Training with Noisy Labeled Dataset

Cheng-Wei Lin
Department of CSIE
National Taiwan University
r10922102@ntu.edu.tw

Yu-Che Huang
Department of CSIE
National Taiwan University
b07902066@ntu.edu.tw

Kai-Hsiang Chou
Department of CSIE
National Taiwan University
b07705022@ntu.edu.tw

Abstract

Training on noisy data has become an important research area since the datasets are not guaranteed to be clean and accurate. In this project, we survey some common techniques of noisy training and present a framework that combines multi-round training and label refurbishment. We showed that our method outperform the normal training process. In the two-class classification problem, even when 40% of the data is mislabeled, our model still gets an accuracy of 85%.

1 Introduction

Deep neural networks (DNNs) have demonstrated outstanding performance, even surpassing human performance, in a variety of tasks, including computer vision, information retrieval, and language processing, due to the recent development of large-scale datasets. Their success, however, relies on the availability of large amounts of accurate data, which can be costly and time-consuming to retrieve. To reduce the cost on labelling, some turn to non-expert sources like Amazon's Mechanical Turk or crowdsourcing. The usage of these sources, however, frequently leads in erroneous labelling. In addition, labels can also be adversarially manipulated by attackers to attack on the models. These faulty labels are referred to as noisy labels since some of them depart from the ground-truth labels and are thus considered corrupted. Because of the learning capability that cause DNNs to overfit on corrupted labels, DNNs are known to be prone to noisy labels, resulting in poor generalizability on a test dataset. Regularization techniques are used in some common mitigations, such as data augmentation, weight decay, dropout, and batch normalization. While these strategies can mitigate the effects of noisy labels, they do not totally eliminate the problem of overfitting.

In this work, we proposed an algorithm that combined several well-known noisy training methods to achieve a better result. We train the model, extract the outlying data by the correlation with the top singular vector of the covariance matrix, as mentioned in **?**, remove them from the dataset, and redo the task. Moreover, we perform the label refurbishment. The removed data is relabeled and added back to the training data if the confidence of the label generated by the model surpasses a threshold.

We set up an experiment to verify the efficiency of the proposed algorithm. From the CIFAR-100 dataset, we selected two similar classes, and randomly shift some of their labels. The experiment shows that our algorithm outperforms the normal training which doesn't apply any noisy label mitigation.

2 Background

2.1 Noisy Learning

In this section, we briefly discuss the existing methods of DNN training on the noisy labeled dataset. Following the taxonomy proposed by Song et al.Song et al. [2021], we categorize deep-learning-based

Preprint. Under review.

noisy learning into six categories. For each method, we would discuss the limitation of these methods and they help inspire our algorithm.

Data Cleaning By removing the examples whose labels are likely to be corrupted can result in clean training data. Given that mislabeled examples tend to have higher weights than correct examples, bagging and boosting can be used to select and remove examples with higher weights. Other methods, such as k-nearest neighbor, outlier detection, and anomaly detection, are also commonly used to filter out the mislabeled examples that have abnormal distribution. However, these methods may cause over-cleaning issues, which remove correct examples and therefore result in a less accurate distribution in the dataset.

Robust Architecture Researchers have tried to fix the issue by changing the architecture of the DNNs. A common approach is to add a Noise Adaptation Layer on top of the softmax layer, in order to model the noise transition matrix (i.e., the noisy label transition pattern.) However, the drawback of these methods is that they treat all examples equally so that they can't identify the mislabeled examples. Another approach is to design dedicated architectures that are capable of estimating the label transition probability. Nevertheless, these methods are limited to specific architectures and can't be applied to other architectures.

Robust Regularization The problem of noisy-labeled data comes from the fact that DNNs are prone to overfitting. Therefore, by avoiding overfitting, the model would be more robust when training on the mislabeled dataset. Some widely-used regularization techniques include data augmentation, weight decay, dropout, and batch normalization. The explicit regularization techniques, such as dropout and weight decay, can be difficult in tuning the hyperparameters or require deeper architecture to maintain the same learning capability. The implicit regularization techniques, such as data augmentation or label smoothing, improve the generalization capability but are often harder to converge.

Loss Adjustment The idea of loss adjustment is to reduce the negative impact of the mislabeled data on the loss of all training data. It can be adjusted into three categories. *Loss correction* estimate the label transition matrix and utilize it to correct the loss. This requires prior knowledge of the clean data so that the transition matrix can be estimated. *Loss reweighting* apply different weights to examples, to reduce the impact of mislabeled data, and emphasize the correct examples. This requires tuning the hyperparameter, which can be difficult in practice. *Label refurbishment* adjust the loss by the refurbished label, which is generated by the convex combination of the original (probably noisy) label and the predicted one. The problem of it is that, if there are too much mislabeled data, the DNNs may overfit these incorrect data. Label refurbishment is used by our algorithm, and we will discuss the impact of the proportion of mislabeled data on the accuracy of noisy training.

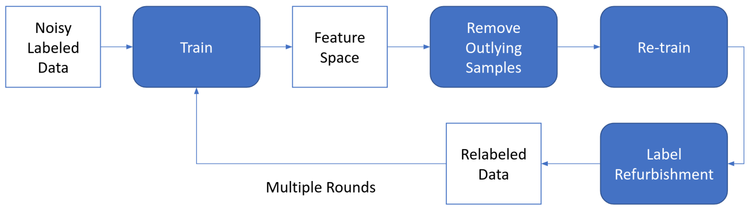
Sample Selection Some recent researches adopted sample selection to remove the mislabeled examples. A common technique is *Multi-network Learning*, which uses several DNNs with different architecture. In the co-learning and collaborative model, a pre-trained mentor model would "guide" the student models using the data that is probably correct. Another common technique is *Multi-round Learning*, which doesn't have to maintain several DNNs. Instead, it iteratively removes the outliers, and retrain with the refined dataset. Though learning using sample selection usually works well, it suffers from accumulated error due to wrong selection. Similar to the problem of label refurbishment, if the ratio of mislabeled examples or uncleared classifications is too high, the DNNs might remove the correctly-labeled examples and overfit the mislabeled examples. We apply the multi-round learning to our algorithm, and will discuss its effectiveness in the experiments.

2.2 Spectral Signature

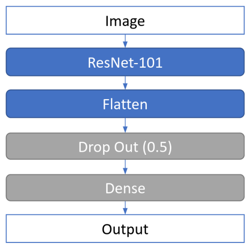
The *backdoor* attack on the DNNs is a type of data poisoning. They are considered dangerous because, while manipulating the output on carefully constructed examples, they do not affect the overall performance, and that the predictions on the benign data look benign. This makes backdoor attacks hard to be detected. Tran et al. **?** proposed mitigation to the backdoor attacks. Combining several techniques of robust statistics, which they called *spectral signature*, can separate the mislabeled data. Intuitively, when the training examples for a certain label consist of two sub-populations, the training set for that label may have been corrupted. There will be a large number of correctly labeled examples and a small number of mislabeled examples. If two populations are sufficiently well-separated, the corrupted examples can be recognized and therefore removed by using singular value decomposition, according to the aforementioned robust statistics procedures. That is, we can detect the outliers in the representation level.

3 Methodology

3.1 Algorithm Design and Implementation



(a) Flow chart of the algorithm



(b) The architecture of the testing model.

Figure 1: The flow chart and the model architecture.

Our algorithm combined multi-round learning and label refurbishment. The flow chart is showed in Figure 1a.

To train a model with noisy labeled data, we first train it normally. Then, for every label class in the dataset, apply spectral signature to the feature space of the class according to the trained model. The examples with the top $\frac{\epsilon}{2^{round}}$ scores are removed from the dataset. The formula $\frac{\epsilon}{2^{round}}$ is chosen such that earilier round should remove more outliers to prevent overfitting from the very start of the training. Next, the cleaned dataset is used to retrain the neural network. Following retraining of the model, the removed labels are then refurbished. We use the newly trained model to predict the label of every removed example, and if the model is confident about the true label, the example is added back to the dataset with the predicted label. The process is repeatedly several times until either the convergence or the threshold reach.

We implement the algorithm in Tensorflow. The source code is attached in the submission.

3.2 Experiment Design

We design a experiment to examine the effectiveness of our algorithm. The task is to classify two classes of examples sampled from CIFAR-100, and the ratio r of the labels are swapped. We choose *(caterpillar, worm)* and *(apple, bus)* for demonstration. We train the model using our algorithm on the constructed noisy dataset containing only two selected classes. After each round, we calculate the overall accuracy of the model and the label accuracy. The label accuracy is defined as the accuracy of the refurbished label, that is, what ratio of the refurbishment is correct. We perform the experiment on several different mislabeled ratio r. The model for testing is ResNet-101. We use a pre-trained ResNet-101 on ImageNet, connected with a dropout (rate=0.5) and a fully-connected layer, and perform a transfer learning on the model. The model architecture is shown in Figure 1b. With accurate labels, this model reaches 0.95 accuracy. In the experiment, every training, including the transfer learning of the base model, uses an SGD optimizer with a learning rate of 0.001 and momentum 0.9, and is trained for 32 epochs. The confidence threshold is set to 0.99.

Given that the outlier detection was performed in the feature space, we wonder if the distance in the feature space affect the effectiveness of our algorithm. Therefore, we choose several pair of classes that is known to be difficult to classify. By this experiment, we can analyze whether the algorithm still works in the case when classifications are unclear.

Method	$r = 0$	$r = 0.1$	$r = 0.2$	$r = 0.3$	$r = 0.4$	$r = 0.45$
Normal Training	0.95	0.88	0.785	0.635	0.525	0.515
Noisy Training	**0.95**	**0.925**	**0.915**	**0.91**	**0.85**	**0.85**
Label Accuracy	0.956	0.901	0.897	0.863	0.825	0.825

Table 1: The testing accuracy of the normal training and noisy training after 64 rounds.

4 Results

Our first experiments showed that, after 64 rounds of training and label refurbishment, our noisy training method successfully removes and refurbishes the mislabeled data, for it clearly outperforms the normal training method. The full experiment results are listed in Table 1. Starting from the case when $r = 0.1$, while the accuracy of the normal training fall by 7%, the accuracy of our noisy training algorithm only falls by 0.25%, and the accuracy of refurbishment is over 0.9. Even in the case when 45% of the data is mislabeled, the noisy training algorithm still maintains the accuracy of 0.85, while the normal training is only a little better than random guessing. Note that since this is a two-class classification problem, the baseline (random guessing) is 50% and that if $r = 0.5$, the label gives no information. Therefore, we only tried the cases when $r < 0.5$.

Then, we look into how each component of the algorithm works.

First of all, we want to examine whether outlier detection works. For the *(caterpillar, worm)* case, we plot the feature space of the model in the first round, that is, when we have not to do any outlier removal or label refurbishment. The histogram is shown in Figure 2a and Figure 2b. To our surprise, there is no clear cut between the two sub-population, as mentioned in **?**, but the noisy training still works. We propose two possible explanations for this. First, in the original work, they also added watermarking to the mislabeled examples, which may help separate two sub-population. Meanwhile, we only swap the label but the content of the images is still the same. Second, while there is no clear cut when the correlation is small, the number of correct data is so overwhelming so that the mislabeled data cannot influence the model, while when the correlation is large, but correct examples and mislabeled examples are removed. Therefore, even though the outlier removal is not as effective as we originally expected, the noisy training still works. As we can see in Figure 2c and Figure 2d, after 64 rounds of training and refurbishment, the histogram is rather clean and most outliers with high correlation are removed. The results for the other experiments *(apple, base)* are shown in Figure 2e to Figure 2f. We also see similar trends.

The second question is, whether the label refurbishment works as expected. In both experiments (Figure 3a and Figure 3b), we can see that the label accuracy (the ratio of correct refurbishment) is an upward trend in general. After 64 rounds of training, the accuracy of the refurbishment has exceed 90%.

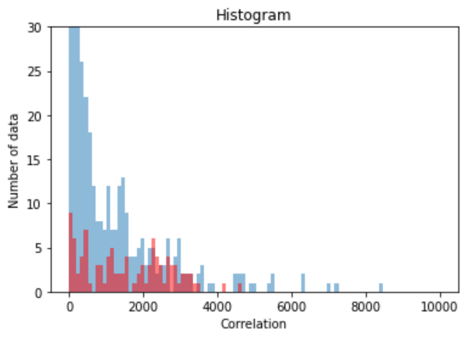
5 Discussion

5.1 Cases when the algorithm fails

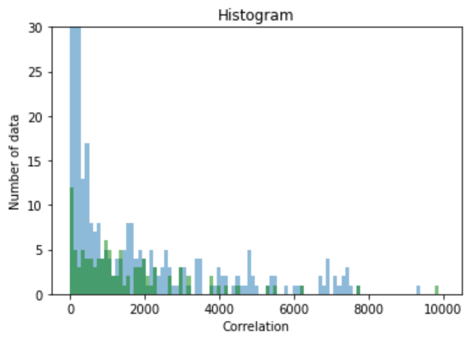
We observed that in some label pairs (*(automobile, truck)* and *(apple, orange)*), our algorithm failed, such that the normal training process outperforms the noisy training process. We suspect that both labels have similar distribution in the feature space. Take *(apple, orange)* with $r = 0.2$ as example. From Figure 4a and Figure 4b, we can see that there are no cuts between two classes. Therefore, by cutting of the outliers, we actually removed the correct examples. Therefore, the label accuracy, as showed in Figure 4c, falls during the training.

5.2 Limitation

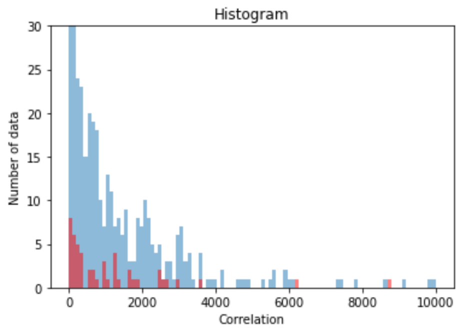
Due to the time constraint, we have not compared our noisy training algorithm with other well-known algorithms. Therefore, while we know that our method outperforms the normal training process, we do not know how much our improvement is.



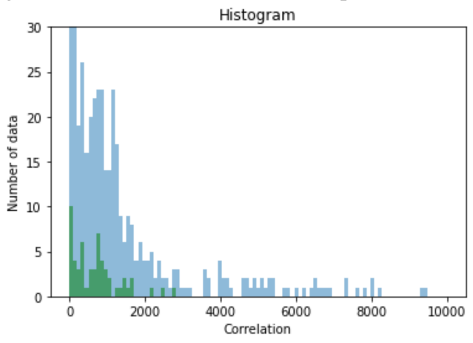
(a) The correlation of caterpillar in the first round. The red bars are the mislabeled examples.



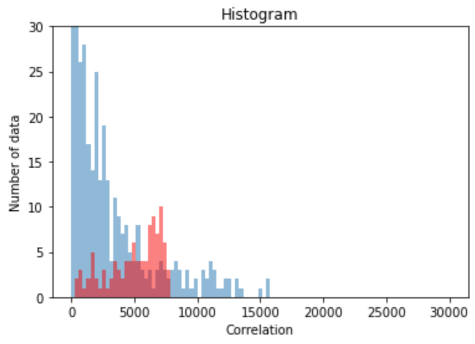
(b) The correlation of worm the first round. The green bars are the mislabeled examples.



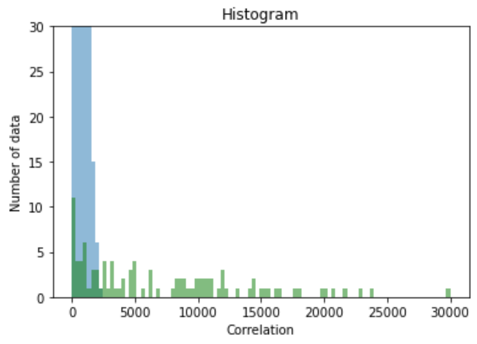
(c) The correlation of caterpillar after 64 rounds. The red bars are the mislabeled examples.



(d) The correlation of worm after 64 rounds. The green bars are the mislabeled examples.



(e) The correlation of apple after 64 rounds. The red bars are the mislabeled examples.



(f) The correlation of bus after 64 rounds. The green bars are the mislabeled examples.

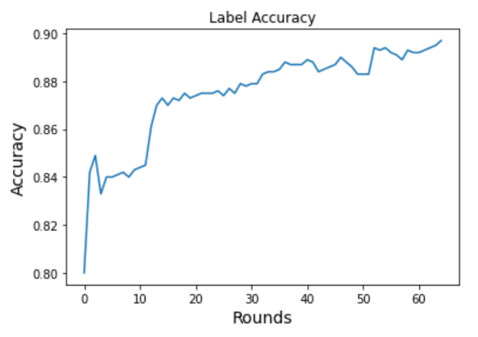
Figure 2: Correlation of the experiments.

Also, the common drawback of multi-round noisy training is that the training process is too slow. In our experiment, we trained 129 models just for a simple task. When the model becomes bigger, the algorithm would take too much time to train in practice.

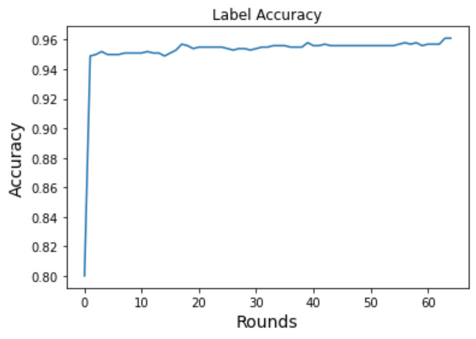
Lastly, due to the long training time, we are unable to perform large scale experiments. As we found that in some label pairs, our method does not work well, but we do not know how often these cases are.

6 Conclusion

In this report, we present a noisy training model that combines multi-round learning and label refurbishment. Dataset can be noisy, i.e. contain mislabeled examples, for many reasons, and sometimes considered inevitable. By repeatedly doing outlier removal and relabeling the incorrect

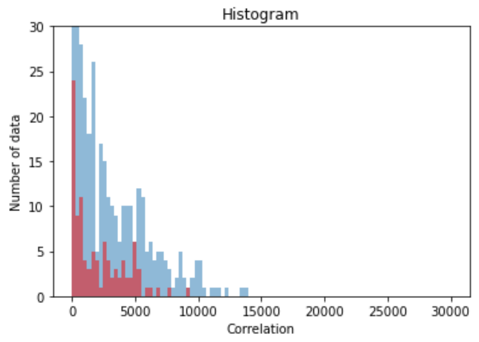


(a) The label accuracy of the noisy learning from *caterpillar* and *worm*.

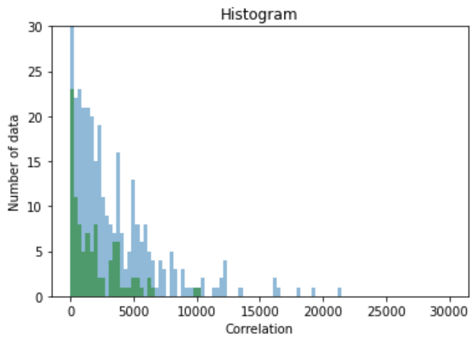


(b) The label accuracy of the noisy learning from *apple* and *bus*.

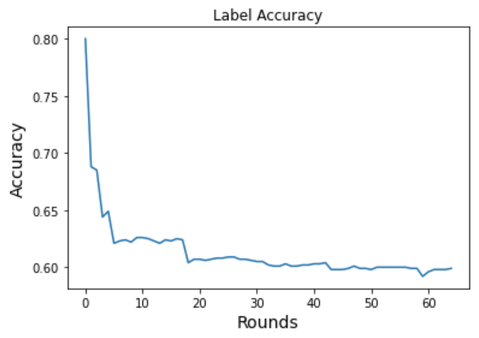
Figure 3: Label accuracy of the experiments.



(a) The correlation of apple after 64 rounds. The green bars are the mislabeled examples.



(b) The correlation of orange after 64 rounds. The green bars are the mislabeled examples.



(c) The label accuracy of the noisy learning from *apple* and *orange*.

Figure 4: The results of failed noisy training on *(apple, orange)*.

examples, we can improve the performance of the model when training on a noisy dataset. Our preliminary experiment shows that this model clear outperforms the normal training process. However, this algorithm does not perform well when two classes get too close in the feature space, and we consider this a possible direction of improvement in the future.

References

H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee. Learning from noisy labels with deep neural networks: A survey, 2021.