Systematic Iterative Reweighting for Learning from Noisy Labels

Ajay Shridhar Joshi

CS15B047

1 Problem Statement

Developing algorithms to make machine learning models robust to incorrect labels in datasets.

2 Introduction

One of the crucial necessity for latest machine learning models is the availability of large-scale datasets. Labeled large-scale datasets are an important requirement to boost the performance of various tasks in the various fields such as computer vision, speech processing, natural language processing. Though large datasets are carefully annotated and verified by multiple annotators, there is a high probability of noise (wrong annotations) when it comes to large-scale datasets. Training a powerful machine learning model with such noisy dataset will lead to biased performance that depends on the wrong labels and deteriorate the performance. Hence, augmenting the machine learning model with the ability to learn from noisy labeled dataset is of paramount importance.

Also, there are alternative sources of data which provide huge amount of data, for eg: images queried from web-search, but often have noisy labels, like the query keywords or the tags surrounding them. Leveraging such sources for training machine learning models also requires the ability to learn from noisy labels.

3 Background and Related Work

The problem of learning from noisy labels has been approached in different ways in literature, some of which are described below. Some of the early works related to learning from noisy labels like [6] study binary classification and prove certain models to be noise-tolerant under certain conditions.

Techniques like [9] and [3] introduce a linear layer that models the probability of corruption of the class label into another class. [7] introduces a loss correction factor which is dependent on corruption probabilities of a label to the other classes. This probability transition matrix is either already known or is estimated using a small clean dataset for validation. Most of these methods assume that the probability of corruption of an example is only dependent on the true class and

the corrupted class, and independent of the individual training instance. Such techniques work when all examples can be mapped to some class in the dataset. They have some difficulty in adapting to scenarios where the noisy examples do not belong to any of the class in the dataset ("open-set" noise).

Approaches like [5] use an extra teacher network which performs instance selection for the student network, which broadly takes the student network progress as input and returns weights for every example. Another approach [4] uses low loss as an indicator of a clean example and throws away high loss instances progressively. It uses 2 networks which exchange low loss training examples, which helps in reducing error flows. This technique essentially does hard (0-1) weighting of examples by throwing high loss instances away. Recent work [10] introduced methods for tackling "open-set" noise, i.e. when the noisy examples cannot be relabeled correctly to some class in the dataset. They performed iterative learning using outlier detection, feature similarity matching and reweighting. [8] uses meta-learning methods to learn the weights to be assigned to the examples.

Our work does not depend on whether the noise is closed-set or open-set. We do soft weighting of examples and also give a principled way of iteratively reweighting examples.

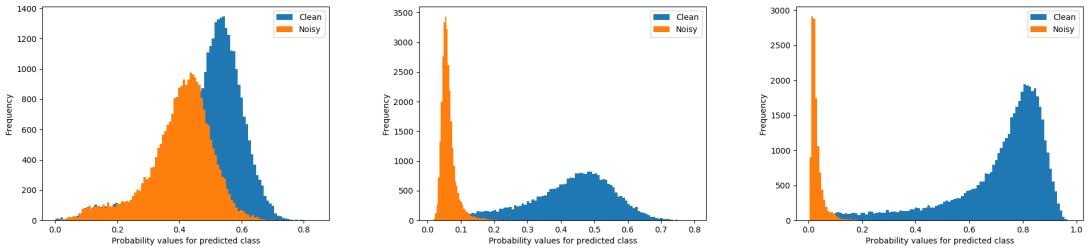
4 Methodology

We approached the problem of learning from noisy labels by trying to separate samples with noisy labels from those with clean labels. We conducted multiple studies, each of which are described below. Observations from each of the studies led to 2 novel algorithms that are discussed at the end of this section.

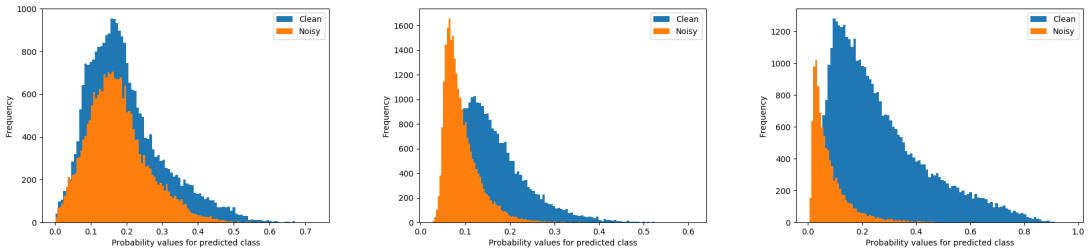
4.1 Characteristics differentiating examples with clean and noisy labels

In this study, we tried to find out the characteristics of clean and noisy labels, which would help to differentiate between them. We hypothesized that probability of an example for its class as predicted by the model could be used as a characteristic to separate between clean and noisy labels. This hypothesis is motivated by [1]. This paper states that deep neural networks(DNNs), which are capable of fitting even random noise, fit simpler patterns present in data in the initial stages of training.

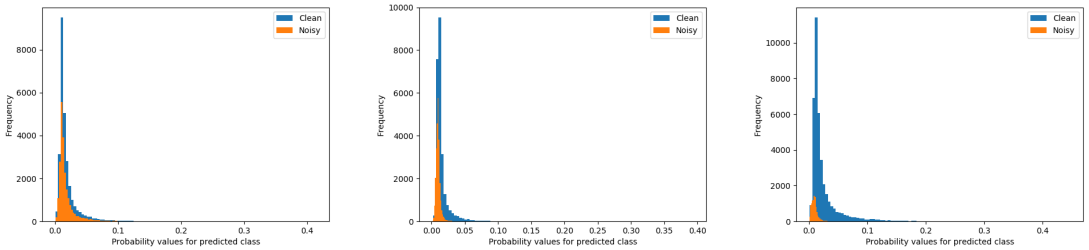
From this observation, we deduced that in the initial stages of training, the DNN will fit the inherent patterns present in the data, which correspond to the clean labels, assuming that the examples with clean labels are in majority. This led us to hypothesize that probability values predicted for clean examples would be higher than those predicted for noisy examples. This was verified through experiments on multiple datasets: MNIST, CIFAR-10, CIFAR-100.(Figure: 1). We made one more important observation from the experiments, that the separation between probability value distributions corresponding to examples with clean and noisy labels increases with reduction in noise levels.



(a) MNIST Dataset



(b) CIFAR-10 Dataset



(c) CIFAR-100 Dataset

Fig. 1: Probability histograms after the first epoch for 3 different dataset with synthetic 45% pair-flip noise, 50% symmetric noise, 20% noise respectively. Pair-flip noise corresponds to examples swapped only into 1 other class and symmetric noise corresponds to examples being swapped into all other classes uniformly.

4.2 Regularization

Model regularization helps limit the complexity of a model by restricting its parameters. This means that a model trained with high regularization will be able to only fit simpler patterns than those that can be fit by a model with low/no regularization. In this problem setting, majority of examples with clean labels correspond to simpler patterns in data, whereas examples with noisy labels correspond to harder patterns in data, along with clean examples which are hard to classify.

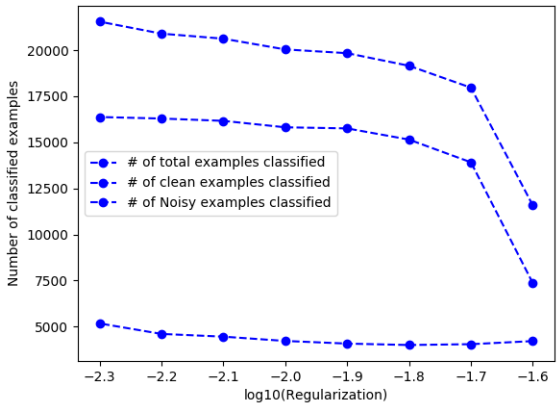


Fig. 2: Plot of number of classified examples by the model on CIFAR-10 dataset with 45% pair-flip noise: total number of correctly classified examples, examples with clean labels, examples with noisy labels

The figure 2 shows how regularization enables some amount of separation of clean examples from noisy ones. At higher values of regularization, the number of clean examples classified is higher than noisy ones, because the model can only fit simpler patterns in data. As regularization decreases in the training process, the model starts to fit noisy data. Thus, the fraction of clean examples classified by the model out of the total classified decreases with decreasing regularization. Algorithm 4.6 is directly based on this observation.

4.3 Reweighting Examples

Training on datasets with noisy labels can achieve good performance if examples with noisy labels are removed from the training set. Section 4.1 states that clean examples can be characterized as having higher class prediction probability from the DNN model than examples with noisy labels. This characterization

motivated the weighting of examples based on their prediction probability values obtained from the model. So, we use a weighted loss function for training:

$$L = \sum_{i=1}^{N} l_i w_i$$

where l_i and w_i correspond to loss and weight for the i^{th} training example and N is size of the training dataset. We tried different functions like linear, quadratic, exponential functions 3 for mapping prediction probability into the weight for an example.

$$calc\_weight(p_{example}, p_{max}, p_{min}) = \frac{p_{example} - p_{min}}{p_{max} - p_{min}}$$

$$calc\_weight(p_{example}, p_{max}, p_{min}) = (\frac{p_{example} - p_{min}}{p_{max} - p_{min}})^2$$

$$calc\_weight(p_{example}, p_{max}, p_{min}) = \frac{exp(\frac{p_{example} - p_{min}}{p_{max} - p_{min}}) - 1}{e - 1}$$

Fig. 3: Linear, Quadratic and Exponential example weighting functions. $p_{example}$ is the probability value for the example, p_{max} and p_{min} are maximum and minimum probability values across the entire training dataset.

Based on the above arguments, it is possible to formulate a training algorithm based on reweighting, independent of the DNN model used. There are some shortcomings associated with this reweighting approach that we observed. First, the predicted probability by the model for an example which is used to determine the weight of an example should be "reasonable", i.e. lower for a noisy and higher for a clean labeled example. Initially, at the start of training, the model predicts randomly and thus the estimates are bad. According to [1], initially the model will learn the simpler patterns in data, thus improving the probability estimate. But, the model will eventually overfit on the noisy data with large number of training steps deteriorating the estimates.

This implies that there is a training stage in between the start and overfitting stage, where the probability estimates are reasonable, and the model has fit simple patterns in data, without overfitting on noisy instances. Thus, this stage is a good one to start our reweighting algorithm. The problem is that this stage depends on difficulty of the dataset, learning rate etc. and thus varies across different datasets and identifying it in every dataset without access to a clean validation set is tough. Section 4.4 tries to present a solution to this problem, using insights from the DNN training process.

4.4 Observations in DNN Training

This section presents a novel idea for finding out a stage in DNN training, where the model has learnt patterns in data corresponding to the clean examples, but has not yet started overfitting on noisy examples.

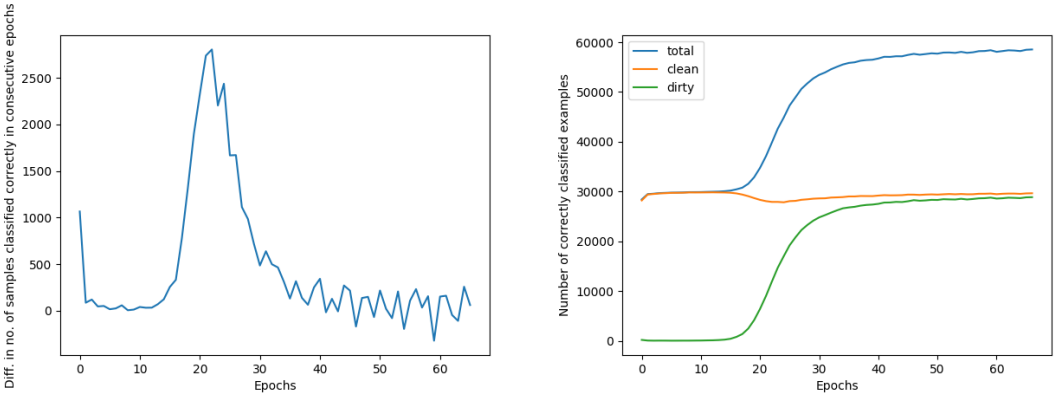
This idea can be understood by having a closer look at 4. Each of the rows presents graphs related to the DNN training on a dataset with a specific noise rate. The first curve shows the difference in the number of classified examples in consecutive epochs and the second one shows the % of clean and noisy examples in all training examples classified correctly according to the noisy training set.

For a particular dataset and noise rate, the first graph depicting the rate of classification is divided roughly into 4 stages: decrease, increase, decrease, constant. Before the minima of the first graph, we can observe from the second graph that clean examples are fit in majority and after it, model starts overfitting on noisy examples.

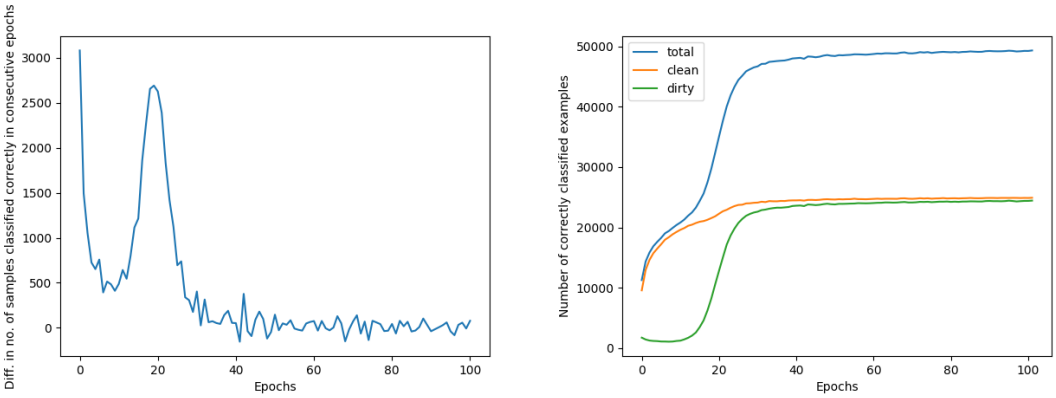
Thus, from these observations, we can infer that the minima of the first graph is a stage where the model has the most reliable probability estimates when training on noisy labels. This stage can be used as a starting point for the reweighting algorithm. We can observe that for different datasets and different noise rates, the minima occurs at different training epochs. Thus, this provides a principled way of finding a training stage where the model has learnt patterns from the data without overfitting on noise. In our algorithm, this helps in starting a reweighting algorithm for training on the noisy dataset.

The optimal epoch to start reweighting E_{opt} can be figured out by running the model for a few epochs, plotting the difference between examples classified in consecutive epochs and finding out the minima in the graph. The minima can be identified using criteria similar to that used in early stopping[2].

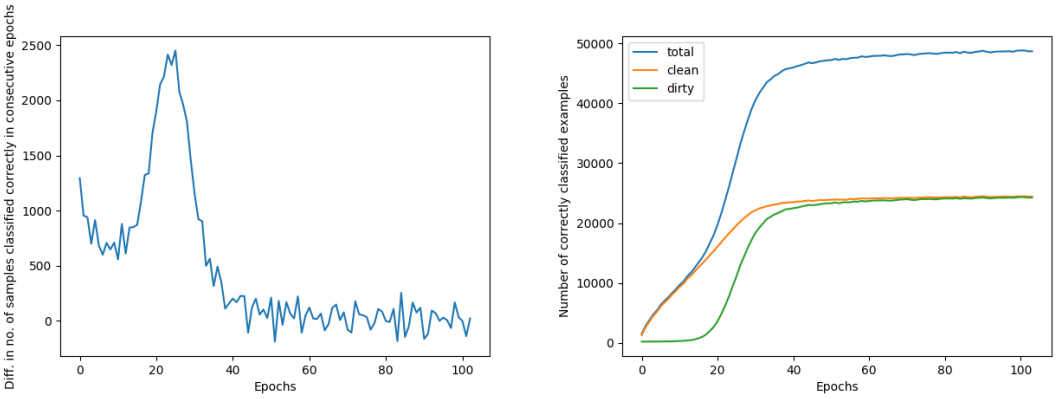
Explanation: This observation can be explained by reasoning over the DNN training process. For clean as well as noisy datasets, initially the rate of classification will be high, which slowly decays down with increase in epochs. The model initially fits clean patterns with a decreasing rate, but due to noise in the dataset, it starts fitting the examples with noisy labels, which causes an increase in the rate of classification. After this increase in rate, the rate decreases similar to that for clean datasets.



(a) MNIST- 50% symmetric noise



(b) CIFAR 10- 50% symmetric noise



(c) CIFAR 100- 50% symmetric noise

Fig. 4: In each row, the first figure shows the difference in number of training examples classified between 2 consecutive epochs and the second figure shows the % of clean and noisy examples in all training examples classified correctly according to the noisy training set.

4.5 Algorithm 1: Re-weighting

Input: Noisy data: D = $\{x_i, y_i\}_{i=1}^{i=N}$, learning Rate η, Number of epochs E_{total} Optimal epoch to start reweighting E_{opt}
Output: Model parameters θ

$\forall x \in \{1, \cdots, N\}, W_x \leftarrow 0.5$
for $i \leftarrow 1$ **to** E_{total} **do**
  Train model with parameters θ, learning rate η, for 1 epoch using the weighted loss $L = \sum_{j=1}^{N} l_j W_j$ using weights W, where W_j and l_j are the weight and cross-entropy loss for j^{th} example
  if $i \geq E_{opt}$ **then**
    $\forall j \in \{1, \cdots, N\}$ $p_j = Pr(\hat{y}_j = k | x_j, \theta)$, where x_j belongs to class k and $\hat{y}$ denotes the model's prediction
    $p_{max}, p_{min} \leftarrow max_j(p_j), min_j(p_j)$, where $j \in \{1, \cdots, N\}$
    $\forall j \in \{1, \cdots, N\}, W_j \leftarrow calc\_weight(p_j, p_{max}, p_{min})$ // *calc_weight* calculates the weight for an example using one of the functions in 3.
  end
end

Algorithm 1: Systematic Iterative Reweighting Algorithm

4.6 Algorithm 2: Weighting using Regularization

Input: Noisy data: D = $\{x_i, y_i\}_{i=1}^{i=N}$
Hyperparameters: Regularization values V in descending order, such that $|V| = k$, Epochs = E
Output: Example weights W, a vector of length N, with $0 \leq W_i \leq 1$ $\forall i$ $1 \leq i \leq N$

Initialize DNN model parameters θ
$T \leftarrow 0$ //Denotes number of examples which are given non-zero weight
$\forall x \in \{1, \cdots, N\}, W_x \leftarrow 0$
for $i \leftarrow 1$ **to** k **do**
  $\lambda \leftarrow V[i]$
  for $e \leftarrow 1$ **to** k **do**
    Train the model with parameters θ for 1 epoch using Cross-Entropy loss and L2 regularization with parameter λ.
    $X_{curr} \leftarrow$ Set of examples classified correctly at end of epoch e
    $X_{prev} \leftarrow$ Set of examples having non-zero weight at end of epoch $e - 1$
    $X_{new} \leftarrow X_{curr} \setminus X_{prev}$
    w $\leftarrow 1 - \frac{T}{N}$
    $\forall x \in X_{new}, W_x \leftarrow w$
    $T \leftarrow T + |X_{new}|$
  end
end

Algorithm 2: Weighting examples using Regularization

The above algorithm generates weights for each example which we used to train a model which optimizes this weighted loss.

5 Experiments

In this section, we evaluate robustness of algorithm 4.5 on multiple datasets, present experimental details, results and ablation studies for our model. Using Algorithm 4.6, we were not able to filter out the noisy examples and resulted in overfitting on the noisy examples.

5.1 Datasets

The datasets used for evaluation of the model are: MNIST, CIFAR-10, CIFAR-100. Noisy datasets are constructed from MNIST, CIFAR-10 and CIFAR-100 datasets by removing some randomly picked training samples from a class and putting them either all into 1 other class or by randomly assigning a label to them which is different from the original class. We perform experiments using 3 specific noise cases for MNIST, CIFAR-10 and CIFAR-100: 50% symmetric noise, 20% symmetric noise and 45% pair-flip noise.

5.2 Implementation Details

We implement all methods and algorithms using PyTorch and conduct experiments on a NVIDIA GeForce GTX 1080 GPU. For all experiments, Adam optimizer is used with the default learning rate(0.001), with a batch size of 128 and we run for 200 epochs for each experiment. We use the neural network architecture as described in [4] for our experiments. We use data augmentation in our experiments, and use the horizontal flip and random rotation transformation with a maximum rotation of 30 degrees. The epoch at which reweighting is started is determined from the plots 4

5.3 Results

We will discuss the results obtained by running the algorithm 4.5 on the datasets MNIST, CIFAR-10, CIFAR-100 on 3 noise cases: 45% pair-flip noise, 50% symmetric noise, 20% noise. Table 2 shows the accuracy obtained by the model on the clean test set of the individual datasets. Figure 6 shows the accuracy curves which include the training accuracy calculated with noisy labels, training accuracy containing clean labels and the accuracy on the clean test set for the 50% symmetric case. Figure 5 shows the histograms of probability values predicted by the model at the end of the 200 epoch training process. We compare the performance of our method with [4] as the baseline.

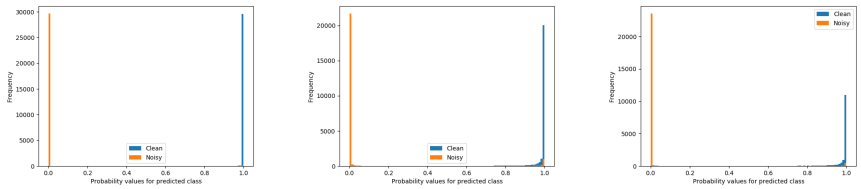
Results on MNIST: The reweighting algorithm applied on the model is able to separate examples with clean labels from those with noisy labels successfully. This is clear from the probability histograms, which show clean separation

Dataset\ Noise	Pair-45	Symmetric-50	Symmetric-20
MNIST	87.63	91.32	97.25
CIFAR10	72.62	74.02	82.32
CIFAR100	34.81	41.37	54.23

Table 1: Results of the baseline paper [4] on the respective clean test datasets after 200 epochs

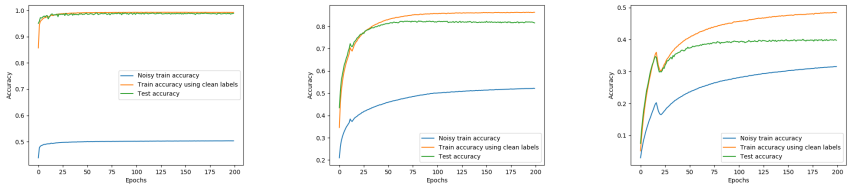
Dataset\ Noise	Pair-45	Symmetric-50	Symmetric-20
MNIST	98.76	98.9	99.42
CIFAR10	81.77	81.57	87.7
CIFAR100	40.00	39.77	58.07

Table 2: Results of our algorithm 4.5 on clean test datasets after 200 epochs



(a) MNIST Dataset (b) CIFAR-10 Dataset (c) CIFAR-100 Dataset

Fig. 5: Probability histograms after the training for 200 epochs for 3 different dataset with 50% symmetric noise.



(a) MNIST Dataset (b) CIFAR-10 Dataset (c) CIFAR-100 Dataset

Fig. 6: Accuracy curves representing training accuracy calculated with noisy labels, clean labels and test accuracy for 200 epochs for 3 datasets with 50% symmetric noise.

of clean and noisy examples. This is also reflected in the high accuracy of $\approx 99\%$ that the model achieves irrespective of the noise case considered. For high noise cases, the accuracy obtained by our algorithm is $\approx 10\%$ higher than the baseline, demonstrating effectiveness of the method to separate noisy examples.

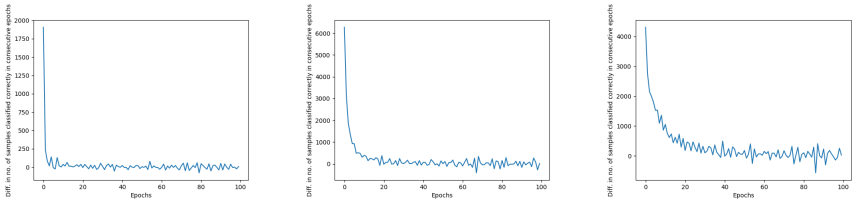
Results on CIFAR-10: Our algorithm achieves 5%-10% higher accuracies than the baseline with a corresponding good separation of examples, which is visible through the probability histograms.

Results on CIFAR-100: Our algorithm performs better than the baseline in 2 noise cases and achieves lower yet similar and comparable performance on the 50% symmetric noise case. The major difficulty in training on CIFAR-100 is the low number of clean training examples available per class. We think that such a problem would not occur in real-world noisy datasets, as they mostly tend to be large in size, due to low cost required to collect them, for eg: images obtained from search engines using queries as labels.

Discussion: Our algorithm performs better than the baseline [4] in every dataset and noise case, except one noise case in CIFAR-100 dataset. The baseline work drops instances based on the low-loss criterion, which is equivalent to giving hard weights of 0 and 1 to examples. Also, number of instances to be dropped follows a fixed schedule. In contrast, our method does explicit soft reweighting of examples, without following any fixed schedule. Also, starting of the reweighting is done using a systematic way based on concrete experimental evidence about the DNN training process. These observations and insights about the DNN training process are not tied to reweighting in any way and can be used by other works independently.

5.4 Ablation Study

Observation of DNN Training on Clean Dataset In section 4.4, we observed the DNN training process on dataset with noisy labels and derived insights regarding the training process. We demonstrate here that running on a clean dataset gives a pattern 7 different from that obtained in 4, proving that the pattern obtained is a characteristic of noise in the labels.



(a) MNIST Dataset (b) CIFAR-10 Dataset (c) CIFAR-100 Dataset

Fig. 7: Difference between number of examples classified by the model in consecutive epochs on clean datasets

6 Future Plan

One of the directions that can be pursued are performing experiments on real-world noisy datasets to check the validity of the approaches described here. Another direction is formulation of novel training algorithms based on regularization.

References

1. Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al.: A closer look at memorization in deep networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 233–242. JMLR. org (2017)
2. Caruana, R., Lawrence, S., Giles, C.L.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In: Advances in neural information processing systems. pp. 402–408 (2001)
3. Goldberger, J., Ben-Reuven, E.: Training deep neural-networks using a noise adaptation layer (2016)
4. Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., Sugiyama, M.: Co-teaching: Robust training of deep neural networks with extremely noisy labels. In: Advances in neural information processing systems. pp. 8527–8537 (2018)
5. Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. arXiv preprint arXiv:1712.05055 (2017)
6. Natarajan, N., Dhillon, I.S., Ravikumar, P.K., Tewari, A.: Learning with noisy labels. In: Advances in neural information processing systems. pp. 1196–1204 (2013)
7. Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., Qu, L.: Making deep neural networks robust to label noise: A loss correction approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1944–1952 (2017)
8. Ren, M., Zeng, W., Yang, B., Urtasun, R.: Learning to reweight examples for robust deep learning. arXiv preprint arXiv:1803.09050 (2018)
9. Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., Fergus, R.: Training convolutional networks with noisy labels. arXiv preprint arXiv:1406.2080 (2014)
10. Wang, Y., Liu, W., Ma, X., Bailey, J., Zha, H., Song, L., Xia, S.T.: Iterative learning with open-set noisy labels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8688–8696 (2018)