
However, for multiple-class classification, we cannot directly estimate $P(\tilde{Y} = 1|Y = 0)$ and $P(\tilde{Y} = 2|Y = 0)$ in the same way. What we can do is estimating $P(\tilde{Y} = 1|Y = 0)$ and $P(\tilde{Y} = 2|Y = 0)$ simply by $P(Y = 1|X = x^0)$ and $P(Y = 2|X = x^0)$, which might not be accurate.



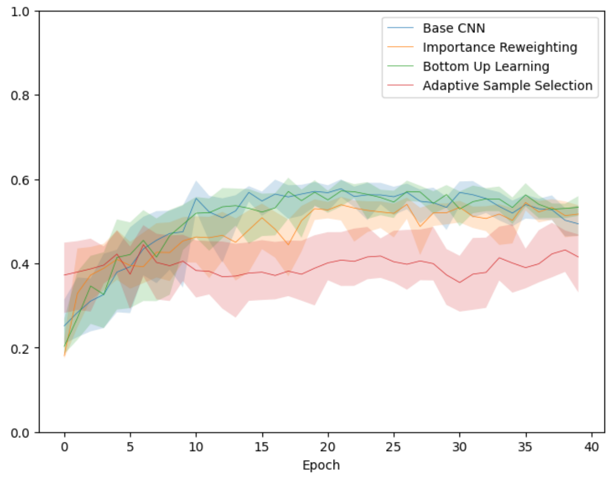
(a) Top 1 Accuracy



(b) Precision



(c) Recall



(d) F1

Figure 6: Performance on CIFAR

Another problem of this method is that, estimated predicted probabilities of produced by the deep neural network's output of a softmax layer can not reliably be used as the true probabilities. In practice, they tend to be too high [5]. In this case, using anchor points to estimate the transition matrix might be biased.

For all the models, their acc, precision, recall and F1 are almost the same as shown in Table 1, 2 and 3. As our scenario is multiple-class classification, it indicates the performance of the models for each class is nearly the same.

Without applying any robustness methods, the base CNN model is surprisingly already very robust to class-dependant label noise. In [9], two potential causes are explained. One is that class-dependant noise label is still a similar class to the original. The network can learn clustering to some extend. Another one is the property of cross entropy loss. When the label of sample x is i, the loss value can be written as -$log[Softmax(x)]_i$. When a CNN predicts x as i with weaker confidence, the penalty gets larger.

The reason that the performance of the bottom up learning model is almost the same as the base model might be that, while the additional adaptation layer is supposed to learn the noise rates to absorb the noise from previous layers, the base CNN model is already robust to label noise. The input fed to the adaptation layer is close to true $P(Y|X)$, which makes it difficult for the additional adaptation layer to learn the noise rates.

Another reason that these roubstness methods failed to achieve better performance might be the difficulty of fine tuning the models. The optimisation hyper-parameters like learning rate, weight decay, momentum and rate schedular vary on different distributions. The hyper-parameters we used in the experiments might not be optimal on our datasets.

Apart from above reasons, challenges associated with the importance reweighting method might be, importance reweighting approach is highly reliant on the accuracy of estimating the conditional distribution $P_{d_\rho}(\hat{Y}|X)$ [15]. In addition, the importance reweighting method can be prone to errors when inappropriate training models are selected, particularly in the context of large neural networks with numerous parameters, as they possess the capacity to memorize all training examples [27].

5 Conclusion

In conclusion, we found that it is difficult to improve CNNs' performance by using these robustness methods. All these methods aim to select (or give more weights to) 'clean' samples for obtaining a degree of robustness against label noise. However, it might be impractical to force the CNNs to learn $P_{d_\rho}(Y|X)$ while they tend to learn $P_d(Y|X)$ directly.

Although the adaptive sample selection does not need any prior noise information, it still significantly relies on optimisation hyper-parameters and is not as efficient as expected.

In the future, as many recent state-of-the-art image process approaches are based on transformers, we can experiment these robustness methods on transformer-based classification approaches, to see if they can improve the performance of transformer approaches or not.

When label noise rates are very high(over 50%), pure supervised approaches might not be suitable any more as the labels are too miss-leading for teaching the model. We can explore other robustness methods utilising unsupervised or weak-supervised methods such as clustering-based robustness methods.

References

[1] Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017.

[2] Alan Joseph Bekker and Jacob Goldberger. Training deep neural-networks based on unreliable labels. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2682–2686. IEEE, 2016.

[3] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1431–1439, 2015.

[4] Aritra Ghosh, Naresh Manwani, and PS Sastry. On the robustness of decision tree learning under label noise. In *Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I 21*, pages 685–697. Springer, 2017.

[5] family=Guo given i=C, given=Chuan, family=Pleiss given i=G, given=Geoff, family=Sun given i=Y, given=Yu, and family=Weinberger given i=KQ, given=Kilian Q. On calibration of modern neural networks. pages 1321–1330.

[6] family=Patel given i=D, given=Deep and family=Sastry given i=PS, given=P. S. Adaptive sample selection for robust learning under label noise.

[7] family=Kingma given i=DP, given=Diederik P. and family=Ba given i=J, given=Jimmy. Adam: A method for stochastic optimization.

[8] family=Cheng given i=J, given=Jiacheng, family=Liu given i=T, given=Tongliang, family=Ramamohanarao given i=K, given=Kotagiri, and family=Tao given i=D, given=Dacheng. Learning with bounded instance- and label-dependent label noise.

[9] family=Hataya given i=R, given=Ryuichiro and family=Nakayama given i=H, given=Hideki. Investigating cnns' learning representation under label noise.

[10] family=Bengio given i=Y, given=Yoshua, family=Louradour given i=J, given=Jérôme, family=Collobert given i=R, given=Ronan, and family=Weston given i=J, given=Jason. Curriculum learning.

[11] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International conference on learning representations*, 2016.

[12] Bo Han, Quanming Yao, Tongliang Liu, Gang Niu, Ivor W Tsang, James T Kwok, and Masashi Sugiyama. A survey of label-noise representation learning: Past, present and future. *arXiv preprint arXiv:2011.04406*, 2020.

[13] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International conference on machine learning*, pages 2712–2721. PMLR, 2019.

[14] Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

[15] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.

[16] Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, pages 6448–6458. PMLR, 2020.

[17] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

[18] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33:275–306, 2010.

[19] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.

[20] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[21] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[22] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[23] Sainbayar Sukhbaatar and Rob Fergus. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, 2(3):4, 2014.

[24] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[25] Ryutaro Tanno, Ardavan Saeedi, Swami Sankaranarayanan, Daniel C Alexander, and Nathan Silberman. Learning from noisy labels by regularized estimation of annotator confusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11244–11253, 2019.

[26] Xiaobo Xia, Bo Han, Nannan Wang, Jiankang Deng, Jiatong Li, Yinian Mao, and Tongliang Liu. Extended t t: Learning with mixed closed-set and open-set noisy labels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3047–3058, 2022.

[27] Xiyu Yu, Tongliang Liu, Mingming Gong, Kayhan Batmanghelich, and Dacheng Tao. An efficient and provable approach for mixture proportion estimation using linear independence assumption. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4489, 2018.

[28] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[29] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22:177–210, 2004.

A Setup Instructions

There is one notebook file in the code directory. The code is tested on google Colab and Apple Macbook M2. To run the code, make sure you have installed Pytorch and Torch Metrics. The code will automatically detect GPU devices and run on them if available.

Change the dataset root directory configuration DATA_ROOT at the beginning if it is necessary. Generally, there is no extra work needed to run the code.