Learning with Noisy Lables

Abstract

The performance of the Deep Neural Network (DNN) highly relies on the quality of the manually annotated datasets. However, due to the superior memorialization ability of DNN, it will overfit the noise when there are noisy labels in the datasets. Therefore, it's necessary to train DNN with better robustness. In this report, we first implement the required model with three types of commonly used layers. Then, we propose a novel method that was inspired by the general pipeline of the teacher-student model. To verify the performance of the aforementioned methods, we conduct extensive experiments on CIFAR-10. Moreover, we discuss the influence of some key hyper-parameters, different loss functions, and the varying noise rates.

1 Introduction

The data-driven deep neural networks (DNNs) have achieved superior performance on many classification tasks [12, 19], which benefit from the large-scale datasets with clean label annotations [16]. However, in real-world applications, it's inevitable to mislabel some samples when collecting the datasets with massive data. According to work [22], DNNs can perfectly fit any training datasets even with completely random-annotated labels, which shows that DNNs are prone to overfitting noise and thereby causing performance degradation. This problem can be called as Learning with Noisy Labels (LNL). In this report, we implement different methods to train DNNs robustly with noisy labels.

2 Related work

In this section, we briefly review the relevant literature about learning with noisy labels with deep neural networks (DNNs). The most existing methods use the noise-tolerant loss function to achieve robust classification. Vahdat et al. [19] and Patrini et al. [12] leveraged the ground-truth noise transition matrix for loss correction. Sukhbaatar et al. [16] obtained noise distribution through the end-to-end noise transformation matrix. However, the aforementioned methods suffer from the lack of memorization ability due to their poor usage of the supervised label. To address this issue, iteratively relabeling [8, 20], reweighting the training samples [11, 14], and proposing a more robust loss function [3, 15] are considered as three optional advanced strategies for improving the performance of a single end-to-end model. Chang et al. [2] reweighted the training samples base on their predicted variance, especially to assign higher weights to samples with higher variance. Amid et al. [1] proposed a logistic loss with bi-temperature to obtain better robustness. Work [18] proposed a loss function to separate part of the noisy samples, which enables the model to memorize more real samples. As an extension and generalization of work [3] and [4], Zhang et al. [24] proposed a novel noise-robust loss function that takes advantage of mean absolute loss and categorical cross-entropy. Yi and Wu [22] proposed an iterative model to update the labels with three different calculated losses. Reed et al. [13] and Tanaka et al. [17] modified the labels with respect to the prediction results.

Unlike the aforementioned approaches that using a single end-to-end model with different training strategies, the relatively higher performance is obtained in some recent works using dual models with similar strategies [5, 7, 9, 10, 23]. The main idea of this approach is to train two models simultaneously to make the models more robust against noisy labels. Han et al. [5] used the small-loss

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

trick within each mini-batch to cross-train two DNNs, and Yu et al. [23] improved this work by weakening the prediction consistency of the two models with disagreement data. Work [7] proposed a pre-trained mentor network for assigning the weights to samples, and used it to guide the training of the student network. Work [9] separated the noisy and clean samples to train two networks where each network leverages the subsets division from the other network. Work [10] trained the teacher model and student model simultaneously, and obtained robustness by minimizing the meta-loss between them. Inspired by the aforementioned methods, we follow the general teacher-student pipeline and combine the results from two different networks to gain useful knowledge for better robustness.

3 Method

3.1 Problem Statement

The c-class image classification problem in this paper is to classify the images into c classes with a certain percentage of noisy labels. Given a set of n training images $X = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, and each image $\mathbf{x_i}$ has a one-hot encoding ground-truth label $\mathbf{y_i} \in \{0,1\}^c$. Note that the label $\mathbf{y_i}$ can be corrupted as $\hat{\mathbf{y}}_\mathbf{i}$, which makes $\mathbf{x_i}$ a noisy sample. When minimizing an empirical risk with the categorical cross-entropy to learn the set of trainable parameters θ, the loss function is denoted as:

$$\mathcal{L}_{ce} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c}\hat{y}_{ij}\log f_j\left(\mathbf{x_i}, \theta\right) \tag{1}$$

where $f(\mathbf{x_i}, \theta)$ denotes the discriminative function for outputting a softmax probability over the c classes for each image, $log(\cdot)$ is an element-wise log function. In the remainder of this section, we describe the proposed method for learning noise-tolerant parameters on noisy labels.

Algorithm 1 Details of the Proposed Method

Require: Training image set $\mathbf{I_t}$, training label set $\mathbf{L_t}$, training index set $\mathbf{D_t}$, testing image set $\mathbf{I_o}$, testing label set $\mathbf{L_o}$, testing index set $\mathbf{D_o}$, parameters for the first optimizer θ_1, parameters for the second optimizer θ_2, the epoch number t, hyperparameters γ and ω.

for $t \leftarrow 1$ **do**
  sample a training batch $(\mathbf{I_t}, \mathbf{L_t}, \mathbf{D_t})$
  Truncated loss $\mathcal{L}_q \leftarrow = \mathrm{ResNet} - 34(\mathbf{I_t}, \mathbf{L_t}, \mathbf{D_t})$
  update parameters $\theta_1 \rightarrow \theta_{1_{new}}$
  update parameters $\theta_2 \rightarrow \theta_{2_{new}}$
end for
for $t \leftarrow 1$ **do**
  sample a training batch $(\mathbf{I_t}, \mathbf{L_t}, \mathbf{D_t})$
  SCE loss $\mathcal{L}_{sce} \leftarrow \mathrm{Prototype}(\mathbf{I_t}, \mathbf{L_t}, \mathbf{D_t})$
  update parameters $\theta_2 \rightarrow \theta_{2_{new}}$
end for
for $t \leftarrow 1$ **do**
  sample a testing batch $(\mathbf{I_o}, \mathbf{L_o}, \mathbf{D_o})$
  $output_{pro} \leftarrow \mathrm{Prototype}(\mathbf{I_o}, \mathbf{L_o}, \mathbf{D_o})$
  $output_{res} \leftarrow \mathrm{ResNet} - 34(\mathbf{I_o}, \mathbf{L_o}, \mathbf{D_o})$
  outputs $\leftarrow -\sum(\gamma\mathrm{Softmax}(\mathrm{output_{pro}}) + \omega\mathrm{Softmax}(\mathrm{output_{res}}))$
end for
return outputs

3.2 Proposed Noise-Tolerant Method

As shown in Algorithm 1, a novel method is proposed in this report which uses the idea of the teacher-student model. At first, we train the ResNet-34 as the teacher model with the truncated loss $\mathcal{L}_q$ [24]. Since we try to use the complex teacher model to guide the training process of the student model, we also utilize the obtained loss to update the parameters of the Prototype (student) model. As a result, the Prototype model can learn the useful feature information which was extracted by the

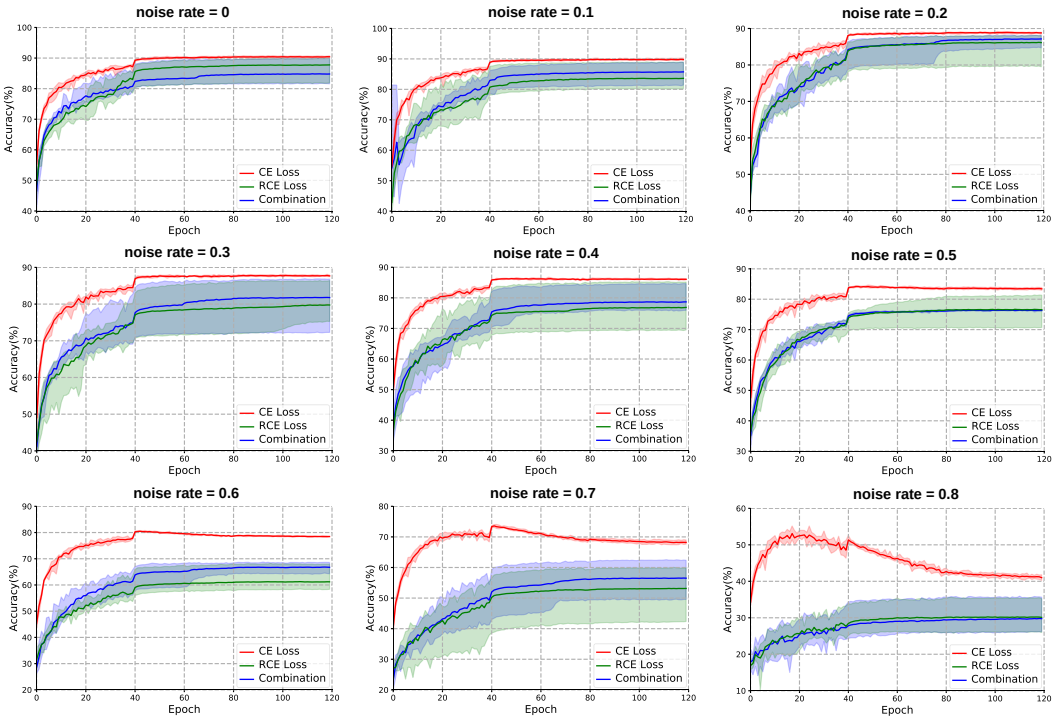

Figure 1: The test accuracy of the required model with different noisy rates (vary from 0 to 0.8 with step 0.1) in 120 epochs.

ResNet-34. Then, we train the Prototype model separately for improving its performance and better fit the training set. After obtaining the parameters with better robustness, the outputs of two models are jointly fed into the softmax function for calculating the probabilities towards different classes. At last, we assign different weights to the two outputs for further considering the difference between the teacher model and the student model. In this report, we also implement the required model of this task that composed of three different types of layers, and its structure is shown in Appendix A.

4 Experiment

4.1 Datasets

We conduct our experiments on CIFAR-10 to verify the effectiveness of the required model and our proposed method. Since the original dataset is clean, we manually corrupt their labels with symmetric noise, which means that the label noise is uniformly distributed among all categories.

Baselines. The comparison methods in this paper are as follows: (i) The required model of this task (as shown in Appendix A). (ii) Our proposed method. Moreover, we utilize three types of loss functions in our experiments, which are cross-entropy loss (CE loss), reversed cross-entropy loss (RCE loss), and the combination of CE loss and RCE loss. This combination can be expressed as:

$$\mathcal{L} = \alpha \mathcal{L}_{ce} + \beta \mathcal{L}_{rce} \tag{2}$$

where $\mathcal{L}$ denotes the obtained result, $\mathcal{L}_{ce}$ and $\mathcal{L}_{rce}$ are the results of CE loss and RCE loss, respectively. α and β are two hyperparameters for more effective and robust learning.

Experimental Setup The structure of the required model can be seen in the Appendix A. We use ResNet-34 [6] as the backbone network of our proposed method.

Moreover, we use the SGD optimizer with momentum 0.9, weight decay 10^{-4} and an initial learning rate of 0.01 which is divided by 10 after 40 and 80 epochs (120 in total). The hyperparameters α and

Table 1: Average test accuracy and standard deviation (5 runs) of the required model. The noise rate η equals 0 means the corresponding dataset is clean. The best result is in **bold**.

Loss Functions	Noise Rate η								
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
CE Loss	$\mathbf{90.51 \pm 0.25}$	$\mathbf{89.94 \pm 0.22}$	$\mathbf{89.02 \pm 0.21}$	$\mathbf{87.92 \pm 0.15}$	$\mathbf{86.46 \pm 0.20}$	$\mathbf{84.34 \pm 0.24}$	$\mathbf{80.56 \pm 0.18}$	$\mathbf{73.61 \pm 0.37}$	$\mathbf{54.35 \pm 0.81}$
RCE Loss	87.85 ± 3.22	83.69 ± 4.21	86.27 ± 3.61	78.52 ± 1.88	76.99 ± 7.13	76.77 ± 3.80	61.34 ± 4.02	53.25 ± 6.56	30.39 ± 3.54
Combination	84.94 ± 4.07	85.79 ± 3.98	87.22 ± 1.37	81.90 ± 5.95	78.77 ± 3.38	76.49 ± 0.15	66.90 ± 2.00	56.64 ± 5.63	29.86 ± 3.79

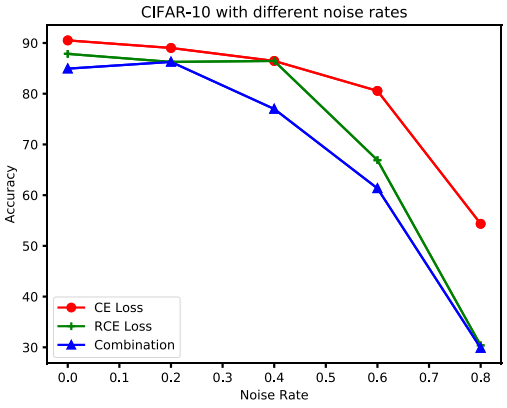

Figure 2: The impact of varied noise rates.

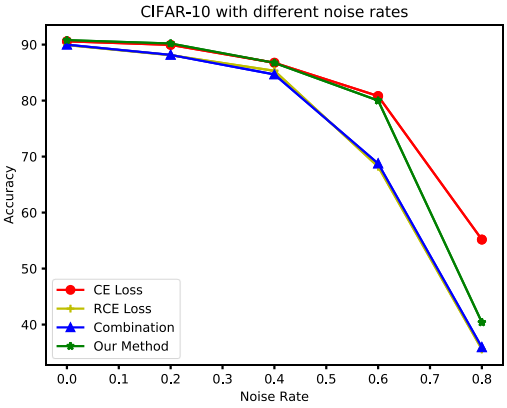

Figure 3: The performance comparison between different methods.

β are set to 0.1 and 1, respectively. Note that methods for data augmentation and generating the noisy labels are the same as in work [24].

Results and Discussion As shown in Table 2, the required method with CE loss can achieve higher accuracy than using other loss functions, which shows that CE loss is more noise-tolerant compare with the other two variants. In work [21], they proved that the combination of CE loss and RCE loss can significantly improve the performance. However, our obtained result is different from their conclusion, the reasons are as follows: (1) we utilize a relatively simple backbone, instead of some commonly used backbones in this fields (e.g., ResNet-34, ResNet-101, etc.), hence it's not sufficient enough for extracting the useful feature information; (2) when using the existing noise-tolerant loss function, the model's performance highly rely on the tricks during the training process. Without employing those tricks, it's difficult to obtain a stable performance when simply transferring those loss functions to collaborate with other backbones. The aforementioned conclusions can be further proved in Figure 1 and 2. Note that in Figure 2, the hyperparameters α and β are set to 0.1 and 1, respectively. It can be observed that CE loss can provide a more stable performance. Under the extreme noisy situation (the noise rate larger than 0.5), the methods using RCE loss (RCE loss only, and the Combination) fails to achieve good results, and it shows that RCE loss requires a more comprehensive backbone for the better results. Moreover, when noise rate equals to 0.8, the test accuracy curve of CE loss decline after 40 epochs, which shows that: (1) CE loss can helps the model to better memorize the massive data with randomly assigned labels. (2) the required model with CE loss is lack of robustness, hence it overfits the noise and cause the performance degradation.

Due to the limitation of the computational resources, we only run our method once on different noise rates. For a fair comparison, we used the best accuracy of the other three methods during their five runs in Figure 3. It can be observed that when the noise rate is less than 0.5, our method can slightly outperform other methods. However, when the dataset is extremely noisy, the obtained performance drops significantly. The reasons are as follows: (1) even we use the noise-tolerant loss function, it fails to help our model to obtain robustness when collaborating with the rest components. (2) the general pipeline of the teacher-student model can greatly help to improve the memorialization ability, and it makes our model easily overfit the noise. (3) the structure of the model is too simple to solve the LNL problem. To combat this drawback, some deeper and more comprehensive backbones should be considered in this task, and the structure of the student model should be further well-designed.

Table 2: Average test accuracy and standard deviation (5 runs) of the required model. The noise rate η equals 0 means the corresponding dataset is clean. The best result is in **bold**.

Hyperparameter β	0.0					0.2					0.4					0.6					0.8				
Hyperparameter α	0.001	0.01	0.1	0	1	0.001	0.01	0.1	0	1	0.001	0.01	0.1	0	1	0.001	0.01	0.1	0	1	0.001	0.01	0.1	0	1
0.001	67.1	66.4	86.22	60.18	90.57	58.04	64.21	84.92	57.32	88.73	54.36	63.01	82.35	51.55	86.41	61.02	67.93	77.0	60.91	55.12	60.54	69.26	63.79	60.22	55.12
0.01	79.91	85.95	88.37	85.47	90.42	84.1	84.61	86.73	79.59	88.88	76.05	76.60	83.65	81.66	86.56	66.45	67.73	78.43	70.45	81.60	52.51	53.70	56.45	46.81	54.76
0.1	90.24	90.48	90.59	90.36	90.75	88.32	88.72	88.83	88.55	89.23	86.52	86.51	86.62	86.79	86.78	80.94	78.7	81.07	80.91	81.45	54.47	54.85	57.61	57.71	59.39
0	32.92	62.95	86.5	14.87	90.79	31.26	59.1	84.44	9.18	89.22	27.40	57.0	81.26	9.72	86.76	31.85	37.23	86.48	9.76	80.81	30.38	60.61	56.24	12.12	55.18
1	90.03	88.73	89.98	89.92	89.1	80.41	88.24	88.16	88.11	87.87	77.97	78.03	84.67	85.22	78.37	81.68	68.85	68.75	68.21	81.79	34.55	32.95	35.93	35.58	42.45

5 Conclusion

In this report, we implemented several methods to make DNNs achieve better performance when fed with noisy datasets. It can be concluded that the CE loss can achieve a relatively better tradeoff between memorialization and generalization ability. Moreover, the noise-tolerant loss functions need to be utilized with some tricks, e.g., deeper backbone and other novel components. For the teacher-student pipeline, the transferred memorialization ability will let the student model meet the same problem as the teacher model. As for future work, we'll focus on highlighting the differences between the teacher and student model, instead of directly transferring the learned information.

References

[1] Ehsan Amid, Manfred KK Warmuth, Rohan Anil, and Tomer Koren. Robust bi-tempered logistic loss based on bregman divergences. In *Advances in Neural Information Processing Systems*, pages 15013–15022, 2019.

[2] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, pages 1002–1012, 2017.

[3] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. *arXiv preprint arXiv:1712.09482*, 2017.

[4] Aritra Ghosh, Naresh Manwani, and PS Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

[5] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, pages 8527–8537, 2018.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313, 2018.

[8] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5447–5456, 2018.

[9] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.

[10] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5051–5059, 2019.

[11] 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.

[12] 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.

[13] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[14] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

[15] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*, pages 5739–5748. PMLR, 2019.

[16] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[17] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

[18] Sunil Thulasidasan, Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. Combating label noise in deep learning using abstention. *arXiv preprint arXiv:1905.10964*, 2019.

[19] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605, 2017.

[20] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 839–847, 2017.

[21] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 322–330, 2019.

[22] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7017–7025, 2019.

[23] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? *arXiv preprint arXiv:1901.04215*, 2019.

[24] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018.

Appendix

A. The network structure of the required model

Table 3: The structure of the required model

layer name	output size	operations
conv1	32×32	$\begin{bmatrix} 3 \times 3, 64, batchnorm, relu \\ 3 \times 3, 64, batchnorm, relu \end{bmatrix} \times 2$
pool1	16×16	2×2 max pool, stride 2
conv2	16×16	$\begin{bmatrix} 3 \times 3, 128, batchnorm, relu \\ 3 \times 3, 128, batchnorm, relu \end{bmatrix} \times 2$
pool2	8×8	2×2 max pool, stride 2
conv3	8×8	$\begin{bmatrix} 3 \times 3, 196, batchnorm, relu \\ 3 \times 3, 196, batchnorm, relu \end{bmatrix} \times 2$
pool3	4×4	2×2 max pool, stride 2
fc1	256	flatten, fc [3136, 256], batchnorm, relu
fc2	1×1	fc [256, 10], softmax