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

Abstract

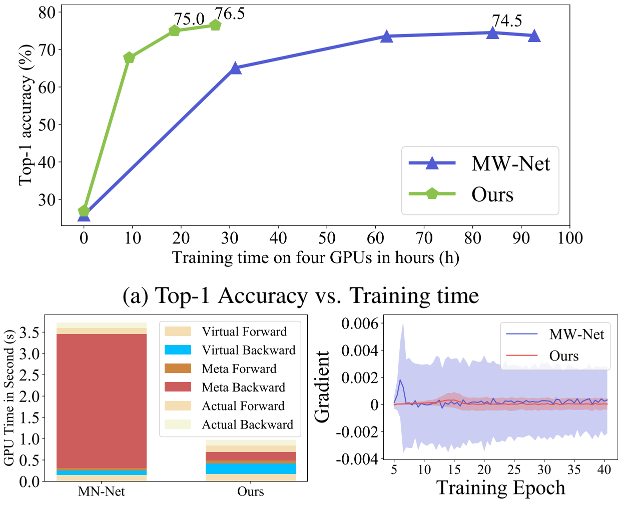
It has been shown that deep neural networks are prone to overfitting on biased training data. Towards addressing this issue, meta-learning employs a meta model for correcting the training bias. Despite the promising performances, super slow training is currently the bottleneck in the meta learning approaches. In this paper, we introduce a novel Faster Meta Update Strategy (FaMUS) to replace the most expensive step in the meta gradient computation with a faster layer-wise approximation. We empirically find that FaMUS yields not only a reasonably accurate but also a low-variance approximation of the meta gradient. We conduct extensive experiments to verify the proposed method on two tasks. We show our method is able to save two-thirds of the training time while still maintaining the comparable or achieving even better generalization performance. In particular, our method achieves the state-of-the-art performance on both synthetic and realistic noisy labels, and obtains promising performance on long-tailed recognition on standard benchmarks. Code are released at https://github.com/youjiangxu/FaMUS.

1. Introduction

Deep neural networks (DNNs) have achieved impressive results in various computer vision applications such as image classification [23, 13], object detection [41, 39, 29], and semantic segmentation [12]. A notable issue is that DNNs are prone to memorizing the training data [60, 47], aggravating training set bias such as noisy training labels [60] or imbalanced class distributions [11, 64]. This significantly degrades the generalization capabilities and results in skewed classifiers or degenerated feature representations.

Numerous works have been proposed to tackle this issue (*e.g*. [20, 9, 40, 24, 28]). Among them, meta-learning [40, 43, 51] has recently emerged as an effective framework to mitigate the training data bias. In a nutshell, it employs a meta-model to correct bias by providing a more precise estimation of the training data. The meta-model is updated by stochastic gradient descent using the *meta gradient* (or the high-order gradient) computed on a small proportion of validation data that is assumed available during training[1]. Recently, meta-learning approaches such as L2R [40], MW-Net [43], and MLC [51] have shown superior performance on several public benchmarks such as CIFAR [22], WebVision [26], and Clothing1M [54].

Despite the promising empirical results [50, 44], slow training is currently the bottleneck that prevents meta-learning from being applied in many applications. The training time of the meta-learning model is approximately



(a) Top-1 Accuracy vs. Training time

(b) Cost in One Iteration

(c) Gradient Variance

Figure 1: (a) Top-1 accuracy vs. Training time (in hours) on the WebVision dataset [26]. We apply our method on the MW-Net model [43] and train them using the identical hardware platform of four NVIDIA V100 GPUs. (b) The average GPU running time (in seconds) of each step in MW-Net per training iteration. Inception-ResNet V2 is used as the backbone. (c) The meta gradient during the training process. The solid line denotes the mean and the shaded region show the standard deviation.

[1] The extra validation dataset is not a requirement in meta-learning. As in our experiments, we can use a subset of pseudo-labeled training data as the validation data. In this case, no extra labels or data are used.

3∼7 times more than the regular DNN training time. For instance, it could take 4 days with 4 NVIDIA V100 GPUs to train MW-Net [43] on a mini subset of WebVision [26, 20] of only ∼50K images.

To understand why the meta-learning approaches are computationally intensive, we may divide the training into three stages: Virtual-Train, Meta-Train, and Actual-Train [51], where each stage consists of a forward and a backward step. Figure 1(b) summarizes the GPU time for each stage using a representative meta-learning model called MW-Net [43]. We find more than 80% of the total computation comes from the Meta-Train backward step in which the *meta gradient* is computed with respect to the loss on the validation data. In this step, the meta gradient is back-propagated through every layer of the network all the way back to the meta-model to update its parameters. Since the regular training does not have such a step, this overhead cost rapidly becomes significant as the number of layers grows in the deep networks.

In this work, we aim at improving the training efficiency of meta-learning while maintaining the generalization capability. We propose a new Meta-Train step, named Faster Meta Update Strategy (FaMUS), to efficiently compute the meta gradient. The plausibility of our method relies on the important finding that the total meta gradient can be reasonably approximated by the meta gradient accumulated from only a few network layers. As a result, instead of accumulating meta gradients from all layers in the Meta-Train step, we design a gradient sampler that is learned to decide, whether or not, to aggregate the meta gradient for each layer. When the learnable gradient sampler is turned off, the meta gradient computation is hence circumvented for the corresponding layer. This saves a considerable amount of computation especially when the gradient samplers for lower layers are turned off.

More importantly, we find the meta gradient yielded by the FaMUS has lower variance. Figure 1(c) shows the total meta gradient of the ground-truth (blue curve) and the approximation by the FaMUS (red curve). It shows that our approximation is reasonably close to the mean but has a much lower variance. We hypothesize this is because the FaMUS learns to select a small number of most informative layers which hence reduces the noisy or redundant signals in the meta gradient. As shown in [35, 34], reduction in gradient variance results in faster and more stable optimization. We observe similar results in our experiments where our method is able to improve the generalization performance of the recent meta-learning methods on noisy training data.

We conduct extensive experiments to verify the efficiency and efficacy of the proposed method. We demonstrate two benefits of our method in overcoming corrupted training labels. First, it speeds up the recent meta-learning methods [40, 43, 51] by at least three times while maintaining the comparable or even better generalization performance. For example, Figure 1(a) shows a faster and better convergence when we applied our method on the MW-Net model. Second, our method achieves new state-of-the-art performance on multiple benchmarks for both synthetic label noise and realistic label noise, including the challenging CNWL benchmark [18]. The comparison is fair as our meta-model is learned without using any extra data. In addition, we also validate our method on the long-tailed recognition task. On the long-tailed CIFAR dataset [6], our method yields competitive performance compared to the recent strong baseline methods.

The contributions of this paper are three-fold. (1) We propose a new Faster Meta Update Strategy to efficiently learn to approximate the meta gradient, which halves two-thirds of the training time of the recent meta-learning methods [40, 43, 51]. (2) We empirically show our approach reduces the variance of the meta gradient and improves the generalization performance of the meta-learning model. (3) Our method achieves state-of-the-art performance on several benchmarks with noisy labels.

2. Related Work

Corrupted/Noisy training labels. Numerous methods have been recently proposed to learn robust deep networks that can overcome corrupted or noisy training labels. These methods address this problem from a variety of directions. For example, several works [8, 37, 53, 47, 56, 1, 51] modeled the noise distribution or the transition matrix to correct noisy training samples. Other approaches tried to reduce the weights assigned to noisy samples [33, 20, 18, 45, 43, 55]. Another effective strategy is to directly identify the clean samples and only select them to train the models [9, 58, 36, 38, 24, 52]. Other contributions in this direction include data augmentation [61, 27, 5], semi-supervised learning [15, 48, 24, 62], *etc.*

Among them, meta-learning [40, 43, 25, 51] has recently emerged as an effective framework for addressing the noisy labels. These methods all learn a meta-model from clean validation examples but differ in the specific ways to correct the biased training labels. For example, L2R [40] directly adjusts the weight for each example. MLNT [25] simulates regular training with synthetic noisy labels. MW-Net [43] learns an explicit weighting function. MLC [51] estimates the noise transition matrix.

This paper aims at improving the training efficiency of the meta-learning models. The results show our method not only significantly reduces the training time of three recent meta-learning approaches but also improves their robustness to noisy labels on several standard benchmarks.

Long-tailed recognition. Long-tailed recognition has been an active research field in computer vision [3, 7, 10, 42, 32, 57, 30, 21, 6, 2, 16, 63, 64]. For example, [3, 10] aimed to increase the number of minority classes by oversampling, while Drummond *et al.* [7] solved this problem by reducing the number of data in majority classes. Some recent studies [42, 32] proposed to balance the number of data for each class. [57, 30] applied the knowledge learned from the head classes to the tail. [21, 6, 2, 63] aimed to manipulate the loss on the class-level based on the data distribution.

Meta-learning based methods [40, 43, 16] have recently achieved promising results on the long-tailed recognition task, in which the meta-model is learned to assign larger weights to the examples of the long-tailed classes. Similar to the noisy labels, meta-learning suffers from slow training speed [40, 43, 51, 16]. We show our method improves the efficiency and accuracy of the meta-learning methods on the long-tailed recognition task, and achieves competitive performance compared with recent strong baselines.

3. Preliminary on Meta-learning

In this section, we briefly introduce the preliminary on meta-learning methods [40, 43] that learn robust deep neural networks from noisy labels by reweighting the training data. We follow the notation in the MW-Net [43] model using corrupted labels as an example. Alternative formulation can be found in [40, 51, 50].

Let $\mathcal{D}^{train} = \{(x_i^{tra}, y_i^{tra})\}_{i=1}^N$ be a noisy training set of N examples, where x_i^{tra} is the i-th training image and $y_i^{tra} \in \{0,1\}^c$ is its one-hot label over c classes. Consider a deep neural network (DNN) as the base model $\Phi(\cdot; w)$ with w denoting its parameters. Generally, we can derive the optimal parameter w^* by minimizing the softmax cross-entropy loss $\ell(\hat{y}, y)$ over the training data, where $\hat{y} = \Phi(x; w)$ is the prediction of the DNN and y is the given label for the input image x.

In the meta-learning methods [40, 43], there is an out-of-sample validation set $\mathcal{D}^{val} = \{(x_j^{val}, y_j^{val})\}_{j=1}^M$, where (x_j^{val}, y_j^{val}) denote the j-th example. M is the size of $\mathcal{D}^{val}$ and $M \ll N$. The extra validation dataset is not always required in meta-learning. See the discussion in Section 5.3.

The meta-learning method employs a meta-model (*e.g.*, instanced by a multilayer perceptron network (MLP) with only one hidden layer [43]) to learn a weight for each training example. Let $\Psi(\cdot; \theta)$ denote the meta-model, parametrized by θ, which maps a loss to a weight scalar. A meta-model can be regarded as a learnable derivation of the self-paced function in SPCL [19]. Let $\mathcal{L}_i^{tra}(w) = \ell(\Phi(x_i^{tra}; w), y_i^{tra})$ be the loss for the i-th example in $\mathcal{D}^{train}$. The optimal parameter w^* can be obtained by computing the weighted loss:

$$w^*(\theta) = \operatorname*{argmin}_{w} \frac{1}{N} \sum_{i=1}^{N} \mathcal{V}_i^{tra}(\theta) \mathcal{L}_i^{tra}(w), \tag{1}$$

where $\mathcal{V}_i^{tra}(\theta) = \Psi(\mathcal{L}_i^{tra}(w); \theta)$ is the generated weight for the i-th training example.

The meta-model is optimized by minimizing the validation loss:

$$\theta^* = \operatorname*{argmin}_{\theta} \frac{1}{M} \sum_{j=1}^{M} \mathcal{L}_j^{val}(w^*(\theta)), \tag{2}$$

where $\mathcal{L}_j^{val}(w^*(\theta)) = \ell(\Phi(x_j^{val}; w^*(\theta)), y_j^{val})$ is the loss for the j-th example in the validation set.

Solving Eq. (1) and Eq. (2) by alternating minimization is intractable for mini-batch gradient descent. Alternatively, an online optimization method is used instead which comprises three steps: Virtual-Train, Meta-Train, and Actual-Train [52].

Consider the t-th iteration. Given a training mini-batch $\mathcal{B}^{train} = \{(x_i^{tra}, y_i^{tra})\}_{i=1}^n$ and a validation mini-batch $\mathcal{B}^{val} = \{(x_j^{val}, y_j^{val})\}_{j=1}^m$, n and m stand for the number of the examples in the mini-batch. For the Virtual-Train, an one-step "virtually" updated DNN can be derived by:

$$\hat{w}(\theta) = w - \alpha \frac{1}{n} \sum_{i=1}^{n} \mathcal{V}_i^{tra}(\theta) \nabla_w \mathcal{L}_i^{tra}(w), \tag{3}$$

where α is the learning rate for the DNN. w is the parameter of the base DNN at the current iteration. This step is called Virtual-Train because $\hat{w}(\theta)$ will not be used to update the parameter of the base DNN.

Then for the Meta-Train, with the latest $\hat{w}(\theta)$, the meta-model is updated by:

$$\theta' = \theta - \beta \frac{1}{m} \sum_{j=1}^{m} \nabla_\theta \mathcal{L}_j^{val}(\hat{w}(\theta)). \tag{4}$$

Similarly, β is the learning rate for the meta-model. θ' is the parameter of the updated meta-model. Notice that $\frac{1}{m}\sum_{j=1}^{m} \nabla_\theta \mathcal{L}_j^{val}(\hat{w}(\theta))$ is called meta gradient, which is expensive to compute. More details will be discussed in Section 4.1.

Finally, in the last step (Actual-Train), the updated meta-model $\Psi(\cdot; \theta')$ is used to update the base DNN model using:

$$w' = w - \alpha \frac{1}{n} \sum_{i=1}^{n} \mathcal{V}_i^{tra}(\theta') \nabla_w \mathcal{L}_i^{tra}(w), \tag{5}$$

where $\mathcal{V}_i^{tra}(\theta')$ is the weight for the i-th example computed by the latest meta-model. This step is called Actual-Train because w' will be used to actually update the parameter of base DNN. Therefore, w' becomes the w in Eq. (3) in the $(t+1)$-th iteration.

4. Faster Meta Update Strategy

In this section, we introduce a Faster Meta Update Strategy (FaMUS) to efficiently approximate the total meta gradients by a layer-wise meta gradient sampling procedure.

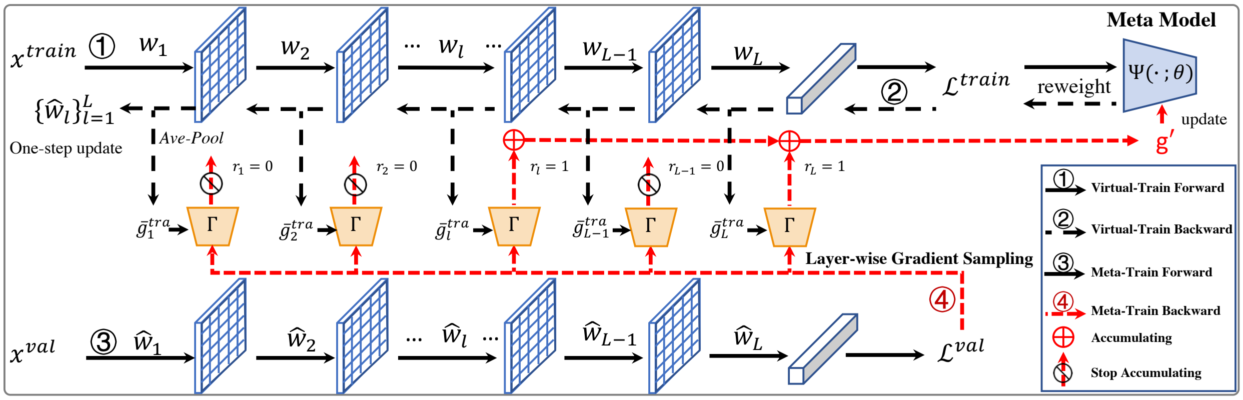


Figure 2: Illustration of the proposed method. We propose a new Meta-Train step, named Faster Meta Update Strategy (*i.e.*, the red line ④), which learns a gradient sampler (denoted as Γ) to aggregate the meta gradient for each layer. In this figure, the meta gradients from the l-th and L-th layers would be aggregated to compute $\mathbf{g}'$ and used to update the meta-model Ψ.

Figure 2 presents the overall training process, where the red line indicates the proposed method. Specifically, we learn a gradient sampler to decide, whether or not, to aggregate the meta gradient for each layer. In the following, we first explain how the meta gradient can be calculated in a layer-wise fashion in Section 4.1. Next, we detail the gradient sampler in Section 4.2 and the final objective for the meta-model in Section 4.3. The full algorithm for Faster Meta Update Strategy is shown in the supplementary materials.

4.1. Layer-wise meta gradient computation

In this section, we discuss the meta gradient computation and show how it can be calculated in a layer-wise fashion. For notational convenience, we simplify Eq. (4) as:

$$\theta' = \theta - \beta \times \mathbf{g}, \tag{6}$$

where $\mathbf{g} = \frac{1}{m}\sum_{j=1}^{m}\nabla_\theta \mathcal{L}_j^{val}(\hat{w}(\theta))$ denotes the meta gradient, which has shown to be computational intensive in recent studies [40, 43, 51].

Without loss of generality, suppose that the base DNN has L layers denoted as $\Phi(\cdot;\{w_l\}_{l=1}^{L})$, where w_l represents the parameter for the l-th layer.

We rewrite the computation of meta gradient using the chain rule:

$$\begin{aligned}\mathbf{g} &= \frac{1}{m}\sum_{j=1}^{m}\frac{\partial \mathcal{L}_j^{val}(\hat{w}(\theta))}{\partial \hat{w}(\theta)}\sum_{i=1}^{n}\frac{\partial \hat{w}(\theta)}{\partial \mathcal{V}_i^{tra}(\theta)}\frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta} \\ &\propto \frac{-\alpha}{nm}\sum_{l=1}^{L}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{m}G_{i,j,l}\right)\frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta}\right),\end{aligned} \tag{7}$$

where $G_{i,j,l} = (\frac{\partial \mathcal{L}_j^{val}(\hat{w})}{\partial \hat{w}_l})^\top \frac{\partial \mathcal{L}_i^{tra}(w)}{\partial w_l}$ is the dot product between the gradient from the j-th validation loss w.r.t. $\hat{w}_l$ and the gradient from the i-th training loss w.r.t. w_l. Intuitively, $G_{i,j,l}$ can be viewed as the similarity between the i-th training example and the j-th validation example according to the l-th layer of the base network. The derivation of Eq. (7) can be found in the supplementary materials.

Two observations can be drawn from Eq. (7). First, it explains the slow Meta-Train step in meta-learning, *i.e.* computing the meta gradient involves enumerating all training examples, all validation examples, and all layers. Second, it shows that the meta gradient can be calculated by first computing the gradient $\sum_{i,j} G_{i,j,:}$ within each individual layer and then aggregating the values together. This finding lays a foundation for the proposed layer-wise meta gradient approximation.

4.2. Layer-wise gradient sampler

We propose to approximate the total meta gradient by aggregating meta gradients sampled from a few layers. We learn a gradient sampler to accumulate the meta gradient for each layer, and formulate the gradient sampler, denoted as $\Gamma(\cdot;\eta_l)$, as follow:

$$\begin{aligned} r_l &= \Gamma(\bar{g}_l^{tra};\eta_l) \\ &= \Gamma(\textit{Avg-Pool}(\frac{1}{n}\sum_{i=1}^{n}\mathcal{V}_i^{tra}(\theta)\frac{\partial \mathcal{L}_i^{tra}(w)}{\partial w_l});\eta_l).\end{aligned} \tag{8}$$

The output to the gradient sampler is the discrete activation status $r_l \in \{0,1\}$.

The input of the gradient sampler is the average gradient $\bar{g}_l^{tra}$ obtained from the Virtual-Train backward step. To be more specific, suppose the gradient tensor for the convolutional kernel has the shape $\mathbb{R}^{D_{out}\times D_{in}\times K_1 \times K_2}$ where D_{out} and D_{in} are the output/input dimensions; K_1 and K_2 are the kernel sizes. The *Avg-Pool* operator averages the gradient tensor across all except the first dimensions while leaving the bias term unchanged. Therefore, the dimension for $\bar{g}_l^{tra} \in \mathbb{R}^{1\times D_{out}}$. The *Avg-Pool* performs a similar operation for the fully connected layer by setting $K_1 = K_2 = 1$.

For efficiency, we adopt a lightweight design for the gradient sampler and implement it by two fully-connected (FC) layers: FC_1 and FC_2, where the first layer FC_1 is followed by a PReLU layer and FC_2 by the "Gumbel-softmax" operator [17]. The hidden size of the fully connected layer is fixed to 128 for all experiments.

Applying the gradient sampler to all layers gives:

$$\mathbf{g}' \propto \frac{-\alpha}{nm} \sum_{l=1}^{L} \mathbb{1}_{[r_l=1]} \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{m} G_{i,j,l} \right) \frac{\partial \mathcal{V}_i^{tra}(\theta)}{\partial \theta} \right), \tag{9}$$

where $\mathbb{1}_{[r_l=1]}$ is the indicator function. As shown in Eq. (9), the meta gradient for the l-th layer is accumulated only if the gradient sampler is turned on (*i.e.* $r_l = 1$).

Finally, we replace $\mathbf{g}$ in Eq. (6) with $\mathbf{g}'$ to update the meta-model.

4.3. Training objective for meta-model

The proposed gradient samplers are jointly optimized with the meta-model. In addition to the cross-entropy loss described in $\mathcal{L}^{val}$ in Eq. (4), we incorporate two auxiliary losses to facilitate learning the gradient samplers.

The first loss is designed to prevent the gradient samplers from activating too many layers. We introduce a loss $\mathcal{L}_r$ regularizing the output of the gradient samplers:

$$\mathcal{L}_r = \| \sum_{l=1}^{L} r_l - K \|_2^2, \tag{10}$$

where K is the expected number of layers to be activated.

Moreover, we add another loss (denoted as $\mathcal{L}_g$) to facilitate learning the meta-model:

$$\mathcal{L}_g = \| \bar{g}_L^{tra} - \bar{g}_L^{val} \|_2^2, \tag{11}$$

where $\bar{g}_L^{tra}$ is the average gradient from training loss discussed in Eq. (8). Likewise $\bar{g}_L^{val}$ is the average gradient from the validation loss, *i.e.* $\bar{g}_L^{val} = \textit{Avg-Pool}(\frac{1}{m}\sum_{j=1}^{m} \frac{\partial \mathcal{L}_j^{val}(\hat{w})}{\partial \hat{w}_L})$. This loss term $\mathcal{L}_g$ captures the prior knowledge that the distance between validation and training gradient should be close. Notice that we only compute the gradients at the last layer L for efficiency.

Finally, the total loss to update the meta-model:

$$\mathcal{L}^{val} = \mathcal{L}_c + \lambda_1 \mathcal{L}_r + \lambda_2 \mathcal{L}_g, \tag{12}$$

where $\mathcal{L}_c$ is the standard cross-entropy loss in Eq. (4). λ_1 and λ_2 are hyperparameters. We will examine the effectiveness of these loss terms in the ablation study.

5. Experiments

We conduct extensive experiments on the noisy labeled data to verify the efficiency and effectiveness of our method for learning robust DNN models. Specifically, we show our method improves the efficiency and generalization performance of the meta-learning methods in Section 5.1. Section 5.2 presents ablation studies to verify our design choices. Section 5.3 compares with the state-of-the-art results on synthetic and realistic noisy labels. In addition, we also experiment on the long-tailed recognition task in Section 5.4. The implementation details and more experimental results are presented in the supplementary materials.

Method	Time (ms)	CIFAR-10			CIFAR-100		
		20%	40%	60%	20%	40%	60%
MW-Net [43]	933	91.9	89.6	84.5	73.1	68.1	61.7
+FaMUS	284(3.3x)	92.9	90.5	85.8	73.6	69.4	62.9
L2R [40]	839	90.5	86.9	82.2	69.3	62.8	50.8
+FaMUS	244(3.4x)	91.3	87.6	82.8	70.7	65.5	51.6

Table 1: Comparison with MW-Net and L2R on CIFAR-10 and CIFAR-100. Percentage numbers represent the noise rate. "Time (ms)" denotes the average running time per training iteration on a single NVIDIA V100 GPU.

Method	Time (ms)	10%	20%	30%	40%
MLC [51]	265	85.23	84.28	82.10	79.89
+FaMUS	84(3.1x)	87.28	85.00	82.65	80.41

Table 2: Comparison with MLC on CIFAR-10 with four different noise rates: $\{10\%, 20\%, 30\%, 40\%\}$. "Time (ms)" denotes the average running time per training iteration on a single NVIDIA V100 GPU.

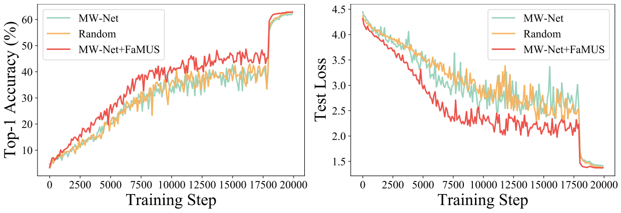
5.1. Comparison with meta-learning methods

This subsection shows our method improves the efficiency and generalization performance of three meta-learning methods: L2R [40], MW-Net [43], and MLC [51].

Setups. We apply our method to three meta-learning methods using their official code and train them under the same settings as reported in their papers [40, 43, 51]. This includes using the same clean validation set to learn the meta-model. The experiments are conducted on the standard CIFAR [22] benchmarks. Following [51], we use the *symmetric* label noise in which a percentage of true labels are randomly replaced with all possible labels, and report the best peak accuracy which is the maximum accuracy on the clean test set during training.

Implementation details. The proposed gradient samplers are jointly optimized with the meta-model by SGD with a momentum of 0.9. The learning rate is fixed as 0.1 throughout the training. λ_1 and λ_2 are both set to 0.1. K is set to 4.

Table 1 and Table 2 show the results on the CIFAR datasets, where "Time" column lists the average running time (in millisecond) per training iteration on a single NVIDIA V100 GPU. It shows that our method accelerates the training time of the three meta-learning methods [40, 43, 51] by at least 3 times. More importantly, our



(a) Top-1 Acc vs. Training Step (b) Test loss vs. Training Step

Figure 3: Test curves under CIFAR-100 with 60% noise.

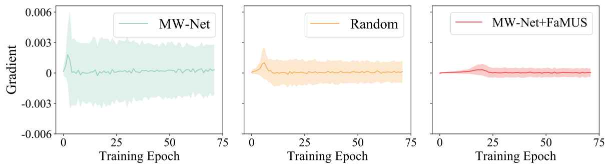


Figure 4: Variance of the meta gradient produced by different methods during the training. All models are trained on the CIFAR-100 dataset with 60% noise.

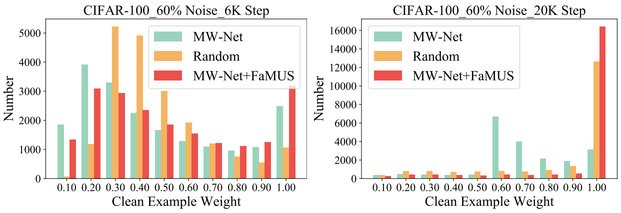


Figure 5: Weight distribution over the clean examples in the *6K* (left) and *20K* (right) training step. All models are trained on the CIFAR-100 dataset with 60% noise.

method improves their generalization performance across all noise rates. To understand the training dynamics, we compare three methods: the best baseline MW-Net [43], the MW-Net with our method, and the Random MW-Net in which each layer is randomly sampled to compute the meta gradient. Figure 3 shows the training curves on the CIFAR-100 dataset with 60% noise. We observe that our method (MW-Net + FaMUS) has the lowest test loss and the highest accuracy throughout the training.

$\mathcal{L}_c$	$\mathcal{L}_r$	$\mathcal{L}_g$	Time (ms)	CIFAR-10	CIFAR-100
✓			933	84.5	61.7
✓		✓	446	85.2	62.2
✓	✓		275	85.3	62.0
✓	✓	✓	284	85.8	62.9

Table 3: Accuracy vs. Training Time on CIFAR-10 and CIFAR-100 with 60% noise. "Time (ms)" denotes the average running time per training iteration on a single NVIDIA V100 GPU.

Model		Time (ms)	ACC
MW-Net [43]		933	61.7
Pre-specified Block	$b = 4$	718	60.8
	$b = 8$	569	61.9
	$b = 12$	326	61.4
Random Layers	$s = 4$	764	61.2
	$s = 8$	826	61.6
	$s = 16$	899	62.1
FaMUS ($K = 4$)		**284**	**62.9**

Table 4: Comparison of sampling strategies on CIFAR-100 with 60% noise. $b \in [1, 12]$ is the index of the residual block. s is the number of randomly selected layers. Our method samples from about 4 layers by setting $K = 4$ in Eq. (10). "Time (ms)" denotes the average running time per training iteration on a single NVIDIA V100 GPU.

We hypothesize that this benefit is related to the low variance in the meta gradient learned by our method. For example, Figure 4 visualizes the variance of the meta gradient during training and shows our method yields a low-variance approximation of the meta gradient. These results suggest that FaMUS can learn to select a small number of most informative layers to compute the meta gradient, which hence reduces the noisy learning signals in the corrupted training data. This observation agrees with the finding in [35, 34] that reduction in gradient variance results in faster and more stable optimization. To substantiate this hypothesis, we examine the meta-models by plotting their weight distribution on the clean examples in Figure 5. We find that in both the early (*6K* step) and late training stages (*20K* step), the meta-models learned by our method tend to assign larger weights to more clean examples. The results in Table 1, Table 2, and Figure 3 demonstrate that our method improves the efficiency and generalization performance of the meta-learning methods. More results can be found in the Appendix C.

5.2. Ablation study

We conduct the ablation studies using the MW-Net method with the WideResNet-28-10 backbone model [59].

Loss function. Table 3 analyzes the impact of the auxiliary loss components in Eq. (12) on the CIFAR datasets with 60% noise. "$\mathcal{L}_c$" denotes the loss of the base meta-learning model (MW-Net). We find that adding "$\mathcal{L}_r$" significantly reduces the training time since it limits the number of layers to be activated. When incorporating both "$\mathcal{L}_r$" and "$\mathcal{L}_g$", our method achieves the best result and improves both the efficiency and accuracy of the base MW-Net model.

Sampling strategy. To verify the design of the proposed gradient sampler, we compare with two predefined sampling strategies: *pre-specified block* and *random layers*. In the Pre-specified Block, we select a residual block (indexed by b and $b \in [1, 12]$), which consists of two convolutional layers and two batch normalization layers, to compute the meta gradients. In the Random Layers, we uniformly select s layers to compute the meta gradients.

Table 4 shows the comparison on the CIFAR-100 dataset with 60% noise rate. For the Pre-specified Block, we find that computing the meta gradient from the top residual

Method	CIFAR-10		CIFAR-100	
	40%	60%	40%	60%
Co-teaching [9]	74.81	73.06	46.20	35.67
L2R [40]	86.92	82.24	62.81	50.81
MW-Net [43]	89.60	84.49	68.11	61.71
MentorNet† [20]	91.20	74.20	66.80	58.80
Mixup† [61]	91.50	86.80	66.80	58.80
M-correction [1]	92.80	90.30	70.10	59.50
MentorMix [18]	94.20	91.30	71.30	64.60
DivideMix [24]	94.90	94.30	75.20	72.00
Ours	**95.37** ±0.15	**94.97** ±0.11	**75.91** ±0.19	**73.58** ±0.28

Table 5: Comparison with the state-of-the-art on CIFAR-10 and CIFAR-100 with 40% and 60% noise rates. † denotes the results are reported by [18].

block ($b = 12$) is the most efficient way, which is about 2x faster than using the bottom block ($b = 4$). As for the Random Layers, the accuracy is improved as the number of layers s increases, while the running time shows a different trend. Our method outperforms all the compared methods both in efficiency and accuracy, suggesting the necessity of the proposed gradient sampler.

5.3. Comparison to state-of-the-art

This subsection compares our method with the state-of-the-art robust learning methods in overcoming both *synthetic* and *realistic noisy labels*.

Datasets. For the realistic noisy labels, we employ three datasets: (mini) WebVision 1.0 [26], Clothing1M [54], and *Controlled Noisy Web Labels* (CNWL) [18]. **WebVision** contains 2.4 million images with noisy labels categorized into the same 1,000 classes as in the ImageNet ILSVRC12. Following the previous works [20, 4], we use the first 50 classes of the Google image subset as the training data. **Clothing1M** has 1 million noisy labeled clothing images crawled from online shopping websites. **CNWL** is a recent benchmark of controlled label noise from the web. Uniquely, it allows for comparing methods on various rates of realistic label noises. We use the Red Mini-ImageNet set [49] that consists of 50K images from 100 classes for training and 5K images for testing.

Implementation details. To fairly compare with the the-state-of-art, we use a subset of pseudo labeled training data as the meta-learning validation set. Inspired by [24], we employ the Gaussian Mixture Model (GMM) to divide the training data into a pseudo-clean and a pseudo-noisy label set. By doing so, no extra clean labels nor data are used to train the meta-model. We find using the pseudo validation set notably improves the performance because the pseudo validation set is much larger than the clean validation set used in the meta-learning method [43]. More discussions can be found in the Appendix E.

Method	WebVision		ILSVRC12	
	top1	top5	top1	top5
F-correction [37]	61.12	82.68	57.39	82.36
Decoupling [33]	62.54	84.74	58.26	82.26
D2L [31]	62.68	84.00	57.80	81.36
MentorNet [20]	63.00	81.40	57.80	79.92
Co-teaching [9]	63.58	85.20	61.48	84.70
Iterative-CV [4]	65.24	85.34	61.60	84.98
MW-Net [43]	74.52	88.89	72.60	88.80
MentorMix [18]	76.00	90.20	72.90	91.10
DivideMix [24]	77.32	91.64	75.20	90.84
Ours	**79.40**	**92.80**	**77.00**	**92.76**

Table 6: Comparison with the state-of-the-art on (mini) WebVision dataset. Numbers denote top-1 (top-5) accuracy on the validation set of WebVision and ImageNet ILSVRC12.

Method	20%	40%	60%	80%	Mean
Cross-entropy	47.36	42.70	37.30	29.76	39.28
Mixup [61]	49.10	46.40	40.58	33.58	42.41
DivideMix [24]	50.96	46.72	43.14	34.50	43.83
MentorMix [18]	51.02	47.14	43.80	33.46	43.85
Ours	**51.42**	**48.06**	**45.10**	**35.50**	**45.02**

Table 7: Results on Controlled Noisy Web Labels [18].

For experiments on CIFAR-10, CIFAR-100, and CNWL, we employ the PreAct ResNet-18 [14] as the base DNN. For experiments on Clothing1M and WebVision, we use ResNet-50 [13] and Inception-ResNet V2 [46], respectively.

Baselines. We briefly introduce the baselines: (1) **Co-teaching** [9], **Decoupling** [33], and **JoCoR** [52] train two networks to improve each other. (2) **F-correction** [37] estimates the noise transition matrix to correct the loss function. (3) **D2L** [31] learns to monitor the dimensionality of subspaces and adapts the loss functions accordingly. (4) **Iterative-CV** [4] iteratively increases the number of the selected samples to train the networks. (5) **MentorNet** [20] is an example-weighting method based on curriculum learning. **MentorMix** [18] further combines the MentorNet with the Mixup [61]. (6) **DivideMix** [24] addresses the corrupted labels in a semi-supervised learning fashion. (7) **M-correction** [1] estimates the probability of a sample being mislabelled and then corrects the loss accordingly.

5.3.1 Results on synthetic noisy labels

Table 5 shows the results on the CIFAR-10 and CIFAR-100 datasets with symmetric label noises. For the compared methods, we directly cite the reported numbers in their papers except for MW-Net [43] and L2R [40] where we report the reproduced results. For our method, we report the average and standard deviation of over three training trials using different random seeds. The gains over baseline methods are statistically significant at the p-value level of 0.05, according to the one-tailed t-test. These results illustrate the

Method	Long-Tailed CIFAR-10				Long-Tailed CIFAR-100			
	100	50	20	10	100	50	20	10
CE loss	70.36	74.81	82.23	86.39	38.32	43.85	51.14	55.71
Focal Loss† [28]	70.38	76.71	82.76	86.66	38.41	44.32	51.95	55.78
CB Focal† [6]	74.57	79.27	84.36	87.49	39.60	45.32	52.59	57.99
LDAM-DRW [2]	77.03	-	-	88.16	44.70	-	-	59.59
BBN [63]	79.82	82.18	-	88.32	42.56	47.02	-	59.12
L2R† [40] with CE loss	74.16	78.93	82.12	85.19	40.23	44.44	51.64	53.73
MW-Net [43] with CE loss	75.21	80.06	84.94	87.84	42.09	46.74	54.37	58.46
[16] with CE loss	76.41	80.51	86.46	88.85	43.35	48.53	55.62	59.58
[16] with LDAM	80.00	82.34	84.37	87.40	44.08	49.16	52.38	58.00
MW-Net with CE loss + FaMUS	79.30	83.15	**87.15**	**89.39**	45.60	49.56	**56.22**	**60.42**
MW-Net with LDAM loss + FaMUS	**80.96**	**83.32**	86.24	87.90	**46.03**	**49.93**	55.95	59.03

Table 8: Top-1 test accuracy of ResNet-32 on the long-tailed CIFAR-10 and CIFAR-100 with four imbalanced factors $\{100, 50, 20, 10\}$. Methods in the bottom block use extra clean data. The best performance is in **bold** and the second best is underscored. † denotes the results are reported by [2].

effectiveness of our method on the synthetic noisy labels.

5.3.2 Results on realistic noisy labels

Table 6 shows the results on the WebVision dataset. As shown, our method consistently outperforms the baselines, achieving the best accuracy on the validation sets of WebVision and ImageNet. In particular, our method performs favorably against very recent methods such as MentorMix [18] and DivideMix [24] in the top-1 accuracy.

Table 7 shows the results on the CNWL dataset. We implement several strong baselines using their official codes released on the CIFAR-100 dataset, *e.g.*, MentorMix [18]. Note that in order to use their implementation, 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 [18]'s results on 32x32 images. More details are discussed in the Appendix E. Table 7 shows that our method outperforms all baseline methods on the realistic web noisy labels. The result is notable because 1) it verifies our method on the challenging CNWL dataset; 2) it demonstrates our consistent improvement across all noise rates as a useful and robust feature since the underlying noise rate is often unknown in practice.

We also apply our method on the Clothing1M dataset, and achieve 74.4% in top-1 accuracy without using extra clean data, which is comparable to recently published methods. The results on the above three datasets demonstrate that our method trained with a noisy validation set is effective for addressing the realistic noisy labels.

5.4. Long-tailed recognition task

In addition to the noisy training label problem, we also evaluate our method on the long-tailed recognition task.

Datasets and implementation details. Four imbalanced factors $\{100, 50, 20, 10\}$ are applied on the long-tailed CIFAR-10 and CIFAR-100 [6]. The number of training samples for each class is randomly removed by $n_i \mu^i$, where i indicates the class index, n_i is the original number of the training samples for the i-th class, and $\mu \in (0, 1)$. The imbalanced factor is the ratio between the largest and the smallest class. Following [43, 16], we do not change the test set and select ten training images per class as the clean validation set. Our method is implemented on the MW-Net model [43] with the ResNet-32 backbone [13].

From Table 8, we find our method consistently outperforms previous meta-learning based methods [40, 43, 16]. Moreover, our method accelerates the training of the meta-learning model MW-Net by 2.9 times. It is noteworthy that even compared to the very recent approaches (*e.g.*, improved L2R [16]), our method still obtains a reasonable performance gain, which illustrates the effectiveness of our method on the long-tailed recognition task.

6. Conclusion

In this paper, we discuss a novel Faster Meta Update Strategy (FaMUS) to efficiently approximate the meta gradients by a layer-wise meta gradient sampling fashion. We empirically show that our method yields not only an accurate but also a low-variance approximation of the meta gradient. The experimental results demonstrate that FaMUS is able to reduce two-thirds of the training time of the meta-learning methods, while achieving a better generalization performance. Our method yields the state-of-the-art performance to address the noisy label problem, and obtains competitive performance on the long-tailed recognition task.

We find meta-model training is considerably influenced by the quantity and quality of the pseudo-clean label set. Future research in this area may include improving the robustness on limited validation data or low-quality pseudo validation data, in addition to further closing the gap in training time.

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. 2, 7

[2] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019. 2, 3, 8

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 2

[4] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *ICML*, 2019. 7

[5] Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. Advaug: Robust adversarial augmentation for neural machine translation. In *ACL*, 2020. 2

[6] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019. 2, 3, 8

[7] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer, 2003. 2, 3

[8] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017. 2

[9] 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 *NeurIPS*, 2018. 1, 2, 7

[10] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*. Springer, 2005. 2

[11] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009. 1

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 7, 8

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 7

[15] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*, 2018. 2

[16] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *CVPR*, 2020. 2, 3, 8

[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. 2017. 5

[18] Lu Jiang, Di Huang, Mason Liu, and Weilong Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *ICML*, 2020. 2, 7, 8

[19] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. Self-paced curriculum learning. In *AAAI*, 2015. 3

[20] 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 *ICML*, 2018. 1, 2, 7

[21] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587, 2017. 2, 3

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1, 5

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[24] Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *ICLR*, 2020. 1, 2, 7, 8

[25] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *CVPR*, 2019. 2

[26] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017. 1, 2, 7

[27] Junwei Liang, Lu Jiang, and Alexander Hauptmann. Simaug: Learning robust representations from simulation for trajectory prediction. In *ECCV*, 2020. 2

[28] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1, 8

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1

[30] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019. 2, 3

[31] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah M Erfani, Shu-Tao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *ICML*, 2018. 7

[32] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 2, 3

[33] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *NeurIPS*, 2017. 2, 7

[34] Andrew Miller, Nick Foti, Alexander D'Amour, and Ryan P Adams. Reducing reparameterization gradient variance. In *NeurIPS*, 2017. 2, 6

[35] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015. 2, 6

[36] Curtis G Northcutt, Lu Jiang, and Isaac L Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 2021. 2

[37] 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. 2, 7

[38] Geoff Pleiss, Tianyi Zhang, Ethan R Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *arXiv preprint arXiv:2001.10528*, 2020. 2

[39] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1

[40] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018. 1, 2, 3, 4, 5, 7, 8

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 1

[42] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 2, 3

[43] 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, 5, 6, 7, 8

[44] Jun Shu, Qian Zhao, Zengben Xu, and Deyu Meng. Meta transition adaptation for robust deep learning with noisy labels. *arXiv preprint arXiv:2006.05697*, 2020. 1

[45] Chi Su, Jianing Li, Shiliang Zhang, Junliang Xing, Wen Gao, and Qi Tian. Pose-driven deep convolutional model for person re-identification. In *ICCV*, pages 3960–3969, 2017. 2

[46] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2016. 7

[47] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *CVPR*, 2018. 1, 2

[48] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *NeurIPS*, 2017. 2

[49] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, 2016. 7

[50] Nidhi Vyas, Shreyas Saxena, and Thomas Voice. Learning soft labels via meta learning. *arXiv preprint arXiv:2009.09496*, 2020. 1, 3

[51] Zhen Wang, Guosheng Hu, and Qinghua Hu. Training noise-robust deep neural networks via meta-learning. In *CVPR*, 2020. 1, 2, 3, 4, 5

[52] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. In *CVPR*, 2020. 2, 3, 7

[53] Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? In *NeurIPS*, 2019. 2

[54] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015. 1, 7

[55] Chenglin Yang, Lingxi Xie, Chi Su, and Alan L Yuille. Snapshot distillation: Teacher-student optimization in one generation. In *CVPR*, 2019. 2

[56] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *CVPR*, 2019. 2

[57] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for face recognition with under-represented data. In *CVPR*, 2019. 2, 3

[58] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *ICML*, 2019. 2

[59] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMCV*, 2016. 6

[60] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. 1

[61] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 2, 7

[62] Zizhao Zhang, Han Zhang, Sercan O Arik, Honglak Lee, and Tomas Pfister. Distilling effective supervision from severe label noise. In *CVPR*, 2020. 2

[63] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, 2020. 2, 3, 8

[64] Linchao Zhu and Yi Yang. Inflated episodic memory with region self-attention for long-tailed visual recognition. In *CVPR*, 2020. 1, 2