Supplementary Materials: Faster Meta Update Strategy for Noise-Robust Deep Learning

Youjiang Xu[1] Linchao Zhu[2] Lu Jiang[3] Yi Yang[2]
[1]Baidu Research [2]ReLER, University of Technology Sydney [3]Google Research
youjiangxu@gmail.com, lujiang@google.com, {linchao.zhu, yi.yang}@uts.edu.au

A. Derivation of the Meta-Train Step

Recall that $\mathbf{g} = \frac{1}{m}\sum_{j=1}^{m} \nabla_\theta \mathcal{L}_j^{val}(\hat{w}(\theta))$ is the meta gradient for the meta model. Following [8, 9], by using the chain rule, the back-propagation of the Meta-Train can be written as,

$$\mathbf{g} = \frac{1}{m}\sum_{j=1}^{m} \frac{\partial \mathcal{L}_j^{val}(\hat{w}(\theta))}{\partial \hat{w}(\theta)} \sum_{i=1}^{n} \frac{\partial \hat{w}^{(t)}(\theta)}{\partial \mathcal{V}_i^{tra}(\theta)} \frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta}, \tag{1}$$

$$= \frac{-\alpha}{nm}\sum_{j=1}^{m} \frac{\partial \mathcal{L}_j^{val}(\hat{w})}{\partial \hat{w}} \sum_{i=1}^{n} \frac{\partial \mathcal{L}_i^{tra}(w)}{\partial w} \frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta}, \tag{2}$$

$$= \frac{-\alpha}{nm}\sum_{i=1}^{n} \left(\sum_{j=1}^{m} \left(\frac{\partial \mathcal{L}_j^{val}(\hat{w})}{\partial \hat{w}} \right)^{\mathsf{T}} \frac{\partial \mathcal{L}_i^{tra}(w)}{\partial w} \right) \frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta}, \tag{3}$$

Suppose that the network (*e.g.*, a MLP model) has L layers, which is denoted as $\Phi(x; \{w_l\}_{l=1}^{L})$, where w_l represents the parameter for l-th layer. With $w = \{w_l\}_{l=1}^{L}$, we can have:

$$\mathbf{g} = \frac{-\alpha}{nm}\sum_{i=1}^{n} \left(\sum_{j=1}^{m} \sum_{l=1}^{L} \left(\frac{\partial \mathcal{L}_j^{val}(\hat{w})}{\partial \hat{w}_l} \right)^{\mathsf{T}} \frac{\partial \mathcal{L}_i^{tra}(w)}{\partial w_l} \right) \frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta}. \tag{4}$$

For notational convenience, let $G_{i,j,l} = \left(\frac{\partial \mathcal{L}_j^{val}(\hat{w})}{\partial \hat{w}_l} \right)^{\mathsf{T}} \frac{\partial \mathcal{L}_i^{tra}(w)}{\partial w_l}$, hence $\mathbf{g}$ can be formulated as:

$$\mathbf{g} = \frac{-\alpha}{nm}\sum_{i=1}^{n} \left(\sum_{l=1}^{L} \left(\sum_{j=1}^{m} G_{i,j,l} \right) \right) \frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta}, \tag{5}$$

$$= \frac{-\alpha}{nm}\sum_{l=1}^{L} \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{m} G_{i,j,l} \right) \frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta} \right) \tag{6}$$

where α is the learning rate for the network. n and m denote the batch size for training mini-batch and the validation mini-batch.

B. The FaMUS Learning Algorithm

In this section, we first summarize the proposed Faster Meta Update Strategy (FaMUS) in Algorithm 1. Then, we introduce how we use it on the pseudo-clean label set to achieve the-state-of-art performance in Algorithm 2.

Algorithm 1 Faster Meta Update Strategy

```
Input: Training data: $\mathcal{D}^{train} = \{(x_i^{tra}, y_i^{tra})\}_{i=1}^{N}$, validation
    data: $\mathcal{D}^{val} = \{(x_j^{val}, y_j^{val})\}_{j=1}^{M}$, base DNN: $\Phi(\cdot, w)$, meta
    model: $\Psi(\cdot, \theta)$, gradient samplers: $\{\Gamma(\cdot, \eta_l)\}_{l=1}^{L}$, and the
    maximum iteration: $T$
Output: updated base DNN: $\Phi(\cdot, w)$; updated meta model:
    $\Psi(\cdot, \theta)$; updated gradient samplers: $\{\Gamma(\cdot, \eta_l)\}_{l=1}^{L}$
 1: for $t = 0, 1, \ldots, T-1$ do
 2:     draw training mini-batch: $\mathcal{B}^{train}$ from $\mathcal{D}^{train}$
 3:     draw validation mini-batch: $\mathcal{B}^{val}$ from $\mathcal{D}^{val}$
          // 1. Virual-Train step:
 4:     update $\hat{w}$ by Eq.(3)
          // 2. Meta-Train step with FaMUS:
 5:     compute $\mathbf{g}'$ by Eq.(11)
 6:     update $\theta' = \theta - \beta \times \mathbf{g}'$
 7:     for $l = 1, 2, \ldots, L$ do
 8:         update $\eta_l' = \eta_l - \delta \times \nabla_{\eta_l}\mathcal{L}^{val}$
 9:         $\eta_l = \eta_l'$ for the $(t+1)$-th batch
10:     end for
          // 3. Actual-Train step:
11:     update $w'$ by Eq.(5)
12:     $w = w'$, and $\theta = \theta'$ for the $(t+1)$-th batch
13: end for
```

Algorithm 1 shows MW-Net [9] with the proposed FaMUS step. We use MW-Net as an example and it is straightforward to extend Algorithm 1 to the other meta-learning models such as L2R [8] and MLC [12] by replacing their original Meta-Train step with the proposed FaMUS. δ is the learning rate for the gradient samplers.

Algorithm 2 shows the proposed FaMUS applying on the pseudo-validation set. Particularly, we train two base DNNs and two GMMs, where the GMM for one base DNN is employed to get a pseudo-clean label set, which will be used as the meta-learning validation set for the other base DNN. Please refer to Section E for how we construct the noisy meta-learning validation set. Note that when compared with previous meta-learning methods [8, 9, 12], our method does

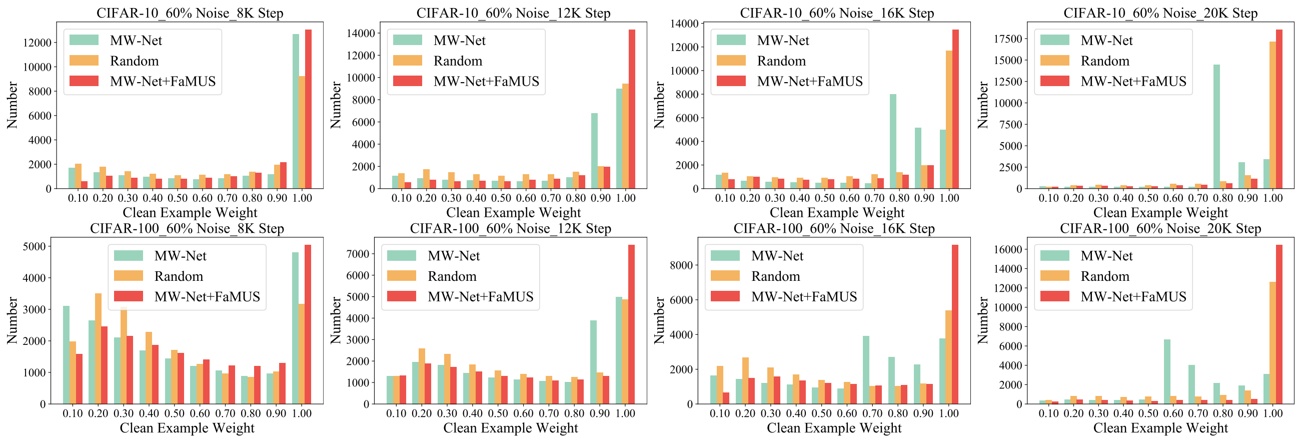
Figure 1: The weight distribution of clean examples on CIFAR-10 and CIFAR-100, both with 60% noise. We illustrate the distribution of four different training steps: $8K$, $12K$, $16K$, and $20K$. "Random" means we uniformly select 4 layers of meta gradients to approximate the total meta gradient for the meta model.

Algorithm 2 FaMUS with the Pseudo-clean Label Set

```
Input: Training data $\mathcal{D}^{train} = \{(x_i^{tra}, y_i^{tra})\}_{i=1}^N$
Output: Base DNNs: $\Phi(\cdot; w^1)$ and $\Phi(\cdot; w^2)$; Meta model: $\Psi(\cdot;\theta^1)$ and $\Psi(\cdot;\theta^2)$; gradient samplers: $\Gamma_l(\cdot;\eta_l^1)$ and $\Gamma_l(\cdot;\eta_l^2)$, where $l \in [1, L]$
1: for $e = 0, 1, \ldots, E-1$ do // separate the training data into a pseudo-clean and a pseudo-noisy label set:
2:     $\mathcal{D}_1^{clean}$, $\mathcal{D}_1^{noisy}$ = GMM($\mathcal{D}^{train}$, $\Phi(\cdot; w^1)$)
3:     $\mathcal{D}_2^{clean}$, $\mathcal{D}_2^{noisy}$ = GMM($\mathcal{D}^{train}$, $\Phi(\cdot; w^2)$)
       // using Alg. 1 to train the base DNN, the meta model, and gradient samplers
4:     $w^1$, $\theta^1$, $\{\eta_l^1\}_{l=1}^L \leftarrow$ Alg. 1($\mathcal{D}^{train}$, $\mathcal{D}_2^{clean}$, $w^1$, $\theta^1$, $\{\eta_l^1\}_{l=1}^L$)
5:     $w^2$, $\theta^2$, $\{\eta_l^2\}_{l=1}^L \leftarrow$ Alg. 1($\mathcal{D}^{train}$, $\mathcal{D}_1^{clean}$, $w^2$, $\theta^2$, $\{\eta_l^2\}_{l=1}^L$)
6: end for
```

not use any clean labels or data.

C. Weight Distribution of Clean Examples

Figure 1 shows the weight distribution of all clean training examples on both CIFAR-10 and CIFAR-100 with 60% noise. We show the weight distributions on four different training steps: $8K$, $12K$, $16K$, and $20K$. We compare our method with two strong baselines: the original MW-Net [9] and Random MW-Net. "Random MW-Net" means we uniformly select four layers of meta gradients to approximate the total meta gradient to update the meta model. As shown, compared with the original MW-Net [9] and Random MW-Net, the meta model learned by our FaMUS tends to give larger weights to the clean examples throughout the training process. The result substantiates that our method is able to improve the capability of the meta-learning methods (*e.g.*, MW-Net).

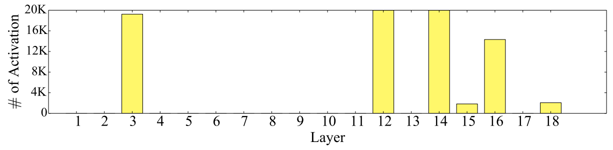
Figure 2: Statistics of the sampled layers by FaMUS on the CIFAR-100 with 60% noise.

D. Visualization of the Sampled Layers

Figure 2 shows the statistics of the selected layers generated by FaMUS. The model is trained on CIFAR-100 with 60% noise. From Figure 2, we find that FaMUS automatically learns to pick a few specific layers, and in particular top layers are more favorable.

E. Implementation Details

In this section, we first detail how we construct the meta-learning validation set. Then we discuss the training details for the comparison with the state-of-the-art and the meta-learning methods. Finally, we describe the training details about the experiments on the long-tailed classification.

Details of meta-learning validation set. Inspired by [5], we employ a two-component GMM to separate the training data into a pseudo-clean label set and a pseudo-noisy label set. Then, the pseudo-clean label set is employed as the meta-learning validation set to train the meta model. To be more specific, for the i-th example, the input to GMM is its training loss $\mathcal{L}_i^{tra}(w)$, while the output of GMM is the corresponding clean probability scale c_i and $c_i \in [0, 1]$. The clean probability c_i is the posterior probability $p(q|\mathcal{L}_i^{tra}(w))$, where q is the Gaussian component with the smaller mean. Note that the GMM can be updated by the Expectation-Maximization algorithm by taking the loss distribution of all training examples $\{\mathcal{L}_i^{tra}(w)\}_{i=1}^n$ as input. We set the maximum iteration number to 10, the convergence threshold to 1e-2, and the number of components to 2.

Training details for comparing with the state-of-the-art. For FaMUS, the learning rate δ is fixed to 1e-1 throughout the training. λ_1 and λ_2 are searched from $\{0, 0.1\}$. K is set to 4 for all experiments. The learning rate β for the meta model is fixed to 1e-3. The training details for the base DNNs are listed below:

CIFAR-10 and CIFAR-100. Following previous work [5], we employ the PreAct ResNet-18 [3] as the base DNN, which is optimized by SGD, with a weight decay of 5e-4 and a momentum of 0.9. The batch size is set to 64. The learning rate α starts at 0.02, decreases to its $\frac{1}{10}$ at the 200 epoch and the 250 epoch, and stops at the 300 epoch.

WebVision. We employ the Inception-ResNet V2 [10] as the base DNN. The batch size is set to 16. The learning rate α starts at 0.02, decreases to its $\frac{1}{10}$ at the 100 epoch, and stops at the 150 epoch. The base DNN is optimized by SGD, with a weight decay 5e-4 and a momentum 0.9.

Clothing1M. We employ the ResNet-50 [2] as the base DNN, which is optimized by SGD, with a momentum of 0.9 and a weight decay of 1e-3. The batch size is set to 32. The learning rate α starts at 0.002, decreases to its $\frac{1}{10}$ at the 100 epoch, and stops at the 150 epoch. For each epoch, we randomly sample 1000 mini-batch from training data while maintaining the label noise balanced.

Controlled Noisy Web Labels (CNWL). We download the dataset from their website [1]. We employ the PreAct ResNet-18 [3] as the base DNN. The batch size is set to 64. The learning rate α starts at 0.02, decreases to its $\frac{1}{10}$ at the 200 epoch and the 250 epoch, and stops at the 300 epoch. The base DNN is optimized by SGD with a momentum of 0.9 and a weight decay of 5e-4. For the baseline models, we run the MentorMix [4][2], Mixup [16][3], DivideMix [5][4] using their official implementations on the CIFAR datasets. Note that in order to use their implementations, we downsample the images of the CNWL Mini-ImageNet dataset from 84x84 to 32x32. This results in new benchmark numbers to compare our baseline methods, and supplements [4]'s results on 32x32 images.

Training details for comparing with meta-learning methods. We implement our FaMUS with three meta-learning based approaches: L2R [8][5], MW-Net [9][6], and MLC [12][7]. All experiments are conducted on the identical hardware platform of one NVIDIA V100 GPU. For L2R, we use WideResNet-28-10 (WRN-28-10) [15] as the base DNN. The batch size is set to 100. The learning rate α starts from 0.05, decreases to its $\frac{1}{10}$ at the 40K and 50K iteration, and stops at 60K iteration. The learning rate β for the meta model is fixed to 1e-3. For MW-Net [9], we employ WRN-28-10 as the base DNN. The batch size is set to 100. The learning rate starts at 0.05, decreases to its $\frac{1}{10}$ at the 18K and the 19K iteration, stops at the 20K iteration. We train the network using SGD with a momentum of 0.9 and a weight decay of 5e-4. The learning rate β for the meta model is fixed to 1e-3. For MLC [12], we use the official code. Particularly, we use the Wide ResNet of depth 40 and widening factor 2 (WRN-40-2) [15]. The batch size is set as 64. We train the network using SGD optimizer with a learning rate of 1e-4, a momentum of 0.9, and a weight decay of 5e-4. The learning rate β for the meta model is fixed to 1e-5.

Method	Accuracy (%)
F-correction [7]	69.84
JoCoR [13]	70.30
M-correction [1]	71.00
MLC [12]	71.06
Joint-Optim [11]	72.16
MLNT [6]	73.47
P-correction [14]	73.49
MW-Net [9]	73.72
MentorMix [4]	74.30
Ours	**74.43**

Table 1: Top-1 Accuracy on Clothing1M.

Training details for the long-tailed classification. For experiments on long-tail CIFAR-10 and CIFAR-100, we use ResNet-32 as base DNN, which is optimized by SGD with a weight decay of 5e-4 and a momentum of 0.9. The learning rate α starts at 0.1, decreases to its $\frac{1}{10}$ at the 160 epoch and 180 epoch, stops at the 200 epoch. As for the meta model, the learning rate β is fixed to 1e-5. As for FaMUS, the learning rate δ is fixed to 1e-2. λ_1 and λ_2 are searched from $\{0, 0.1\}$. K is set to 4.

F. Results on Clothing1M

Table 1 compares the state-of-the-art on Clothing1M dataset. From Table 1, we can find that our method is

[1]https://google.github.io/controlled-noisy-web-labels

[2]https://github.com/google-research/google-research/tree/master/mentormix

[3]https://github.com/facebookresearch/mixup-cifar10

[4]https://github.com/LiJunnan1992/DivideMix

[5]https://github.com/uber-research/learning-to-reweight-examples

[6]https://github.com/xjtushujun/Meta-weight-net

[7]https://github.com/ZhenWang-PhD/Training-Noise-Robust-Deep-Neural-Networks-via-Meta-Learning

Method	CIFAR-10		CIFAR-100	
	40%	60%	40%	60%
DivideMix	95.06 ±0.14	94.46 ±0.21	75.15 ±0.12	71.41 ±0.61
MW-Net + Pseudo-clean	93.64 ±0.35	92.91 ±0.15	74.61 ±0.25	70.55 ±0.57
Ours	95.37 ±0.15	94.97 ±0.11	75.91 ±0.19	73.58 ±0.28

Table 2: Results on CIFAR-10 and CIFAR-100. "Pseudo-clean" refers to the pseudo-clean label set constructed by GMMs. Percentage numbers denote the noise rate.

competitive with recently published methods. In particular, our method outperforms several meta-learning approaches [9, 6, 12]. Note that these meta-learning methods are learned with clean validation examples, while our method is trained by the pseudo-clean label set generated by GMMs. Compared with the recently proposed method (*e.g.*, MentorMix [4]), our method gets a comparable performance.

G. More Experimental Results

Table 2 compares our method with two strong baselines: DivideMix [5] (reported in the main paper) along with another method called "MW-Net + Pseudo-clean" which trains the MW-Net model with the same pseudo-clean label set constructed by GMMs as ours. Since all the methods are trained on the pseudo-clean label set, this experiment is to examine the contribution of our methodology.

We report the average and the standard deviation of over three training trials using different random seeds for each method. Table 2 shows that our method outperforms all baseline methods with a statistically significant difference at the p-value level of 0.05, according to the one-tailed unpaired t-test. These results substantiate the superior performance stems from the proposed method as opposed to the pseudo-clean label set.

References

[1] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *ICML*, 2019. 3

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 3

[4] Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *ICML*, 2020. 3, 4

[5] Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *ICLR*, 2020. 2, 3, 4

[6] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, 2019. 3, 4

[7] 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 *CVPR*, 2017. 3

[8] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018. 1, 3

[9] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019. 1, 2, 3, 4

[10] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2016. 3

[11] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018. 3

[12] Zhen Wang, Guosheng Hu, and Qinghua Hu. Training noise-robust deep neural networks via meta-learning. In *CVPR*, 2020. 1, 3, 4

[13] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In *CVPR*, 2020. 3

[14] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *CVPR*, 2019. 3

[15] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMCV*, 2016. 3

[16] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 3