COMP6248 REPRODUCIBILITY CHALLENGE

DIVIDEMIX: LEARNING WITH NOISY LABELS AS SEMI-SUPERVISED LEARNING

Xiliang He, Zhuoming Liu, Liming Luo, Jinyuan Cai
MSc Artificial Intelligence
University of Southampton
{xh5n20, zl3a20, ll1u20, jc2u20}@soton.ac.uk

1 INTRODUCTION

In the process of learning neural networks, our team members constantly realize the importance of data annotation, but often the data sets we have processed are full of noise, which are easy to overfit in the ordinary neural networks, and the generalization performance is degraded accordingly. After some investigation, we found that this problem stems from the high cost of quality labeling of large amounts of data, so we decided to find a framework that can better deal with noise-labeled data, which makes DivideMix an ideal project for us.

The aim of this report is to reimplement DivideMix framework that reduces the expensive annotation cost in deep neural networks. In the area of reducing the cost of labeling, the traditional solutions tend to minimise the cost of obtaining label information, but doing so inevitably brings along the side effect of noisy labels. As a result, the focus of this area has recently shifted to noise-labeled learning and semi-supervised learning with unlabeled data. Hence, DivideMix was proposed by paper 'DivideMix: Learning with Noisy Labels as Semi-supervised Learning'[3] in ICLR 2020 conference.

Compared with the two other noise-labeled learning methods that have made progress over the same period (LNL and SSL[2], which have made significant advances in loss correction and low entropy prediction in semi-supervised learning, respectively), the results of DivideMix on multiple benchmarks have significantly improved, with two major contributions:

- Two networks were trained simultaneously by fitting Gaussian Mixture Model (GMM) with single sample loss distribution, using co-divide to avoid the possible confirmation bias in self-training.
- MixMatch [1] is implemented with label co-refinement and co-guessing to better deal with noisy labels.

The reimplemented code is available at https://github.com/COMP6248-Reproducability-Challenge/DivideMix_reproducibility.

2 IMPLEMENTATION OVERVIEW

The paper provides a link to share the original code. Due to this code is available and clean, we use the original code to make experiments, and make some refinements to improve its performance and efficiency, which is introduced in the discussion section.

In the first part of the code, the code uses GMM [4] to separate clean and noisy samples due to its flexibility in the sharpness of distribution. If we denote $D = (x, y) = \{(x_i, y_i)\}_{i=1}^{N}$ as the training data and $y_i \in \{0, 1\}^C$ as the one-hot label over C classes, we can use the cross-entropy function below $\ell(\theta)$ to fit sampls.

$$\ell(\theta) = \{\ell_i\}_{i=1}^{N} = \left\{ -\sum_{c=1}^{C} y_i^c \log \left(\mathrm{p}_{\text{model}}^{\mathrm{c}} \ (x_i; \theta) \right) \right\}_{i=1}^{N} \tag{1}$$

N parameters are generated from the GMM model. The distribution with a larger mean belongs to the noisy samples and a distribution wit a smaller mean belongs to the clean samples. Therefore, we

use the EM algorithm to calculate the probability that each distribution belongs to a clean sample, and a threshold (set as 0.5) divides the data into labeled and unlabeled.

If noisy labels are incorrectly classified as clean, it is easy to overfit to the mistake samples and make the predictions on the test set over-confident. Therefore, the paper proposes the key idea - train two neural network simultaneously, and names it co-divided. The two networks teach each other implicitly at each epoch and explicitly at each mini-batch. However, in order to make the model converge faster, several epochs are implemented as warm-up stages so that it is easier to get better initialization parameters.

When it comes to the second module, after splitting the data, the semi-supervised learning method based on Mixmatch is implemented. MixMatch is essentially a framework for data augmentation. The original code use two idea to improve its performance (co-refinement and co-guessing respectively). The co-refinement is a linear combination of the true value and predicted value of the training labels, based on the probability that the predicted label belongs to the clean samples. However, during the experiment, we find that the noise ratio in several data set that we used is low, and the distinguishing effect between noisy samples and clean samples is pretty well. Therefore, the predicted value and true value of training labels is pretty similar so that the co-refinement can hardly affect the result of experiment. In terms of co-guessing, it means that to use the mean of the prediction results of two different networks to predict the unlabeled samples.

Finally, we calculate the loss of labeled data set x^i and unlabeled data set u^i respectively. The loss function $\mathcal{L}_{\mathcal{X}}$ and $\mathcal{L}_{\mathcal{U}}$ are shown bellow:

$$\mathcal{L}_{\mathcal{X}} = -\frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \sum_{c} p_c \log \left(\mathrm{p}^{\mathrm{c}}_{\mathrm{model}} \left(x; \theta \right) \right) \qquad \mathcal{L}_{\mathcal{U}} = \frac{1}{|\mathcal{U}'|} \sum_{x,p \in \mathcal{U}'} \| p - \mathrm{p}_{\mathrm{model}} \left(x; \theta \right) \|_2^2 \tag{2}$$

Considering the model tends to classify all samples as the same class when there are a lot of noise samples, a regular term is added here to help the model to classify the samples into different categories on average. The function is as follows:

$$\mathcal{L}_{\mathrm{reg}} = \sum_{c} \pi_c \log \left(\pi_c / \frac{1}{|\mathcal{X}'| + |\mathcal{U}'|} \sum_{x \in \mathcal{X}' + \mathcal{U}'} \mathrm{p}^{\mathrm{c}}_{\mathrm{model}} \left(x; \theta \right) \right) \tag{3}$$

The complete loss function of the final model is shown bellow, and the π_c is a prior distribution.

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_u \mathcal{L}_{\mathcal{U}} + \lambda_r \mathcal{L}_{\mathrm{reg}} \tag{4}$$

3 RESULTS

In this part, we use the CIFAR-10 and CIFAR-100 image dataset to reproduce DivideMix and set the same hyperparameters proposed in that paper: number of augmentations $M = 2$, sharpening temperature $T = 0.5$, and clean probability threshold $\tau = 0.5$. However, due to limited compute devices, when the number of iterations is set as 300, it would take more than 25 hours per training with a Nvidia RTX 2070, so we decide to reduce the number of iterations to 100 and see how the results differ in accuracy.

3.1 CALCULATE ACCURACY AND AUC

Outputs from the model are evaluated by accuracy and AUC. We predict image samples in the test set and record the accuracy after each iteration in a training. The accuracy curve is shown in the left of Figure 1. Likewise, we record the AUC of labeled samples, and plot the curve shown in the right of Figure 1.

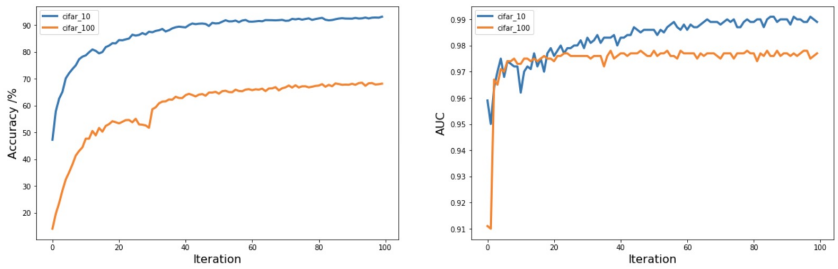

Figure 1: [left] Accuracy curves of predicting test images after each iteration. [right] The AUC of labeled samples after each iteration.

It can be seen from left of Figure 1 that as the number of iterations increases to 80, the prediction accuracy on the test set of CIFAR-10 increases to more than 90%, and that of CIFAR-100 grows to more than 60%, which basically reaches the level of results calculated in the original paper.

In addition, the right image shows that, for the labeled samples of CIFAR-10, the AUC is high and gradually increases, which means that the probability of correctly classifying the labeled samples is extremely high, and some samples that are not correctly labeled can be rectified during training, while there is no continuous AUC increase in CIFAR-100.

3.2 Highly differentiated cross-entropy(CE) Loss

Different from other methods of learning with noisy labels, one of the advantages of DivideMix is that it can easily minimise loss for clean samples while maintaining a larger loss for most noisy samples. The normalized CE loss distribution of clean samples and noise samples for 100 iterations is shown in the Figure 2.

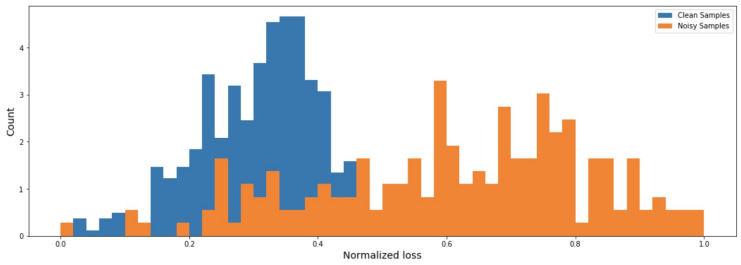

Figure 2: CE loss distribution.

4 Discussion

4.1 A General Review

DivideMix is the first machine learning technique that ”co-divide” the training data into labeled and unlabeled sets by modeling the per-sample loss distribution with Gaussian Mixture Modelling. Combining with MixMatch to train the model in a semi-supervised manner is a smart move, which produces promising results compared with other existing models. We apply DivideMix with some state-of-the-art models on CIFAR-100 with 20% noise for 100 epochs, and it can be observed that DivideMix outperforms other models. DivideMix ends up with 70.8% accuracy - 8.2% larger than that of Mixup and 9.1% larger than that of MetaLearning. To some degree it testifies DivideMix’s good performance according to the author’s implementations in the paper.

However, we find that DivideMix can have a poor performance with less noise (e.g. 5%) and the test accuracy changes very unstably, which should be related to co-divide process when building two GMMs. On the other hand, DivideMix requires to train two models in order to build GMMs and to split the training data into labeled clean data and unlabeled noisy data. By implementing the

model it is proved by us that the warm-up phase is very time consuming. It takes a lot of time on Co-dividing but the payback is very limited.

4.2 CHANGES WE MADE

As DivideMix does not perform well on less noise sample, we propose to apply the equality strategy to the training phase by computing temporary clean probabilities for two outputs, and it turns out to be a good improvement (shown in Fig.3).

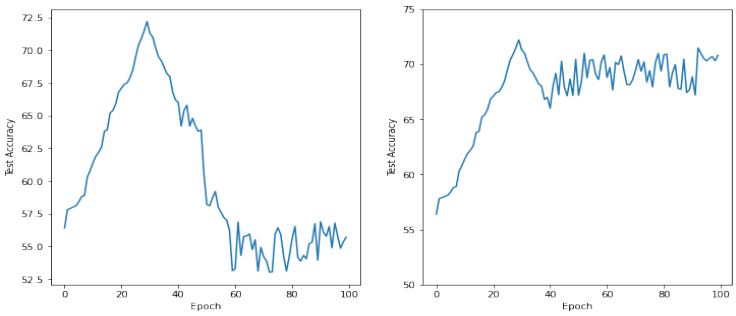

Figure 3: Test accuracy on 5% noise CIFAR-10 before (left) and after (right) making changes

In terms of training time, residual neural network (ResNet) used in the paper is a complex neural network to train and it takes about 8 hours to train on the CIFAR-100 dataset. We wonder whether the networks applied really matter, so we substitute a small convolutional neural network (CNN) for ResNet here. We implement the changed model on CIFAR-100 dataset, and the test accuracy ends up with 70.4, which is very close to the result of the original model. The training time is reduced down to around 6 hours and a half - 18% shorter than the original implementation. We are very glad to see that we can shorten the training time without sacrificing the effectiveness.

5 CONCLUSION

As a novel framework for learning with noisy labels by in a semi-supervised learning manner, DivideMix performs effectively and provides with a new perspective to deal with noisy labels. We successfully carry out this model and reproduce some baseline results and testify the author's original implementation. Since the frameworks does not work well on low noise samples and requires a large amount of computing power, two changes are proposed to improve the framework's performance on its effectiveness and efficiency.

There are some interesting points and further work that can be done for this framework. Through our experiments it is observed that the warm-up stage sometimes turns out to be overfitting and thus influence the training stage. We wonder what causes this and how this can be solved. Besides, since the ResNet required to train in the original paper may not be optimal since we have a overall better performance replacing it with a small CNN, it is worth exploring and testing what networks may work better for the framework and improve the model efficiency.

REFERENCES

[1] David Berthelot et al. "Mixmatch: A holistic approach to semi-supervised learning". In: *arXiv preprint arXiv:1905.02249* (2019).

[2] Yves Grandvalet, Yoshua Bengio, et al. "Semi-supervised learning by entropy minimization." In: *CAP*. 2005, pp. 281–296.

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

[4] Haim Permuter, Joseph Francos, and Ian Jermyn. "A study of Gaussian mixture models of color and texture features for image classification and segmentation". In: *Pattern Recognition* 39.4 (2006), pp. 695–706.