# Label Noise Correction: Reproduction and Extensions

**Lingqi Huang**
Department of Statistics
University of Michigan - Ann Arbor
hlingqi@umich.edu

**Liangqi Tang**
Department of Statistics
University of Michigan - Ann Arbor
tanglq@umich.edu

Source Code Link: https://github.com/LiangqiTang/LabelNoiseCorrection

## 1 Introduction and Motivation

By 2019, Convolutional Neural Networks (He et al. (2015)) had established themselves as the leading approach for addressing many computer vision tasks(DeTone et al. (2016),Ono et al. (2018),Beluch et al. (2018),Krishna et al. (2017),Zhao et al. (2017),Redmon et al. (2016)) due to their ability to model complex patterns when provided with large amounts of labeled data. However, the labeling process—whether automatic or manual—is prone to errors, resulting in mislabeled samples, commonly referred to as **noisy samples**. Recent studies (Zhang et al. (2018)) have shown that CNNs tend to easily fit noisy labels, which negatively impacts their generalization performance.

To robustly handle noisy samples—particularly at high noise levels—without discarding them, which potentially removes useful information about the data distribution(Ding et al. (2018)), we implemented the algorithm proposed by Arazo et al. (2019). Contrary to most successful approaches that assume the existence of a known set of clean data(Ren et al. (2019), Hendrycks et al. (2019)), this algorithm models the loss distribution of clean and noisy samples using a beta mixture model. By combining bootstrapping (Reed et al. (2015)) and mixup data augmentation (Zhang et al. (2018)), we achieve a robust per-sample loss correction. We then trained a ResNet-18 model for the CIFAR-10 and ResNet-50 model for TinyImageNet datasets under the newly proposed loss function.

Our works are based on paper by Arazo et al. (2019). The re-implemented experimental results significantly outperform those obtained using the standard cross-entropy loss and standard mixup method at high noise levels. Additionally, we introduced several adjustments to the proposed algorithm to better handle extreme noise. Specifically, we set the weight of inputs in dynamic mixup to follow a beta distribution at noise levels of 80% and 85%, and a binomial distribution at a noise level of 90%. These adjustments resulted in substantial improvements in accuracy compared to the original experimental results.

## 2 Problem Statement

The problem setting involves a learner provided with a dataset of images, denoted as $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^N$ (details of the dataset are available in the Appendix 9.2). Here, $\boldsymbol{x}_i$ represents the image data, and $y_i$ denotes the corresponding one-hot encoded class label. However, the dataset may contain noise, meaning some images are mislabeled. Importantly, the learner does not know which samples are mislabeled or the proportion of mislabeled samples.

We simulate this setting by randomly altering the labels in the ground-truth training dataset while keeping the test dataset unchanged. The probability that the label of each sample is altered, which approximates the proportion of mislabeled samples, is referred to as the noise level. The learner trains the model using the noisy training dataset and subsequently applies it to the test dataset to make predictions. Finally, the model's performance is evaluated by calculating its prediction accuracy on the test dataset. Our goal is to train the model on noisy data while still achieving high accuracy.

## 3 Related Work

Recent studies (Ding et al. (2018)) addressing label noise focus on two scenarios: closed-set and open-set label noise. Our experiments are conducted under the closed-set scenario, where the set of possible labels $S$ is known and fixed, and all samples, including noisy ones, have their true labels within this set. For instance, when simulating label noise at an 80% noise level using the CIFAR-10 dataset (which has 10 classes), each sample has an 80% probability of being mislabeled, with the incorrect label randomly assigned to one of the other 9 classes. Consequently, the average noise level of the resulting dataset approximates 80%.

Approaches for handling label noise typically involve either removing corrupted data or applying loss correction methods. Simply discarding noisy samples, however, eliminates valuable information about the data distribution and fails to leverage noisy samples for representation learning (Ding et al. (2018)). Loss correction methods (Reed et al. (2015), Jiang et al. (2019), Zhang et al. (2018)) work by directly modifying the loss function or using the probabilities involved in loss computation to mitigate the misleading influence of noisy labels. However, these methods have limitations, as the noisy labels inevitably affect the learning objective. Additionally, relabeling approaches, such as curriculum learning (Guo et al. (2018)), often assume the availability of clean data, which restricts their applicability in scenarios where such data is not accessible.

The approach and algorithm we implemented address noisy labels using only the training loss of each sample, without requiring a clean dataset. Specifically, we fit a two-component beta mixture model to the training loss of each sample to distinguish between clean and noisy samples. Using the estimated probability of a sample being noisy, derived via the EM algorithm, we calculate a robust loss that integrates the bootstrapping method (Reed et al. (2015)) and mixup data augmentation (Zhang et al. (2018)). Under the ResNet-18 model and CIFAR-10 dataset, the newly proposed loss significantly improves accuracy in scenarios with 50% and 80% noise compared to cross-entropy loss or the standard mixup method. Furthermore, our adjusted dynamic mixup algorithm achieves substantially better results than those reported by Arazo et al. (2019) at noise levels of 80%, 85%, and 90%.

## 4 Theoretical Foundation and Algorithm of Learning with Label Noise

### 4.1 Dynamic Bootstrap method for Label Correction

In a common setting, the image classification problem can be formulated as: Given a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ and $y_i \in \{0,1\}^C$ being the one-hot encoding ground truth label corresponding to $x_i$ (in our study, $y_i$ could be noisy), let $h_\theta$ be the model and $\theta$ represent the parameters of the model, the goal is to fit $\theta$ by optimizing categorical cross-entropy loss function:

$$l(\theta) = \sum_{i=1}^{N} l_i(\theta) = -\sum_{i=1}^{N} y_i^T \log(h_\theta(x_i)) \tag{1}$$

where $h_\theta(x)$ is the predicted softmax probability by the model and $\log(\cdot)$ is applied to each element of the prediction vector.

However, when our sample is **under high-level noisy**, the standard categorical cross-entropy loss may be ill-suited to the task because it encourages fitting label noise (Zhang et al. (2017). Reed et al. (2015) proposed the **static hard bootstrapping loss** that introduced a perceptual term to the standard cross-entropy loss for label:

$$l_B = -\sum_{i=1}^{N} ((1-w_i)y_i + w_i z_i)^T \log(h_i) \tag{2}$$

where $w_i = 0.2$ weights the model prediction $z_i$(one-hot encoding predicted vector) in the loss function. Reed et al. (2015) also proposed the **static soft bootstrapping loss** use predicted softmax probabilities $h_i$ instead of $z_i$. However, applying fixed weights for all samples does not prevent fitting the noisy samples, Arazo et al. (2019) proposed **dynamic hard and soft bootstrapping losses** by using the beta mixture model to individually weight each sample, that to set $w_i = p(k = 1|l_i)$ be the probability of noisy sample which is estimated by beta mixture model(BMM) after each training epoch using cross-entropy loss for each sample $l_i$. We will explain the BMM model in section 4.2.

### 4.2 Beta Mixture Model(BMM)

The implementation of a BMM model on the loss is supported by a key observation made by Arazo et al. (2019): random labels take longer to learn than clean labels, meaning that noisy samples have higher loss during the early epochs of training (see Appendix Figure 1), allowing clean and noisy samples to be distinguished from the loss distribution alone (see Appendix Figure 2).

The probability density function of a mixture model of $K$ components are on loss $l$ is defined as:

$$p(l) = \sum_{k=1}^{K} \lambda_k p(l|k) \tag{3}$$

Arazo et al. (2019) proposed beta mixture model(BMM) (i.e. $p(l|k) \sim Beta(\alpha_k, \beta_k)$) that allows modeling both symmetric and skewed distribution over $[0, 1]$, which benefit for approximating the normalized loss distribution($l \in [0, 1]$) for mixture of clean and noisy samples where Gaussian Mixture model does not suit for. We use the Expectation Maximization(EM) algorithm to fit BMM to the observations, which will explained in section 4.2.1 and 4.2.2.

#### 4.2.1 E-step for BMM

We introduce the $\gamma_k(l) = p(k|l)$ to be the posterior probability of point $l$(latent variable) have been generated by mixture component $k$. Fixing parameters $\lambda_k, \alpha_k, \beta_k$, we update the latent variable using Bayes rule:

$$\gamma_k(l) = \frac{\lambda_k p(l|\alpha_k, \beta_k)}{\sum_{j=1}^{K} \lambda_j p(l|\alpha_j, \beta_j)} \tag{4}$$

#### 4.2.2 M-step for BMM

Fixing $\gamma_k(l)$, we update $\alpha_k, \beta_k, \lambda_k$ using a weighted version of the method of moments:

$$\alpha_k = \bar{l}_k \left( \frac{\bar{l}_k(1 - \bar{l}_k)}{s_k^2} - 1 \right), \quad \beta_k = \frac{\alpha_k(1 - \bar{l}_k)}{\bar{l}_k}, \quad \lambda_k = \frac{1}{N} \sum_{i=1}^{N} \gamma_k(l_i) \tag{5}$$

where

$$\bar{l}_k = \frac{\sum_{i=1}^{N} \gamma_k(l_i) l_i}{\sum_{i=1}^{N} \gamma_k(l_i)}, \quad s_k^2 = \frac{\sum_{i=1}^{N} \gamma_k(l_i)(l_i - \bar{l}_k)^2}{\sum_{i=1}^{N} \gamma_k(l_i)} \tag{6}$$

and $\{l_i\}_{i=1}^{N}$ are computed after each epoch using the cross-entropy loss function. In our re-implemented experiment, the EM algorithm will iterate at most 10 times, which assures convergence for most cases. To avoid the unstableness of the algorithm when observations(normalized sample loss $l_i$) near to 0 and 1, we bound the observations in $[\epsilon, 1 - \epsilon]$ where we pick $\epsilon = 10^{-4}$. Finally, we obtain the probability of a sample being clean or noise through the posterior probability $p(k|l)$, which we can plug it into the bootstrapping procedure.

### 4.3 Integration of Mixup and Dynamic Bootstrapping

Zhang et al. (2018) proposed the **mixup** technique that exhibits strong robustness to label noise, that trains on convex combinations of sample pairs($x_p$ and $x_q$) and their labels($y_p$, $y_q$):

$$x = \delta x_p + (1 - \delta) x_q, \quad l = \delta l_p + (1 - \delta) l_q \tag{7}$$

The mixup provides a mechanism to combine clean and noisy samples, computing a more representative loss to guide the training process. But under high-levels of noise mixing samples that both have incorrect labels is prevalent, the effectiveness of this method will reduce. Arazo et al. (2019) integrate **mixup** and **dynamic bootstrapping** to implement a robust per-sample loss correction approach:

$$\begin{aligned} l^* = &-\delta[((1 - w_p) y_p + w_p z_p)^T \log(h)] - \\ &(1 - \delta)[((1 - w_q) y_q + w_q z_q)^T \log(h)] \end{aligned} \tag{8}$$

where $w_p = p(k = 1|l_p)$ and $w_q = p(k = 1|l_q)$ are estimated by the BMM model after each epoch, $z_p, z_q$ are computed by doing an extra forward pass using original $x_p, x_q$, and $h$ is the softmax

probability produced from current model using mixup data. Arazo et al. (2019) also introduced the regularization term(Tanaka et al. (2018)) that preventing the assignment of all samples to a single class for singly minimizing loss:

$$R = \sum_{c=1}^{C} p_c \log\left(\frac{p_c}{\bar{h}_c}\right)$$

that $p_c$ denote the prior probability distribution for class $c$ and $\bar{h}_c$ is the mean softmax probability for the model for class $c$ across all samples in the dataset. Then we add $\eta R$ to $l^*$ with $\eta$ be the regularization coefficients.($\eta$ is set to 1 in all re-implemented experiments)

### 4.4 Mixup Guided by BMM

As previously mentioned, the effectiveness of the mixup method decreases under high levels of noise, i.e, 85% and higher. This occurs because clean samples are likely to be contaminated by noisy ones, leading to incorrect modification of the training objective. Arazo et al. (2019) shows the result of failure of convergence using M-DYR-H under 90%, and to achieve convergence, Arazo et al. (2019) proposed the dynamic mixup approach:

$$x = \left(\frac{\delta_p}{\delta_p + \delta_q}\right) x_p + \left(\frac{\delta_q}{\delta_p + \delta_q}\right) x_q \tag{9}$$

where $\delta_p = p(k = 0|\ell_p)$ and $\delta_q = p(k = 0|\ell_q)$. With this approach, we aim to assign greater weight to samples that are more likely to be clean during the mixup process.

## 5 Experiments

### 5.1 Re-implemented Experiments

To replicate the experimental results reported in Arazo et al. (2019), we used the CIFAR-10 dataset with the same seed, parameters, model architecture (ResNet-18), and methodology, achieving similar outcomes. Specifically, we reproduced the results using Cross Entropy (CE), Standard Mixup (M), and Mixup with Dynamic Bootstrapping and Regularization (M-DYR-H) under the BMM model at four different noise levels (0%, 20%, 50%, and 80%) (See Table 1).

When re-implementing the M-DYR-H model, we observed that Arazo et al. (2019) implicitly assumed that the mixture distribution with the larger initial mean would retain the larger mean throughout the EM algorithm, representing the noisy distribution. However, we believe a more reasonable approach is to compare the means of the two mixture distributions after the EM algorithm converges and designate the one with the larger mean as the noisy distribution. Using this adjustment, we re-implemented the model with a noise level of 80% and achieved overall better results than those reported in the original paper.

Additionally, we applied the GMM model under M-DYR-H method. Further details regarding parameters and algorithms can be found in the appendix.

### 5.2 Mixup Guided by BMM in High Noise Levels

As Arazo et al. (2019) mentioned, the proposed approach M-DYR-H fails to converge in CIFAR-10 with 90% noise. Here, We further adjusted the MD-DYR-SH(dynamic mixup + dynamic bootstrapping + regularization + soft to hard) as (9) mentioned to get better performance.(See Table 2) Under noise level of 80% and 85%, instead of setting the weight of $x_p$ and $x_q$ to be $\lambda = \frac{\delta_p}{\delta_p + \delta_q}$ and $1 - \lambda$, we draw $\lambda$ from a beta distribution such that

$$\lambda \sim Beta\left(\frac{\delta_p}{\delta_p + \delta_q}, \frac{\delta_q}{\delta_p + \delta_q}\right)$$

The intuition behind this approach is that we perform a conditional draw from the beta distribution(Little and Rubin (1986), chap4), which helps reduce the variation introduced by the neural network, while ensuring that the sampled $\lambda$ remains unbiased. This method outperforms the results of Arazo et al. (2019), achieving a approximately 3% higher accuracy.

Under a 90% noise level, the Beta distribution-guided dynamic mixup strategy achieves only about 56% accuracy, which is significantly lower than the performance of MD-DYR-SH reported by Arazo et al. (2019). To address this limitation, we propose an improved method by setting $n = 10000$, and replacing the weights for $x_p$ and $x_q$ , with a conditional draw. Specifically, we define $\lambda_1 = n\hat{p}$, where

$$\hat{p} \sim Binom\left(n, \frac{\delta_p}{\delta_p + \delta_q}\right)$$

Similarly, we obtain $\lambda_2$ by using the same procedure. We then update the weights such that the weight for $x_p$ becomes $\frac{\lambda_1}{\lambda_1+\lambda_2}$ and the weight for $x_q$ to be $\frac{\lambda_2}{\lambda_1+\lambda_2}$. This approach achieves a 4% higher accuracy compared to the results reported by Arazo et al. (2019).

### 5.3 Implemented Experiments on TinyImageNet

In this section, we implemented the main algorithms from Arazo et al. (2019), including MixUp and bootstrapping under BMM (Hard), on the TinyImageNet dataset.(See Table 3) We chose ResNet50 for its higher expressiveness. Due to limited computational resources, we froze all parameters except for the last linear layer and limited the training to 100 epochs. The results fell short of our expectations, as the MixUp + dynamic bootstrapping method did not outperform MixUp alone. Additionally, there are signs of underfitting in M-ImageNet and overfitting in M-DYR-H (BMM)-ImageNet.

### 5.4 Experiment Tables

| **Alg./Noise level (%)** | | **0** | **20** | **50** | **80** |
|---|---|---|---|---|---|
| CE | Best | 94.5(94.7) | 85.89(86.8) | 80.64(79.8) | 63.4(63.3) |
| | Last | 94.43(94.6) | 83.26(82.9) | 59.88(58.4) | 27.46(26.3) |
| M | Best | **95.48**(95.3) | 92.74(95.6) | 86.98(87.1) | 71.58(71.6) |
| | Last | **95.32**(95.2) | 91.96(92.3) | 77.12(77.6) | 43.81(46.7) |
| M-DYR-H(BMM) | Best | 93.84(93.6) | **93.95**(94.0) | **91.84**(92.0) | **88.15**(86.8) |
| | Last | 90.12(93.4) | **93.78**(93.8) | **91.63**(91.9) | **87.81**(86.6) |
| M-DYR-H(GMM) | Best | 93.49 | 93.26 | 91.0 | 85.99 |
| | Last | 90.34 | 92.84 | 90.8 | 82.3 |

Table 1: Validation accuracy on CIFAR-10 for joint mixup and bootstrapping under BMM(GMM) model. Key: CE(cross-entropy), M(mixup), M-DYR-H(mixup + dynamic bootstrapping + regularization + Hard). The number in the parentheses are the experiment result from Arazo et al. (2019). Bold indicates best performance we re-implemented.

.

| **Alg./Noise level (%)** | | **80** | **85** | **90** |
|---|---|---|---|---|
| MD-DYR-SH(Beta) | Best | **85.65**(82.4) | **83.1**(79.1) | 57.45(69.1) |
| | Last | **84.1**(77.8) | **80.37**(73.9) | 49.73(68.7) |
| MD-DYR-SH(Binomial) | Best | 82.0(82.4) | 79.57(79.1) | **73.09**(69.1) |
| | Last | 74.57(77.8) | 72.21(73.9) | **71.74**(68.7) |

Table 2: Validation accuracy on CIFAR-10 is presented for both dynamic mixup and dynamic bootstrapping under the Beta Mixture Model (BMM). Key: MD-DYR-SH refers to the combination of dynamic mixup, dynamic bootstrapping, soft-to-hard transitions, and regularization. The values in parentheses within each cell represent the results from the original MD-DYR-SH as reported by Arazo et al. (2019), while the values outside the parentheses correspond to the results achieved by our newly proposed method. Bold indicates the best performance we re-implemented.

| Alg./Noise level (%) | | 0 | 20 | 50 | 80 |
|---|---|---|---|---|---|
| M-imagenet | Best | 46.28 | **45.87**(53.2) | **44.13**(41.7) | **37.53**(18.9) |
| | Last | **46.27** | **45.58**(49.4) | **44.13**(31.1) | **37.34**(8.7) |
| M-DYR-H(BMM)-imagenet | Best | **46.73** | 45.10(51.8) | 42.72(44.4) | 35.17(18.3) |
| | Last | 45.29 | 44.44(51.6) | 32.99(43.6) | 15.67(17.7) |

Table 3: Validation accuracy on TinyImageNet for joint mixup and bootstrapping under BMM(Hard) model. Key:M(mixup), M-DYR-H(mixup + dynamic bootstrapping + regularization + Hard). The number in the parentheses are the experiment result from Arazo et al. (2019). Bold indicates best performance we re-implemented.

.

# 6 Conclusions

## 6.1 Experiments Analysis and Methods Strength

1. As shown in Table 1, we successfully re-implemented the main algorithms from Arazo et al. (2019). Notably, the MixUp + Hard Bootstrapping method proved highly effective, significantly improving prediction accuracy under high noise levels.

2. The strength of this algorithm lies in its ability to retain all samples, including noisy ones, thereby preserving more information compared to other methods. Additionally, it operates in an unsupervised manner, eliminating the need to know how samples are mislabeled, as the algorithm can distinguish between noisy and clean samples on its own. Therefore, This method can be easily applied in other settings without requiring prior knowledge of how a clean dataset has been mislabeled.

3. According to Arazo et al. (2019), "Gaussian is a poor approximation to the clean set distribution," though the paper does not provide detailed explanations for why GMM is less effective. To explore this further, we re-implemented the experiments using GMM. As shown in Table 1, GMM indeed performs worse than BMM, particularly during the final epochs. These experimental results support the conclusion presented in Arazo et al. (2019).

4. As shown in Table 2, guided by the intuition from random sampling theories, we modified the MixUp method using BMM and randomly sampled from Beta or Binomial distributions. This adjustment improved the performance of dynamic MixUp under high noise levels, indicating that combining this approach with other methods could further enhance its effectiveness.

## 6.2 Limitations

1. Even though we successfully re-implemented the experiment results, the training process is notably slow. After each epoch, we must run an Expectation-Maximization (EM) algorithm to determine the weights for data mixup, which imposes a significant computational load on the CPU. Consequently, for larger datasets like TinyImageNet, the training process becomes highly time-consuming, and the EM algorithm may face challenges in achieving convergence.

2. The experimental results demonstrate strong performance on the CIFAR-10 dataset, but the method may struggle with datasets containing a large number of classes, such as TinyImageNet and CIFAR-100. Specifically, under high noise levels, the results from Arazo et al. (2019) show a rapid accuracy decline on CIFAR-100 as the noise level increases from 50% to 80%. Additionally, our newly proposed dynamic mixup method appears to perform well only on CIFAR-10. Whether this approach can generalize to other datasets remains uncertain and requires further theoretical analysis and experimental validation on a broader range of datasets.

3. When running the algorithm proposed by Arazo et al. (2019), a warm-up phase of 100 epochs is required to stabilize the model. However, this process is time-consuming and may lead to overfitting. Additionally, for datasets like TinyImageNet, fine-tuning the parameters to strengthen the model becomes necessary. This, however, introduces further computational overhead, as parameter tuning can be highly time-intensive and may fail to converge effectively.

## 7 Contribution by Group Members

Liangqi Tang contributed to the paper by writing the **Problem Statement** section, part of the **Theoretical Foundation and Algorithm of Learning with Label Noise**, the **Experiments** section for TinyImageNet, and the first part of the **Conclusion**. Liangqi was responsible for Git version control, as well as building and modifying the original code for the experiments, debugging, and fine-tuning the parameters. Additionally, Liangqi used a computational server to implement the experiments more efficiently. Moreover, Liangqi conceived the idea of comparing the mean values of Beta distributions to distingtuish noisy and clean distribution and combining random sampling from the Beta distribution with the dynamic Mixup method.

Lingqi Huang contributed to the project by constructing the sections on **Introduction and Motivation**, **Related Work**, part of **Theoretical Foundation and Algorithm of Learning with Label Noise**, and the **Limitations**. He also proposed the **Binomial Dynamic Mixup approach**, implemented the corresponding code, and conducted several experiments on his laptop. Additionally, Lingqi analyzed the experimental results and documented them in the report.

## 8 Acknowledgement

Our project is based on the work in Arazo et al. (2019) and falls within the deep learning field. We built our code upon the basic architecture provided in the paper, developing supplementary functions to meet our specific needs and modifying the code to implement our experiments.

Additionally, we used ChatGPT for minor revisions of our academic writing.

## References


Arazo, E., Ortego, D., Albert, P., O'Connor, N. E., and McGuinness, K. (2019). Unsupervised label noise modeling and loss correction.

Beluch, W. H., Genewein, T., Nurnberger, A., and Kohler, J. M. (2018). The power of ensembles for active learning in image classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9368–9377.

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2016). Deep image homography estimation.

Ding, Y., Wang, L., Fan, D., and Gong, B. (2018). A semi-supervised two-stage approach to learning from noisy labels.

Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M. R., and Huang, D. (2018). Curriculumnet: Weakly supervised learning from large-scale web images.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. (2019). Using trusted data to train deep networks on labels corrupted by severe noise.

Jiang, J., Ma, J., Wang, Z., Chen, C., and Liu, X. (2019). Hyperspectral image classification in the presence of noisy labels. *IEEE Transactions on Geoscience and Remote Sensing*, 57(2):851–865.

Krishna, R., Hata, K., Ren, F., Fei-Fei, L., and Niebles, J. C. (2017). Dense-captioning events in videos. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 706–715.

Little, R. J. A. and Rubin, D. B. (1986). *Statistical analysis with missing data*. John Wiley & Sons, Inc., USA.

Ono, Y., Trulls, E., Fua, P., and Yi, K. M. (2018). Lf-net: Learning local features from images.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2015). Training deep neural networks on noisy labels with bootstrapping.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2019). Learning to reweight examples for robust deep learning.

Tanaka, D., Ikami, D., Yamasaki, T., and Aizawa, K. (2018). Joint optimization framework for learning with noisy labels.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239.

# 9 Appendix

## 9.1 github

Source codes, model parameters and experiment results are available at github link

## 9.2 Dataset Structure

**CIFAR10**:

The CIFAR-10 dataset is a widely used benchmark in computer vision research. It consists of 60,000 color images with a resolution of $32 \times 32$ pixels, divided into 10 classes such as airplanes, cars, and birds. Each class contains 6,000 images, with a training set of 50,000 images and a test set of 10,000 images. The dataset is balanced, with each class having an equal number of samples. The labels are provided in a one-hot encoded format, facilitating classification tasks.

**Tiny ImageNet**:

Tiny ImageNet is a scaled-down version of the larger ImageNet dataset, designed for image classification research. It contains 200 classes, each with 100,000 training images, 10,000 validation images, and 10,000 test images, totaling 120,000 color images. Each color image has a resolution of $64 \times 64$ pixels. The dataset structure includes labeled folders for each class, with corresponding validation and test files organized via a reference mapping file. This dataset poses a more challenging classification problem due to its larger number of classes and lower image resolution compared to CIFAR-10.

## 9.3 Training loop

**Algorithm 1** Training Procedure for Noisy Label Correction

**Initialize:** Learning rate $\eta$, number of epochs $N$, noise level $\lambda$, batch size $B$
Load dataset $\mathcal{D}$ with transformations $\mathcal{T}$
Initialize model $f_\theta$, optimizer $O$, scheduler $S$
Apply noise $\mathcal{N}(\lambda)$ to training labels
**for** each epoch $t = 1$ to $N$ **do**
    Update learning rate $\eta_t \leftarrow S(O, t)$
    **if** Mixup strategy = None **then**
        Minimize loss: $\theta \leftarrow \theta - \eta_t \nabla_\theta \mathcal{L}_{CE}(f_\theta(\mathcal{D}))$
    **else if** Mixup strategy $\in \{\text{Static}, \text{Dynamic}\}$ **then**
        Apply corresponding mixup strategy: $\mathcal{L}_{\text{Mix}}(\theta, \mathcal{D})$
    **end if**
    Track training loss $\mathcal{L}_{\text{track}} \leftarrow \text{Evaluate}(f_\theta, \mathcal{D})$
    Compute validation accuracy: $\text{Acc}_{val} \leftarrow \text{Evaluate}(f_\theta, \mathcal{D}_{val})$
    **if** $\text{Acc}_{val} > \text{BestAcc}$ **then**
        Update best model: $f_{\theta^*} \leftarrow f_\theta$
    **end if**
**end for**

## 9.4 BMM and GMM

We refer to the plot from Arazo et al. (2019) to demonstrate why clean and noisy labels can be separated during the early stages of training.

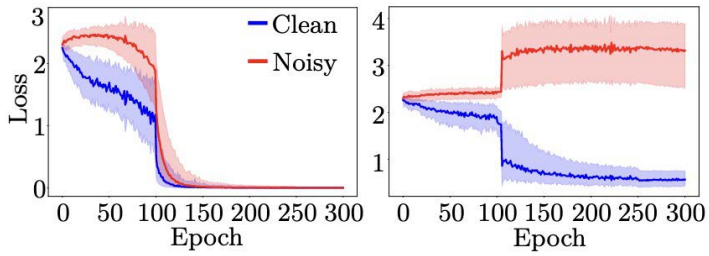


Figure 1: Cross-entropy loss on CIFAR-10 under 80% label noise for clean and noisy samples. Left: training with cross-entropy loss results in fitting the noisy labels. Right: using our proposed objective prevents fitting label noise while also learning from the noisy samples. The heavy lines represent the median losses and the shaded areas are the interquartile ranges.

We reference the plot from Arazo et al. (2019) to illustrate how a mixture model can fit the loss distribution, estimate the posterior probability, and compare the performance of BMM and GMM models.

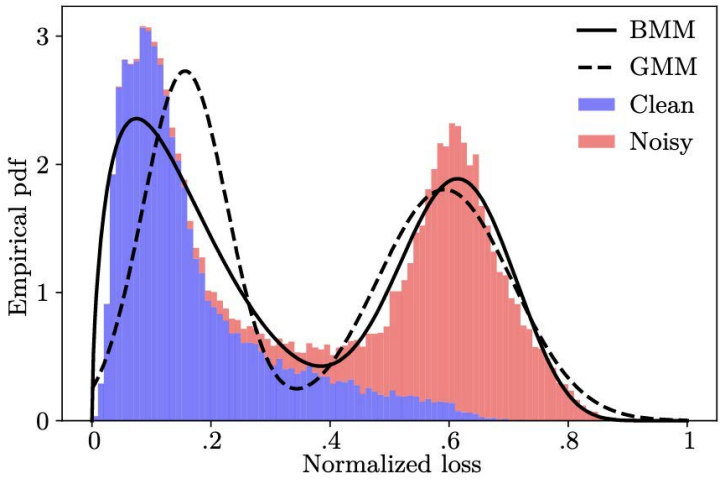


Figure 2: Empirical PDF and estimated GMM and BMM models for 50% label noise in CIFAR-10 after 10 epochs with standard cross-entropy loss and learning rate of 0.1. Clean and noisy samples are colored for illustrative purposes. The BMM model better fits the skew toward zero loss of the noisy samples.

### 9.5 Hyperparameters

#### 9.5.1 CIFAR10

When trying to re-implement the algorithm on CIFAR10, we reference most of the parameters setting from Arazo et al. (2019)

**Preprocessing** Images are normalized and augmented by random horizontal flipping. We use 32×32 random crops after zero padding with 4 pixels on each side.

**Network** A ResNet-18 is trained from scratch. Default PyTorch initialization is used on all layers.

**Optimizer** SGD with momentum (0.9), weight decay of $10^{-4}$, and batch size 128

**Training schedule without mixup** Training for 120 epochs in total. We reduce the initial learning rate (0.1) by a factor of 10 after 30, 80, and 110 epochs. Warm-up for 30 epochs, i.e. bootstrapping (when used) starts in epoch 31.

**Training schedule with mixup** Training for 300epochs in total. We reduce the initial learning rate (0.1) by a factor of 10 after 100 and 250 epochs. Warm-up for 105 epochs, i.e. bootstrapping starts in epoch 106 when used

#### 9.5.2 TinyImageNet

When trying to re-implement the algorithm on TinyImageNet, we slightly modified the parameters setting from Arazo et al. (2019) based on our computational resource

**Preprocessing** Images are normalized and augmented by random horizontal flipping. We use 224×224 random crops without padding.

**Network** A pre-trained Resnet50 is used by transfer learning. Due to the limitation of our computational resources, we froze all the parameters except the last linear layer.

**Optimizer** Adam , weight decay of $10^{-4}$, and batch size 256

**Training schedule without mixup** Training for 120 epochs in total. We reduce the initial learning rate (0.1) by a factor of 10 after 30, 80, and 110 epochs. Warm-up for 30 epochs, i.e. bootstrapping (when used) starts in epoch 31.

**Training schedule with mixup** Training for 100 epochs in total. We reduce the initial learning rate (0.1) by a factor of 10 after 30 and 80 epochs. Warm-up for 35 epochs, i.e. bootstrapping starts in epoch 36 when used.