Noisy CIFAR100

Apetrei Razvan-Emanuel, Nechifor Alexandru

January 2025

1 Abstract

Deep Neural Networks (DNNs) require large datasets for effective training, which must be correctly annotated—a process that is both time-consuming and costly. Beyond the inherent costs, labeling errors introduced by human annotators can significantly impair the training process and performance of DNNs. This paper addresses the challenge of training DNNs on datasets with noisy labels, using CIFAR-100 as the benchmark. Our approach integrates several advanced techniques from existing literature, including modified loss functions and semi-supervised learning, to effectively manage noisy labels and improve model robustness. The experiments on the noisy CIFAR-100 benchmark show significant improvements compared to the baseline method which consists of only using PreActResNet18 and a few transformations.

2 Introduction

Deep Neural Networks (DNNs) have become incredibly successful in recent years, thanks to the availability of large datasets that make training these models possible. However, most datasets are limited and don't cover every real-life scenario, which can make training for specific tasks quite challenging. To overcome this, various techniques have been developed to gather large-scale labeled data, like downloading images from social media or using automated labeling methods. The downside of these methods is that they often introduce incorrect labels, or "noisy labels," which can mess up the training process. Studies have shown that DNNs tend to memorize these noisy labels quickly, which hurts their overall performance.

A different approach to training is semi-supervised learning (SSL), where some of the data is left unlabeled and the algorithm generates pseudo-labels for these samples. SSL can reduce the cost of labeling data by humans, but it comes with its own challenges, like higher computational demands to get the model to converge on these generated labels.

The link between semi-supervised learning and learning with noisy labels (LNL) hasn't been explored much. In this paper, we suggest a method that combines the strengths of both approaches to improve the accuracy of DNNs.

We test our method on a noisy version of the CIFAR-100 dataset. Our approach uses two models that train each other by separating noisy and clean labels, while also applying techniques like label co-refinement and co-guessing to deal with label noise.

The paper is structured as follows: In *Related Work*, we summarize some existing solutions for dealing with noisy data. Next, we explain our method, including how we use two models to work together and handle noisy labels. Finally, we show our results and compare them to other methods, demonstrating how our approach improves performance in image classification tasks.

3 Related Work

Most methods for training Deep Neural Networks (DNNs) with noisy labels focus on modifying the loss function to account for the label noise. These approaches generally fall into two main categories: relabeling noisy samples and re-weighting training samples. The first category includes methods that use techniques like directed graphical models, knowledge graphs, and DNNs to relabel noisy samples. However, these methods typically require access to a small, clean subset of labeled data to guide the relabeling process. Explicit loss correction approaches, such as those using corruption matrices or feature subspaces, also benefit from having some clean data to improve accuracy.

The second category focuses on re-weighting training samples and separating clean and noisy data to adjust the loss function. Studies have shown that DNNs tend to memorize simple features before learning noisy samples. Based on this, it is widely accepted that samples with lower loss values are more likely to be clean. Using this insight, a teacher-student framework is often applied, where a "teacher" DNN assigns weights to samples for a "student" DNN. These weights are typically calculated using a loss function different from the one used by the student.

State-of-the-art semi-supervised learning (SSL) approaches for handling noisy labels often regularize predictions by encouraging consistent and low-entropy outputs for unlabeled samples. This is usually done by adding an additional term to the loss function, which helps to reduce the impact of noisy labels while improving the model's generalization.

4 Method Description

In this section, we describe our approach for designing a noise-robust DNN. The foundation of our method lies in the observation that DNNs tend to learn simple features more quickly than noisy labels, which enables us to effectively separate clean and noisy data. Our method draws inspiration from previous works, particularly the teacher-student framework, but takes a different approach by utilizing Co-Teaching. In Co-Teaching, two models collaborate by preparing data for each other to train on, ensuring that one model learns from

the cleaner subset of data identified by the other.

A key aspect of our approach is the integration of Semi-Supervised Learning (SSL). After dividing the dataset into clean and noisy subsets, we treat the noisy samples as unlabeled data. These samples undergo a labeling process that involves data augmentation and averaging predictions across multiple augmentations to generate an overall label and a confidence probability. These generated labels are then used to train the models further, enhancing their ability to generalize effectively.

Each component of our method, including data separation, the Co-Teaching framework, and the SSL-based labeling process, will be discussed in detail in the following subsections.

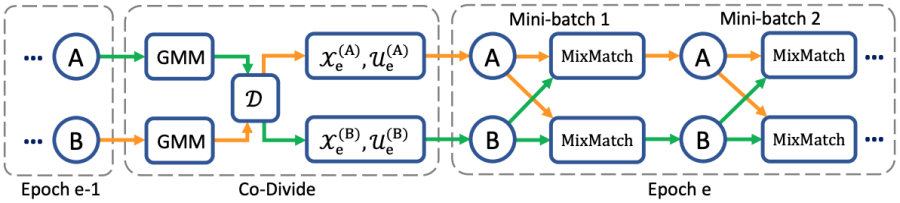

Figure 1: An overview of the method

4.1 Co-Divide by Loss Modeling

As mentioned previously, we utilize two models. Instead of adopting a teacher-student dynamic, we allow the models to teach each other and prepare the data for each other. The primary motivation for this approach is to prevent overfitting to noisy labels, a common issue when relying on a single model, which can become overconfident and biased toward incorrect data. Additionally, this setup enables a cleaner separation of the data. For clean datasets, the commonly used loss function is Cross-Entropy, which minimizes the difference between predictions and the true labels. However, with noisy labels, the loss function may lead to undesirable convergence, focusing on the noise rather than the correct patterns in the data.

To address this, we introduce a *warmup* phase, where we initially assume all data is correct and train both models for a small number of epochs. After the warm up phase, each model outputs probabilities for each sample. We employ a two-component Gaussian Mixture Model (GMM) to separate the data into two subsets. The GMM assigns probabilities to samples based on their loss values, which tend to be smaller for clean data. A threshold τ is then applied to divide the dataset into clean and noisy subsets.

The use of two models is crucial because a single model tends to confirm its own errors, causing noisy samples to gradually appear as clean due to reduced loss values. In contrast, the two models maintain different weights and data distributions, thanks to random initialization and randomized data loading.

This divergence allows the two networks to filter out different types of errors, making the model more robust to noise.

During the warm up phase, the type of noise present in the data plays a significant role. For symmetric (uniformly random) label noise, the Cross-Entropy loss function generally suffices. However, for asymmetric label noise, the network quickly overfits to the noisy labels during warm up, producing overconfident (low entropy) predictions. This results in most samples having near-zero normalized loss, making it difficult for the GMM to distinguish between clean and noisy data. To mitigate this, we introduce a penalization term in the form of a negative entropy, which is added to the loss function. This term, denoted as $\mathcal{H}$, is defined as:

$$\mathcal{H} = -\sum_{c} P^{c}_{\text{model}}(x;\theta) \log\left(P^{c}_{\text{model}}(x;\theta)\right), \tag{1}$$

where:

- $\sum_c$: Summation over all possible classes c.
- $P^{c}_{\text{model}}(x;\theta)$: The probability predicted by the model for class c, given input x and model parameters θ. This probability is typically derived from the softmax function applied to the model's outputs.
- $\log\left(P^{c}_{\text{model}}(x;\theta)\right)$: The natural logarithm of the predicted probability for class c, used to compute the information content of the prediction.

By incorporating this entropy-based penalization term, we ensure that the loss values remain well-distributed, allowing the GMM to better separate clean and noisy data, particularly under conditions of asymmetric noise.

4.2 MixMatch with Label Co-Guessing and Co-Refinement

At each epoch, after dividing the training data, we train the two models one at a time while keeping the other fixed. For each mini-batch, we utilize MixMatch for semi-supervised learning (SSL). MixMatch leverages unlabeled data by combining consistency regularization (i.e., encouraging the model to output consistent predictions for perturbed versions of the same data), entropy minimization (i.e., encouraging confident predictions on unlabeled data), and MixUp augmentation (i.e., encouraging the model to exhibit linear behavior between samples).

To adapt MixMatch for noisy data, we implement the following steps: First, we perform label co-refinement for labeled samples by linearly combining the ground-truth label y_b with the network's prediction p_b (averaged across multiple augmentations of x_b), guided by the clean probability w_b produced by the other network:

$$\bar{y}_b = w_b y_b + (1 - w_b) p_b. \tag{2}$$

Next, we apply a sharpening function to the refined label to reduce its temperature:

$$\hat{y}_b = \text{Sharpen}(\bar{y}_b, T) = \frac{\bar{y}_b^{c\,1/T}}{\sum_{c=1}^{C} \bar{y}_b^{c\,1/T}}, \quad \text{for } c = 1, 2, \ldots, C. \tag{3}$$

Afterward, we use the predictions from both models to obtain more reliable guessed labels for unlabeled samples.

The resulting improved labeled and unlabeled datasets, denoted as $\hat{X}$ and $\hat{U}$, respectively, are then augmented. To "mix" the data, each sample is interpolated with another sample randomly selected from the combined set of $\hat{X}$ and $\hat{U}$. Specifically, for a pair of samples (x_1, x_2) and their corresponding labels (p_1, p_2), the mixed samples (x', p') are computed as follows:

$$\lambda \sim \text{Beta}(\alpha, \alpha), \tag{4}$$
$$\lambda' = \max(\lambda, 1 - \lambda), \tag{5}$$
$$x' = \lambda' x_1 + (1 - \lambda') x_2, \tag{6}$$
$$p' = \lambda' p_1 + (1 - \lambda') p_2. \tag{7}$$

MixMatch transforms $\hat{X}$ and $\hat{U}$ into X' and U'. The loss on X' is calculated as the cross-entropy loss, while the loss on U' is the mean squared error:

$$\mathcal{L}_X = -\frac{1}{|X'|} \sum_{x,p \in X'} \sum_c p_c \log(p^c_{\text{model}}(x; \theta)), \tag{8}$$

$$\mathcal{L}_U = \frac{1}{|U'|} \sum_{x,p \in U'} \|p - p_{\text{model}}(x; \theta)\|^2. \tag{9}$$

Problems arise when the noise level is high, as the network is encouraged to predict the same class to minimize loss. To address this, we introduce a regularization term on the model's average output for the mini-batch, which diversifies the predicted classes:

$$\mathcal{L}_{\text{reg}} = \sum_c \pi_c \log \left(\frac{\pi_c}{\frac{1}{|X'|+|U'|} \sum_{x \in X'+U'} p^c_{\text{model}}(x; \theta)} \right). \tag{10}$$

Finally, the total loss is defined as:

$$\mathcal{L} = \mathcal{L}_X + \lambda_u \mathcal{L}_U + \mathcal{L}_{\text{reg}}. \tag{11}$$

We use λ_u to control the strength of the unsupervised loss.

5 Results

In this section, we describe our experimental setup and present the results of our implementation, comparing its performance against other approaches from the literature. This comparison allows us to evaluate how well the proposed architecture performs relative to existing methods in handling noisy labels.

5.1 Experiment Setup

As mentioned earlier, we used a Noisy CIFAR-100 dataset as a benchmark for comparison. The dataset is available at `http://www.noisylabels.com` and contains finely tuned noise at approximately 40%. This dataset closely resembles the standard CIFAR-100, comprising 50K training samples and 10K testing samples. Noise is evenly distributed among the training instances.

For our two classifiers, we utilized an 18-layer PreAct ResNet architecture and trained it using stochastic gradient descent (SGD) with a momentum of 0.9, a weight decay of 0.0005, and a batch size of 128. The network was trained for 200 epochs, with an initial learning rate of 0.02, which was reduced by a factor of 10 after 100 epochs. The warm-up period lasted for 30 epochs.

We observed that the hyperparameters required minimal tuning. For the level of noise in our dataset, we set the sharpening temperature $T = 0.5$, MixUp parameter $\alpha = 4$, and the threshold $\tau = 0.5$. For the unsupervised loss weight λ_u, the optimal value was determined to be 150.

Method	**CIFAR-100 (40% Noise)**
DivideMix	67.6
Cross-Entropy	46.7
Bootstrap (Reed et al., 2015)	46.6
F-correction (Patrini et al., 2017)	46.6
Co-teaching+ (Yu et al., 2019)	51.8
Mixup (Zhang et al., 2018)	57.3
P-correction (Yi & Wu, 2019)	57.5
Meta-Learning (Li et al., 2019)	59.2
M-correction (Arazo et al., 2019)	66.1

Table 1: Comparison with other methods from the literature in best test accuracy (%) on CIFAR-100 with 40% noise.

As observed, other techniques show weaker results for this dataset, with some performing better on other types of noise or requiring an additional small, clean dataset for guidance. While certain methods demonstrate stronger results, their performance diminishes as the noise percentage increases. In contrast,

the architecture presented in this paper achieves consistently strong results, demonstrating robustness to noise.

6 Ablation Study

In this section, we compare the baseline model (PreActResNet18) and our proposed DivideMix solution. The comparison is based on training and test losses, training and test accuracies, and key performance metrics such as training time, peak GPU memory usage, and inference time.

6.1 Training and Test Loss

Figure 2: Training and Test Loss Over Epochs for Baseline and DivideMix

Figure 2 shows the training and test loss across epochs for both models. DivideMix demonstrates a significantly faster reduction in both training and test losses compared to the baseline, indicating its robustness in handling noisy labels. Notably, DivideMix achieves a lower final test loss, confirming its improved generalization capabilities.

An interesting thing to observe here is the behavior of the test loss for the baseline. We can see that it increases and decreases rather often, most likely

because the training noise contains noise which makes it overfit on that noise increasing its test loss.

6.2 Training and Test Accuracy

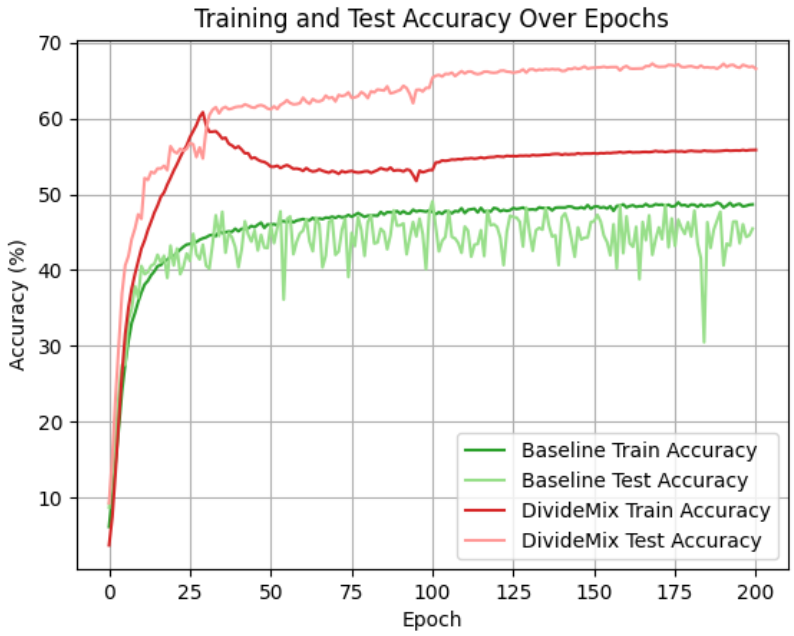

Figure 3: Training and Test Accuracy Over Epochs for Baseline and DivideMix

Figure 3 compares the training and test accuracy of the two models. The DivideMix model outperforms the baseline by a significant margin in test accuracy, particularly in the earlier epochs, showcasing its ability to mitigate overfitting and improve performance on the noisy dataset.

Same as before, if we look at the baseline method of just using PreActResNet18 we can see that it quickly learns the training set then overfits on the noise shown by the erratic behavior of the test accuracy.

6.3 Performance Metrics

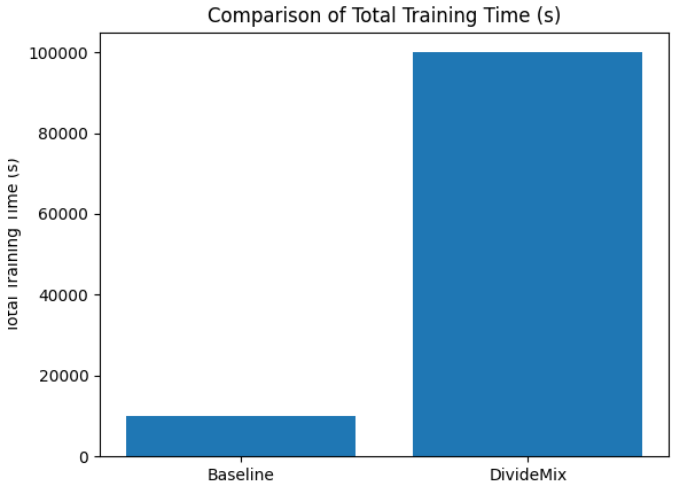

Figure 4: Comparison of Total Training Time

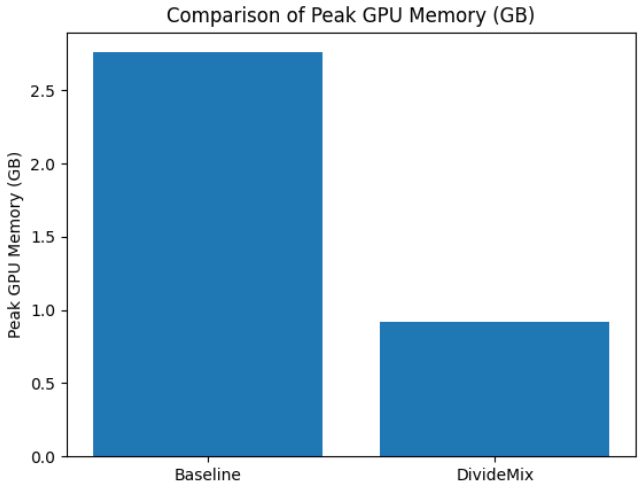

Figure 5: Comparison of Peak GPU Memory Usage

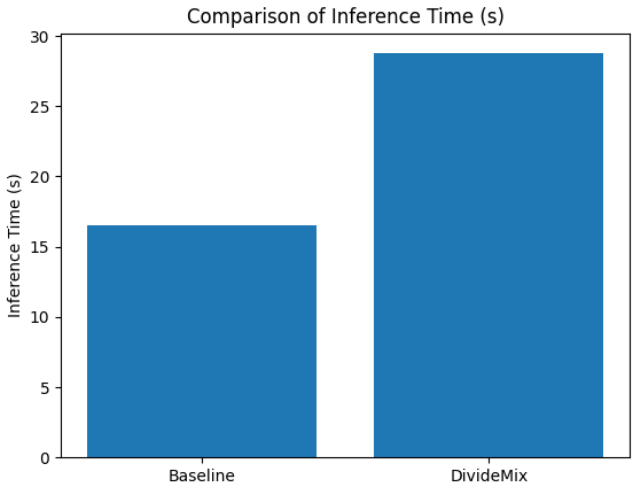

Figure 6: Comparison of Inference Time

Figures 4, 5, and 6 illustrate the comparison of total training time, peak GPU memory usage, and inference time, respectively. DivideMix incurs slightly higher computational costs in terms of training time and GPU memory usage, which is expected given its more complex architecture. However, the increase is reasonable considering the significant improvement in accuracy and robustness. Inference time is also marginally higher for DivideMix but remains within acceptable limits for practical applications.

6.4 Summary

The ablation study highlights that the DivideMix model, leveraging a more sophisticated architecture, consistently outperforms the baseline in terms of accuracy and generalization, albeit with slightly increased computational requirements. These results validate that the usage of DivideMix is a significant improvement over the baseline method.

7 Conclusion

In this paper, we addressed the challenge of training with noisy labels using the Noisy CIFAR-100 dataset as a benchmark. We reviewed existing methods in the literature and their approaches to handling noise. Subsequently, we introduced an architecture that combines label separation with semi-supervised learning to achieve robustness to noise. Additionally, we modified the loss functions to account for both noisy and unlabeled samples.

We presented experimental results comparing our approach to established methods, highlighting its advantages over the conventional Cross-Entropy loss function, which does not account for noise. Furthermore, we analyzed the benefits of our architecture in terms of accuracy, speed, and memory utilization. Our approach demonstrated improved performance over existing methods, showcasing how combining known techniques can lead to better results and provide a novel perspective on dealing with noisy datasets.

8 Bibliography

References

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

[2] Reed, Scott, 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.

[3] Patrini, Giorgio, et al. "Making deep neural networks robust to label noise: A loss correction approach." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. Available: `https://openaccess.thecvf.com/content_cvpr_2017/html/Patrini_Making_Deep_Neural_CVPR_2017_paper.html`

[4] Yu, Xingrui, et al. "How does disagreement help generalization against label corruption?." *International Conference on Machine Learning*, PMLR, 2019. Available: `https://proceedings.mlr.press/v97/yu19b.html`

[5] Zhang, Hongyi. "mixup: Beyond empirical risk minimization." *arXiv preprint arXiv:1710.09412*, 2017. Available: `https://arxiv.org/abs/1710.09412`

[6] Yi, Kun, and Jianxin Wu. "Probabilistic end-to-end noise correction for learning with noisy labels." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. Available: `https://openaccess.thecvf.com/content_CVPR_2019/html/Yi_Probabilistic_End-To-End_Noise_Correction_for_Learning_With_Noisy_Labels_CVPR_2019_paper.html`

[7] Li, Junnan, et al. "Learning to learn from noisy labeled data." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. Available: `https://openaccess.thecvf.com/content_CVPR_2019/html/Li_Learning_to_Learn_From_Noisy_Labeled_Data_CVPR_2019_paper.html`

[8] Arazo, Eric, et al. "Unsupervised label noise modeling and loss correction." *International Conference on Machine Learning*, PMLR, 2019. Available: https://proceedings.mlr.press/v97/arazo19a.html