Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, Deyu Meng[*]
Xi'an Jiaotong University
dymeng@mail.xjtu.edu.cn

Abstract

Current deep neural networks (DNNs) can easily overfit to biased training data with corrupted labels or class imbalance. Sample re-weighting strategy is commonly used to alleviate this issue by designing a weighting function mapping from training loss to sample weight, and then iterating between weight recalculating and classifier updating. Current approaches, however, need manually pre-specify the weighting function as well as its additional hyper-parameters. It makes them fairly hard to be generally applied in practice due to the significant variation of proper weighting schemes relying on the investigated problem and training data. To address this issue, we propose a method capable of adaptively learning an explicit weighting function directly from data. The weighting function is an MLP with one hidden layer, constituting a universal approximator to almost any continuous functions, making the method able to fit a wide range of weighting functions including those assumed in conventional research. Guided by a small amount of unbiased meta-data, the parameters of the weighting function can be finely updated simultaneously with the learning process of the classifiers. Synthetic and real experiments substantiate the capability of our method for achieving proper weighting functions in class imbalance and noisy label cases, fully complying with the common settings in traditional methods, and more complicated scenarios beyond conventional cases. This naturally leads to its better accuracy than other state-of-the-art methods.

1 Introduction

DNNs have recently obtained impressive good performance on various applications due to their powerful capacity for modeling complex input patterns. However, DNNs can easily overfit to biased training data[1], like those containing corrupted labels [2] or with class imbalance[3], leading to their poor performance in generalization in such cases. This robust deep learning issue has been theoretically illustrated in multiple literatures [4, 5, 6, 7, 8, 9].

In practice, however, such biased training data are commonly encountered. For instance, training samples collected from crowdsourcing systems [10] or search engines [11, 12] always contain corrupted labels [13, 14, 15, 16, 17, 18, 19, 20]. Another popular type of training set bias is class imbalance. In real datasets, a few classes always account for most of the data, while most classes are under-represented. Effective learning with these biased training data, which is regarded to be deviated from evaluation/test ones, is thus an important while challenging issue in machine learning [1, 21].

Sample reweighting approach is a commonly used strategy against this robust learning issue. The main methodology is to design a weighting function mapping from training loss to sample weight (with hyper-parameters), and then iterates between calculating weights from current training loss values

---

[1]We call the training data biased when they are generated from a joint sample-label distribution deviating from the distribution of evaluation/test set[1].
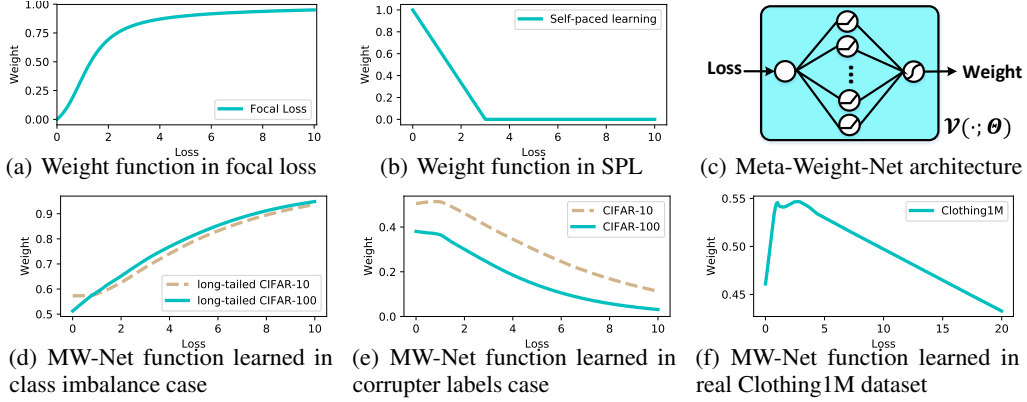
Figure 1: (a)-(b) weight functions set in focal loss and self-paced learning (SPL). (c) Meta-Weighting-Net architecture. (d)-(f) Meta-Weighting-Net functions learned in class imbalance (imbalanced factor 100), noisy label (40% uniform noise), and real dataset, respectively, by our method.

and minimizing weighted training loss for classifier updating. There exist two entirely contradictive ideas for constructing such a loss-weight mapping. One makes the function monotonically increasing as depicted in Fig. 1(a), i.e., enforce the learning to more emphasize samples with larger loss values since they are more like to be uncertain hard samples located on the classification boundary. Typical methods of this category include AdaBoost [22, 23], hard negative mining [24] and focal loss [25]. This sample weighting manner is known to be necessary for class imbalance problems, since it can prioritize the minority class with relatively higher training losses.

On the contrary, the other methodology sets the weighting function as monotonically decreasing, as shown in Fig. 1(b), to take samples with smaller loss values as more important ones. The rationality lies on that these samples are more likely to be high-confident ones with clean labels. Typical methods include self-paced learning(SPL) [26], iterative reweighting [27, 20] and multiple variants [28, 29, 30]. This weighting strategy has been especially used in noisy label cases, since it inclines to suppress the effects of samples with extremely large loss values, possibly with corrupted incorrect labels.

Although these sample reweighting methods help improve the robustness of a learning algorithm on biased training samples, they still have evident deficiencies in practice. On the one hand, current methods need to manually set a specific form of weighting function based on certain assumptions on training data. This, however, tends to be infeasible when we know little knowledge underlying data or the label conditions are too complicated, like the case that the training set is both imbalanced and noisy. On the other hand, even when we specify certain weighting schemes, like focal loss [25] or SPL [26], they inevitably involve hyper-parameters, like focusing parameter in the former and age parameter in the latter, to be manually preset or tuned by cross-validation. This tends to further raise their application difficulty and reduce their performance stability in real problems.

To alleviate the aforementioned issue, this paper presents an adaptive sample weighting strategy to automatically learn an explicit weighting function from data. The main idea is to parameterize the weighting function as an MLP (multilayer perceptron) network with only one hidden layer (as shown in Fig. 1(c)), called *Meta-Weight-Net*, which is theoretically a universal approximator for almost any continuous function [31], and then use a small unbiased validation set (meta-data) to guide the training of all its parameters. The explicit form of the weighting function can be finally attained specifically suitable to the learning task.

In summary, this paper makes the following three-fold contributions:

1) We propose to automatically learn an explicit loss-weight function, parameterized by an MLP from data in a meta-learning manner. Due to the universal approximation capability of this weight net, it can finely fit a wide range of weighting functions including those used in conventional research.

2) Experiments verify that the weighting functions learned by our method highly comply with manually preset weighting manners used in tradition in different training data biases, like class imbalance and noisy label cases as shown in Fig. 1(d) and 1(e)), respectively. This shows that the weighting scheme learned by the proposed method inclines to help reveal deeper understanding for

data bias insights, especially in complicated bias cases where the extracted weighting function is with complex tendencies (as shown in Fig. 1(f)).

3) The insights of why the proposed method works can be well interpreted. Particularly, the updating equation for Meta-Weight-Net parameters can be explained by that ==the sample weights of those samples better complying with the meta-data knowledge will be improved, while those violating such meta-knowledge will be suppressed.== This tallies with our common sense on the problem: we should reduce the influence of those highly biased ones, while emphasize those unbiased ones.

The paper is organized as follows. Section 2 presents the proposed meta-learning method as well as the detailed algorithm and analysis of its convergence property. Section 3 discusses related work. Section 4 demonstrates experimental results and the conclusion is finally made.

2   The Proposed Meta-Weight-Net Learning Method

2.1   The Meta-learning Objective

Consider a classification problem with the training set $\{x_i, y_i\}_{i=1}^N$, where x_i denotes the i-th sample, $y_i \in \{0, 1\}^c$ is the label vector over c classes, and N is the number of the entire training data. $f(x, \mathbf{w})$ denotes the classifier, and $\mathbf{w}$ denotes its parameters. In current applications, $f(x, \mathbf{w})$ is always set as a DNN. We thus also adopt DNN, and call it the classifier network for convenience in the following.

Generally, the optimal classifier parameter $\mathbf{w}^*$ can be extracted by minimizing the loss $\frac{1}{N} \sum_{i=1}^N \ell(y_i, f(x_i, \mathbf{w}))$ calculated on the training set. For notation convenience, we denote that $L_i^{train}(\mathbf{w}) = \ell(y_i, f(x_i, \mathbf{w}))$. In the presence of biased training data, sample re-weighting methods enhance the robustness of training by imposing weight $\mathcal{V}(L_i^{train}(\mathbf{w}); \Theta)$ on the i-th sample loss, where $\mathcal{V}(\ell; \Theta)$ denotes the weight net, and Θ represents the parameters contained in it. The optimal parameter $\mathbf{w}$ is calculated by minimizing the following weighted loss:

$$\mathbf{w}^*(\Theta) = \arg\min_{\mathbf{w}} \ \mathcal{L}^{train}(\mathbf{w}; \Theta) \triangleq \frac{1}{N} \sum_{i=1}^N \mathcal{V}(L_i^{train}(\mathbf{w}); \Theta) L_i^{train}(\mathbf{w}). \tag{1}$$

Meta-Weight-Net: Our method aims to automatically learn the hyper-parameters Θ in a meta-learning manner. To this aim, we formulate $\mathcal{V}(L_i(\mathbf{w}); \Theta)$ as a MLP network with only one hidden layer containing ==100 nodes==, as shown in Fig. 1(c). We call this weight net as ***Meta-Weight-Net*** or ***MW-Net*** for easy reference. Each hidden node is with ReLU activation function, and the output is with the Sigmoid activation function, to guarantee the output located in the interval of $[0, 1]$. Albeit simple, this net is known as a universal approximator for almost any continuous function [31], and thus can fit a wide range of weighting functions including those used in conventional research.

Meta learning process. The parameters contained in MW-Net can be optimized by using the meta learning idea [32, 33, 34, 35]. Specifically, assume that we have a small amount unbiased meta-data set (i.e., with clean labels and balanced data distribution) $\{x_i^{(meta)}, y_i^{(meta)}\}_{i=1}^M$, representing the meta-knowledge of ground-truth sample-label distribution, where M is the number of meta-samples and $M \ll N$. The optimal parameter Θ^* can be obtained by minimizing the following meta-loss:

$$\Theta^* = \arg\min_{\Theta} \ \mathcal{L}^{meta}(\mathbf{w}^*(\Theta)) \triangleq \frac{1}{M} \sum_{i=1}^M L_i^{meta}(\mathbf{w}^*(\Theta)), \tag{2}$$

where $L_i^{meta}(\mathbf{w}) = \ell\left(y_i^{(meta)}, f(x_i^{(meta)}, \mathbf{w})\right)$ is calculated on meta-data.

2.2   The Meta-Weight-Net Learning Method

Calculating the optimal Θ^* and $\mathbf{w}^*$ require two nested loops of optimization. Here we adopt an online strategy to update Θ and $\mathbf{w}$ through a single optimization loop, respectively, to guarantee the efficiency of the algorithm.

Formulating learning manner of classifier network. As general network training tricks, we employ SGD to optimize the training loss (1). Specifically, in each iteration of training, a mini-batch of training samples $\{(x_i, y_i), 1 \le i \le n\}$ is sampled, where n is the mini-batch size. Then the updating equation of the classifier network parameter can be formulated by moving the current $\mathbf{w}^{(t)}$ along the descent direction of the objective loss in Eq. (1) on a mini-batch training data:
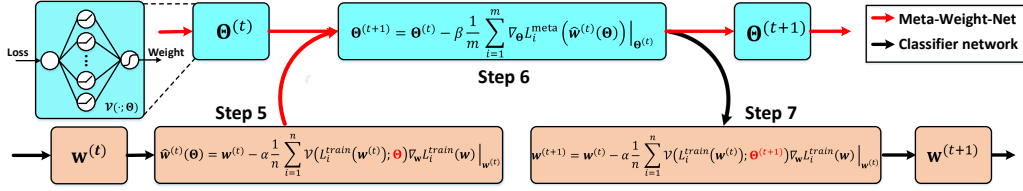
Figure 2: Main flowchart of the proposed MW-Net Learning algorithm (steps 5-7 in Algorithm 1).

$$\hat{\mathbf{w}}^{(t)}(\Theta) = \mathbf{w}^{(t)} - \alpha \frac{1}{n} \times \sum_{i=1}^{n} \mathcal{V}(L_i^{train}(\mathbf{w}^{(t)}); \Theta) \nabla_{\mathbf{w}} L_i^{train}(\mathbf{w}) \Big|_{\mathbf{w}^{(t)}}, \qquad (3)$$

where α is the step size.

Algorithm 1 The MW-Net Learning Algorithm

Input: Training data $\mathcal{D}$, meta-data set $\widehat{\mathcal{D}}$, batch size n, m, max iterations T.
Output: Classifier network parameter $\mathbf{w}^{(T)}$
1: Initialize classifier network parameter $\mathbf{w}^{(0)}$ and Meta-Weight-Net parameter $\Theta^{(0)}$.
2: **for** $t = 0$ **to** $T - 1$ **do**
3:     $\{x, y\} \leftarrow$ SampleMiniBatch$(\mathcal{D}, n)$.
4:     $\{x^{(meta)}, y^{(meta)}\} \leftarrow$ SampleMiniBatch$(\widehat{\mathcal{D}}, m)$.
5:     Formulate the classifier learning function $\hat{\mathbf{w}}^{(t)}(\Theta)$ by Eq. (3).
6:     Update $\Theta^{(t+1)}$ by Eq. (9).
7:     Update $\mathbf{w}^{(t+1)}$ by Eq. (5).
8: **end for**

Updating parameters of Meta-Weight-Net: After receiving the feedback of the classifier network parameter updating formulation $\hat{\mathbf{w}}^{(t)}(\Theta)$ [2] from the Eq .(3), the parameter Θ of the Meta-Weight-Net can then be readily updated guided by Eq. (2), i.e., moving the current parameter $\Theta^{(t)}$ along the objective gradient of Eq. (2) calculated on the meta-data:

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{m} \sum_{i=1}^{m} \nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{\Theta^{(t)}}, \qquad (4)$$

where β is the step size.

Updating parameters of classifier network: Then, the updated $\Theta^{(t+1)}$ is employed to ameliorate the parameter $\mathbf{w}$ of the classifier network, i.e.,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \frac{1}{n} \times \sum_{i=1}^{n} \mathcal{V}(L_i^{train}(\mathbf{w}^{(t)}); \Theta^{(t+1)}) \nabla_{\mathbf{w}} L_i^{train}(\mathbf{w}) \Big|_{\mathbf{w}^{(t)}}. \qquad (5)$$

The MW-Net Learning algorithm can then be summarized in Algorithm 1, and Fig. 2 illustrates its main implementation process (steps 5-7). All computations of gradients can be efficiently implemented by automatic differentiation techniques and generalized to any deep learning architectures of classifier network. The algorithm can be easily implemented using popular deep learning frameworks like PyTorch [36]. It is easy to see that both the classifier network and the MW-Net gradually ameliorate their parameters during the learning process based on their values calculated in the last step, and the weights can thus be updated in a stable manner, as clearly shown in Fig. 6.

2.3 Analysis on the Weighting Scheme of Meta-Weight-Net

The computation of Eq. (9) by backpropagation can be rewritten as[3]:

$$\Theta^{(t+1)} = \Theta^{(t)} + \frac{\alpha\beta}{n} \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} G_{ij} \right) \frac{\partial \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}}, \qquad (6)$$

---

[2]Notice that Θ here is a variable instead of a quantity, which makes $\hat{\mathbf{w}}^t(\Theta)$ a function of Θ and the gradient in Eq. (9) be able to be computed.

[3]Derivation can be found in supplementary materials.

where $G_{ij} = \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^{T} \frac{\partial L_j^{train}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}}$. Neglecting the coefficient $\frac{1}{m}\sum_{i=1}^{m} G_{ij}$, it is easy to see that each term in the sum orients to the ascend gradient of the weight function $\mathcal{V}(L_j^{train}(\mathbf{w}^{(t)});\Theta)$. $\frac{1}{m}\sum_{i=1}^{m} G_{ij}$, the coefficient imposed on the j-th gradient term, represents the similarity between the gradient of the j-th training sample computed on training loss and the average gradient of the mini-batch meta data calculated on meta loss. That means if the learning gradient of a training sample is similar to that of the meta samples, then it will be considered as beneficial for getting right results and its weight tends to be more possibly increased. Conversely, the weight of the sample inclines to be suppressed. This understanding is consistent with why well-known MAML works [37, 38, 39].

2.4 Convergence of the MW-Net Learning algorithm

Our algorithm involves optimization of two-level objectives, and therefore we show theoretically that our method converges to the critical points of both the meta and training loss function under some mild conditions in Theorem 3 and 4, respectively. The proof is listed in the supplementary material.

Theorem 1. *Suppose the meta loss function is Lipschitz smooth with constant L, and $\mathcal{V}(\cdot)$ is differential with a δ-bounded gradient and twice differential with its Hessian bounded by $\mathcal{B}$, and the loss function ℓ have ρ-bounded gradients with respect to training/meta data. Let the learning rate α_t satisfies $\alpha_t = \min\{1, \frac{k}{T}\}$, for some $k > 0$, such that $\frac{k}{T} < 1$, and $\beta_t, 1 \le t \le N$ is a monotone descent sequence, $\beta_1 = \min\{\frac{1}{L}, \frac{c}{\sigma\sqrt{T}}\}$ for some $c > 0$, such that $\frac{\sigma\sqrt{T}}{c} \ge L$. Then the proposed algorithm can achieve $\mathbb{E}[\|\nabla G(\Theta^{(t)})\|_2^2] \le \epsilon$ in $\mathcal{O}(1/\epsilon^2)$ steps. More specifically,*

$$\min_{0 \le t \le T} \mathbb{E}[\|\nabla \mathcal{L}^{meta}(\Theta^{(t)})\|_2^2] \le \mathcal{O}(\frac{C}{\sqrt{T}}), \tag{7}$$

where C is some constant independent of the convergence process, and σ is the variance of drawing uniformly mini-batch sample at random.

Theorem 2. *Suppose the loss function ℓ is Lipschitz-smooth with constant L, and have ρ-bounded gradients with respect to training/meta data. Let the learning rate α_t satisfies $\alpha_t = \min\{1, \frac{k}{T}\}$, for some $k > 0$, such that $\frac{k}{T} < 1$, and $\beta_t, 1 \le t \le N$ is a monotone descent sequence.*

$$\lim_{t \to \infty} \mathbb{E}[\|\nabla \mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] = 0. \tag{8}$$

3 Related Work

Sample Weighting Methods. The idea of reweighting examples can be dated back to dataset resampling [40, 41] or instance re-weight [42], which pre-evaluates the sample weights as a pre-processing step by using certain prior knowledge on the task or data. To make the sample weights fit data more flexibly, more recent researchers focused on pre-designing a weighting function mapping from training loss to sample weight, and dynamically ameliorate weights during training process [43, 44]. There are mainly two manners to design the weighting function. One is to make it monotonically increasing, specifically effective in class imbalance case. Typical methods include the boosting algorithm (like AdaBoost [22]) and multiple of its variations [45], hard example mining [24] and focal loss [25], which impose larger weights to ones with larger loss values. On the contrary, another series of methods specify the weighting function as monotonically decreasing, especially used in noisy label cases. For example, SPL [26] and its extensions [28, 29], iterative reweighting [27, 20] and other recent work [46, 30], pay more focus on easy samples with smaller losses. The limitation of these methods are that they all need to manually pre-specify the form of weighting function as well as their hyper-parameters, raising their difficulty to be readily used in real applications.

Meta Learning Methods. Inspired by meta-learning developments [47, 48, 49, 37, 50], recently some methods were proposed to learn an adaptive weighting scheme from data to make the learning more automatic and reliable. Typical methods along this line include FWL [51], learning to teach [52, 32] and MentorNet [21] methods, whose weight functions are designed as a Bayesian function approximator, a DNN with attention mechanism, a bidirectional LSTM network, respectively. Instead of only taking loss values as inputs as classical methods, the weighting functions they used (i.e., the meta-learner), however, are with much more complex forms and required to input complicated

information (like sample features). This makes them not only hard to succeed good properties possessed by traditional methods, but also to be easily reproduced by general users.

A closely related method, called ==L2RW== [1], adopts a similar meta-learning mechanism compared with ours. The major difference is that the weights are implicitly learned there, without an explicit weighting function. This, however, might lead to unstable weighting behavior during training and unavailability for generalization. In contrast, with the explicit yet simple Meta-Weight-Net, our method can learn the weight in a more stable way, as shown in Fig. 6, and can be easily generalized from a certain task to related other ones (see in the supplementary material).

Other Methods for Class Imbalance. Other methods for handling data imbalance include: [53, 54] tries to transfer the knowledge learned from major classes to minor classes. The metric learning based methods have also been developed to effectively exploit the tailed data to improve the generalization ability, e.g., triple-header loss [55] and range loss [56].

Other Methods for Corrupted Labels. For handling noisy label issue, multiple methods have been designed by correcting noisy labels to their true ones via a supplemental clean label inference step [14, 17, 57, 16, 21, 1, 18]. For example, GLC [18] proposed a loss correction approach to mitigate the effects of label noise on DNN classifiers. Other methods along this line include the Reed [58], Co-training [19], D2L [59] and S-Model [15].

4 Experimental Results

To evaluate the capability of the proposed algorithm, we implement experiments on data sets with class imbalance and noisy label issues, and real-world dataset with more complicated data bias.

4.1 Class Imbalance Experiments

We use Long-Tailed CIFAR dataset [60], that reduces the number of training samples per class according to an exponential function $n = n_i \mu^i$, where i is the class index, n_i is the original number of training images and $\mu \in (0, 1)$. The imbalance factor of a dataset is defined as the number of training samples in the largest class divided by the smallest. We trained ResNet-32 [61] with softmax cross-entropy loss by SGD with a momentum 0.9, a weight decay 5×10^{-4}, an initial learning rate 0.1. The learning rate of ResNet-32 is divided by 10 after 80 and 90 epoch (for a total 100 epochs), and the learning rate of WN-Net is fixed as 10^{-3}. We randomly selected 10 images per class in validation set as the meta-data set. The compared methods include: 1) **BaseModel**, which uses a softmax cross-entropy loss to train ResNet-32 on the training set; 2) **Focal loss** [25] and **Class-Balanced** [60] represent the state-of-the-arts of the predefined sample reweighting techniques; 3) **Fine-tuning**, fine-tune the result of BaseModel on the meta-data set; 4) **L2RW** [1], which leverages an additional meta-dataset to adaptively assign weights on training samples.

Table 1 shows the classification accuracy of ResNet-32 on the test set and confusion matrices are displayed in Fig. 3 (more details are listed in the supplementary material). It can be observed that: 1) Our algorithm evidently outperforms other competing methods on datasets with class imbalance, showing its robustness in such data bias case; 2) When imbalance factor is 1, i.e., all classes are with same numbers of samples, fine-tuning runs best, and our method still attains a comparable performance; 3) When imbalance factor is 200 on long-tailed CIFAR-100, the smallest class has only two samples. An extra fine-tuning achieves performance gain, while our method still perform well in such extreme data bias.

To understand the weighing scheme of MW-Net, we depict the tendency curve of weight with respect to loss by the learned MW-Net in Fig. 1(d), which complies with the classical optimal weighting manner to such data bias. i.e., larger weights should be imposed on samples with relatively large losses, which are more likely to be minority class sample.

4.2 Corrupted Label Experiment

We study two settings of corrupted labels on the training set: 1) ==Uniform noise.== The label of each sample is independently changed to a random class with probability p following the same setting in [2]. 2) ==Flip noise.== The label of each sample is independently flipped to similar classes with total probability p. In our experiments, we randomly select two classes as similar classes with equal

Table 1: Test accuracy (%) of ResNet-32 on long-tailed CIFAR-10 and CIFAR-100, and the best and the second best results are highlighted in **bold** and ***italic bold***, respectively.

Dataset Name	Long-Tailed CIFAR-10						Long-Tailed CIFAR-100					
Imbalance	200	100	50	20	10	1	200	100	50	20	10	1
BaseModel	65.68	70.36	74.81	82.23	86.39	92.89	34.84	38.32	43.85	51.14	55.71	70.50
Focal Loss	65.29	70.38	76.71	82.76	86.66	***93.03***	35.62	38.41	44.32	51.95	55.78	***70.52***
Class-Balanced	***68.89***	***74.57***	***79.27***	***84.36***	***87.49***	92.89	36.23	39.60	45.32	***52.59***	***57.99***	70.50
Fine-tuning	66.08	71.33	77.42	83.37	86.42	93.23	**38.22**	***41.83***	***46.40***	52.11	57.44	**70.72**
L2RW	66.51	74.16	78.93	82.12	85.19	89.25	33.38	40.23	44.44	51.64	53.73	64.11
Ours	**68.91**	**75.21**	**80.06**	**84.94**	**87.84**	92.66	***37.91***	**42.09**	**46.74**	**54.37**	**58.46**	70.37

Table 2: Test accuracy comparison on CIFAR-10 and CIFAR-100 of WRN-28-10 with varying noise rates under uniform noise. Mean accuracy (±std) over 5 repetitions are reported ('—' means the method fails).

Datasets / Noise Rate		BaseModel	Reed-Hard	S-Model	Self-paced	Focal Loss	Co-teaching	D2L	Fine-tining	MentorNet	L2RW	GLC	Ours
CIFAR-10	0%	95.60±0.22	94.38±0.14	83.79±0.11	90.81±0.34	**95.70±0.15**	88.67±0.25	94.64±0.33	***95.65±0.15***	94.35±0.42	92.38±0.10	94.30±0.19	94.52±0.25
	40%	68.07±1.23	81.26±0.51	79.58±0.33	86.41±0.29	75.96±1.31	74.81±0.34	85.60±0.13	80.47±0.25	87.33±0.22	86.92±0.19	***88.28±0.03***	**89.27±0.28**
	60%	53.12±3.03	73.53±1.54	—	53.10±1.78	51.87±1.19	73.06±0.25	68.02±0.41	78.75±2.40	82.80±1.35	82.24±0.36	***83.49±0.24***	**84.07±0.33**
CIFAR-100	0%	79.95±1.26	64.45±1.02	52.86±0.99	59.79±0.46	**81.04±0.24**	61.80±0.25	66.17±1.42	***80.88±0.21***	73.26±1.23	72.99±0.58	73.75±0.51	78.76±0.24
	40%	51.11±0.42	51.27±1.18	42.12±0.99	46.31±2.45	51.19±0.46	46.20±0.15	52.10±0.97	52.49±0.74	***61.39±3.99***	60.79±0.91	61.31±0.22	**67.73±0.26**
	60%	30.92±0.33	26.95±0.98	—	19.08±0.57	27.70±3.77	35.67±1.25	41.11±0.30	38.16±0.38	36.87±1.47	48.15±0.34	***50.81±1.00***	**58.75±0.11**

probability. Two benchmark datasets are employed: CIFAR-10 and CIFAR-100 [62]. Both are popularly used for evaluation of noisy labels [59, 19].1000 images with clean labels in validation set are randomly selected as the meta-data set. We adopt a Wide ResNet-28-10 (WRN-28-10) [63] for uniform noise and ResNet-32 [61] for flip noise as our classifier network models[4].

The comparison methods include: **BaseModel**, referring to the similar classifier network utilized in our method, while directly trained on the biased training data; the robust learning methods **Reed** [58], **S-Model** [15] , **SPL** [26], **Focal Loss** [25], **Co-teaching** [19], **D2L** [59]; **Fine-tuning**, fine-tuning the result of **BaseModel** on the meta-data with clean labels to further enhance its performance; typical meta-learning methods **MentorNet** [21], **L2RW** [1], **GLC** [18]. We also trained the baseline network only on 1000 meta-images. The performance are evidently worse than the proposed method due to the neglecting of the knowledge underlying large amount of training samples. We thus have not involved its results in comparison.

All the baseline networks were trained using SGD with a momentum 0.9, a weight decay 5×10^{-4} and an initial learning rate 0.1. The learning rate of classifier network is divided by 10 after 36 epoch and 38 epoch (for a total of 40 epoches) in uniform noise, and after 40 epoch and 50 epoch (for a total of 60 epoches) in flip noise. The learning rate of WN-Net is fixed as 10^{-3}. We repeated the experiments 5 times with different random seeds for network initialization and label noise generation.

We report the accuracy averaged over 5 repetitions for each series of experiments and each competing method in Tables 2 and 3. It can be observed that our method gets the best performance across almost all datasets and all noise rates, except the second for 40% Flip noise. At 0% noise cases (unbiased ones), our method performs only slightly worse than the BaseModel. For other corrupted label cases, the superiority of our method is evident. Besides, it can be seen that the performance gaps between ours and all other competing methods increase as the noise rate is increased from 40% to 60% under uniform noise. Even with 60% label noise, our method can still obtain a relatively high classification accuracy, and attains more than 15% accuracy gain compared with the second best result for CIFAR100 dataset, which indicates the robustness of our methods in such cases.

Fig. 4 shows the performance comparison between WRN-28-10 and ResNet32 under fixed flip noise setting. We can observe that the performance gains for our method and BaseModel between two networks takes the almost same value. It implies that the performance improvement of our method is not dependent on the selection of the classifier network architectures.

As shown in Fig. 1(e), the shape of the learned weight function depicts as monotonic decreasing, complying with the traditional optimal setting to this bias condition, i.e., imposing smaller weights on samples with relatively large losses to suppress the effect of corrupted labels. Furthermore, we plot the weight distribution of clean and noisy training samples in Fig. 5. It can be seen that almost all

---

[4]We have tried different classifier network architectures as classifier networks under each noise setting to show our algorithm is suitable to different deep learning architectures. We show this effect in Fig.4, verifying the consistently good performance of our method in two classifier network settings.
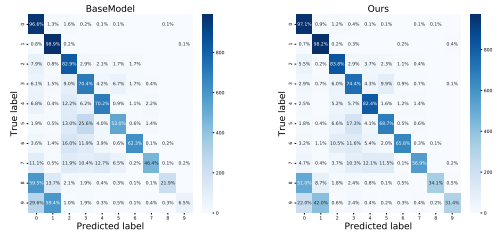
Figure 3: Confusion matrices for the Basemodel and ours on long-tailed CIFAR-10 with imbalance factors 200.
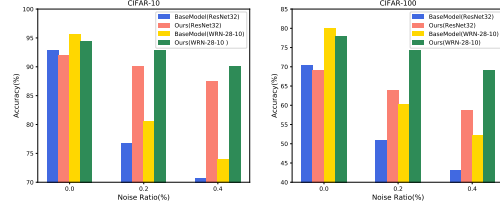

Figure 4: Performance comparison for different classifier networks (WRN-28-10 and ResNet32) under CIFAR flip noise.

Table 3: Test accuracy comparison on CIFAR-10 and CIFAR-100 of ResNet-32 with varying noise rates under flip noise.

Datasets / Noise Rate		BaseModel	Reed-Hard	S-Model	Self-paced	Focal Loss	Co-teaching	D2L	Fine-tuning	MentorNet	L2RW	GLC	Ours
CIFAR-10	0%	92.89±0.32	92.31±0.25	83.61±0.13	88.52±0.21	93.03±0.16	89.87±0.10	92.02±0.14	93.23±0.23	92.13±0.30	89.25±0.37	91.02±0.20	92.04±0.15
	20%	76.83±2.30	88.28±0.36	79.25±0.30	87.03±0.34	86.45±0.19	82.83±0.85	87.66±0.40	82.47±3.64	86.36±0.31	87.86±0.36	89.68±0.33	90.33±0.61
	40%	70.77±2.31	81.06±0.76	75.73±0.32	81.63±0.52	80.45±0.97	75.41±0.21	83.89±0.46	74.07±1.56	81.76±0.28	85.66±0.51	88.92±0.24	87.54±0.23
CIFAR-100	0%	70.50±0.12	69.02±0.32	51.46±0.20	67.55±0.27	70.02±0.53	63.31±0.05	68.11±0.26	70.72±0.22	70.24±0.21	64.11±1.09	65.42±0.23	70.11±0.33
	20%	50.86±0.27	60.27±0.76	45.45±0.25	63.63±0.30	61.87±0.30	54.13±0.55	63.48±0.53	56.98±0.50	61.97±0.47	57.47±1.16	63.07±0.53	64.22±0.28
	40%	43.01±1.16	50.40±1.01	43.81±0.15	53.51±0.53	54.13±0.40	44.85±0.81	51.83±0.33	46.37±0.25	52.66±0.56	50.98±1.55	62.22±0.62	58.64±0.47

large weights belongs to clean samples, and the noisy samples's weights are smaller than that of clean samples, which implies that the trained Meta-Weight-Net can distinguish clean and noisy images.

Fig. 6 plots the weight variation along with training epochs under 40% noise on CIFAR10 dataset of our method and L2RW. y-axis denotes the differences of weights calculated between adjacent epochs, and x-axis denotes the number of epochs. Ten noisy samples are randomly chosen to compute their mean curve, surrounded by the region illustrating the standard deviations calculated on these samples in the corresponding epoch. It is seen that the weight by our method is continuously changed, gradually stable along iterations, and finally converges. As a comparison, the weight during the learning process of L2RW fluctuates relatively more wildly. This could explain the consistently better performance of our method as compared with this competing method.

4.3 Experiments on Clothing1M

To verify the effectiveness of the proposed method on real-world data, we conduct experiments on the Clothing1M dataset [64], containing 1 million images of clothing obtained from online shopping websites that are with 14 categories, e.g., T-shirt, Shirt, Knitwear. The labels are generated by using surrounding texts of the images provided by the sellers, and therefore contain many errors. We use the 7k clean data as the meta dataset. Following the previous works [65, 66], we used ResNet-50 pre-trained on ImageNet. For preprocessing, we resize the image to 256×256, crop the middle 224×224 as input, and perform normalization. We used SGD with a momentum 0.9, a weight decay 10^{-3}, and an initial learning rate 0.01, and batch size 32. The learning rate of ResNet-50 is divided by 10 after 5 epoch (for a total 10 epoch), and the learning rate of WN-Net is fixed as 10^{-3}.

The results are summarized in Table. 4. which shows that the proposed method achieves the best performance. Fig. 1(f) plots the tendency curve of the learned MW-Net function, which reveals abundant data insights. Specifically, when the loss is with relatively small values, the weighting function inclines to increase with loss, meaning that it tends to more emphasize hard margin samples with informative knowledge for classification; while when the loss gradually changes large, the weighting function begins to monotonically decrease, implying that it tends to suppress noise labels

Table 4: Classification accuracy (%) of all competing methods on the Clothing1M test set.

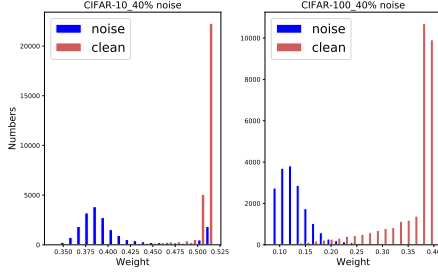| #	Method	Accuracy	#	Method	Accuracy
1	Cross Entropy	68.94	5	Joint Optimization [66]	72.23
2	Bootstrapping [58]	69.12	6	LCCN [67]	73.07
3	Forward [65]	69.84	7	MLNT [68]	73.47
4	S-adaptation [15]	70.36	8	Ours	73.72

8

Figure 5: Sample weight distribution on training data under 40% uniform noise experiments.
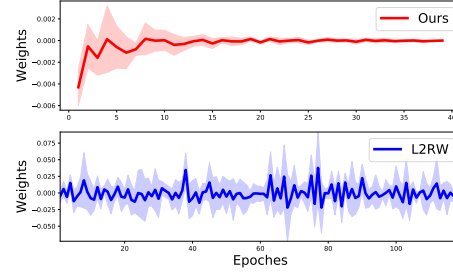

Figure 6: Weight variation curves under 40% uniform noise experiment on CIFAR10 dataset.

samples with relatively large loss values. Such complicated essence cannot be finely delivered by conventional weight functions.

5   Conclusion

We have proposed a novel meta-learning method for adaptively extracting sample weights to guarantee robust deep learning in the presence of training data bias. Compared with current reweighting methods that require to manually set the form of weight functions, the new method is able to yield a rational one directly from data. The working principle of our algorithm can be well explained and the procedure of our method can be easily reproduced. Our empirical results show that the propose method can perform superior in general data bias cases, like class imbalance, corrupted labels, and more complicated real cases. Note that the learned weight function is with an explicit form, and can be easily generalized to other tasks on the similar training data to impose weights on samples (we list some initial attempts in supplementary material). Besides, such an adaptive weight learning approach is hopeful to be employed to other weight setting problems in machine learning, like ensemble methods and multi-view learning.

References

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

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

[3] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, 2008.

[4] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *NeurIPS*, 2017.

[5] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *ICML*, 2017.

[6] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[7] Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *ICLR*, 2018.

[8] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2012.

[9] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 2018.

[10] Wei Bi, Liwei Wang, James T Kwok, and Zhuowen Tu. Learning to predict from crowdsourced data. In *UAI*, 2014.

[11] Junwei Liang, Lu Jiang, Deyu Meng, and Alexander Hauptmann. Learning to detect concepts from webly-labeled video data. In *IJCAI*, 2016.

[12] Bohan Zhuang, Lingqiao Liu, Yao Li, Chunhua Shen, and Ian D Reid. Attend in groups: a weakly-supervised deep learning framework for learning from web data. In *CVPR*, 2017.

[13] Sainbayar Sukhbaatar and Rob Fergus. Learning from noisy labels with deep neural networks. In *ICLR workshop*, 2015.

[14] Samaneh Azadi, Jiashi Feng, Stefanie Jegelka, and Trevor Darrell. Auxiliary image regularization for deep cnns with noisy labels. In *ICLR*, 2016.

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

[16] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. Learning from noisy labels with distillation. In *ICCV*, 2017.

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

[18] 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.

[19] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: robust training deep neural networks with extremely noisy labels. In *NeurIPS*, 2018.

[20] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *NeurIPS*, 2018.

[21] 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.

[22] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[23] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[24] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[25] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[26] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NeurIPS*, 2010.

[27] De la Torre Fernando and J. Black Mkchael. A framework for robust subspace learning. *International Journal of Computer Vision*, 54(1):117–142, 2003.

[28] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *ACM MM*, 2014.

[29] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *NeurIPS*, 2014.

[30] Yixin Wang, Alp Kucukelbir, and David M Blei. Robust probabilistic modeling with bayesian data reweighting. In *ICML*, 2017.

[31] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[32] Lijun Wu, Fei Tian, Yingce Xia, Yang Fan, Tao Qin, Lai Jian-Huang, and Tie-Yan Liu. Learning to teach with dynamic loss functions. In *NeurIPS*, 2018.

[33] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016.

[34] Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. Learning to learn from weak supervision by full supervision. In *NeurIPS Workshop*, 2017.

[35] Luca Franceschi, Paolo Frasconi, Saverio Salzo, and Massimilano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS Workshop*, 2017.

[37] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[38] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2, 2018.

[39] Amir Erfan Eshratifar, David Eigen, and Massoud Pedram. Gradient agreement as an optimization objective for meta-learning. *arXiv preprint arXiv:1810.08178*, 2018.

[40] 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.

[41] Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *ICCV*, 2017.

[42] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ICML*, 2004.

[43] Charles Elkan. The foundations of cost-sensitive learning. In *IJCAI*, 2001.

[44] 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*, 2018.

[45] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 2019.

[46] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *NeurIPS*, 2017.

[47] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[48] Jun Shu, Zongben Xu, and Deyu Meng. Small sample learning in big data era. *arXiv preprint arXiv:1808.04572*, 2018.

[49] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[50] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[51] Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. Fidelity-weighted learning. In *ICLR*, 2018.

[52] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *ICLR*, 2018.

[53] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *NeurIPS*, 2017.

[54] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *CVPR*, 2018.

[55] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016.

[56] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *ICCV*, 2017.

[57] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge J Belongie. Learning from noisy large-scale datasets with minimal supervision. In *CVPR*, 2017.

[58] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR workshop*, 2015.

[59] 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.

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

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

[62] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

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

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

[65] 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.

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

[67] Jiangchao Yao, Hao Wu, Ya Zhang, Ivor W Tsang, and Jun Sun. Safeguarded dynamic label regression for noisy supervision. In *AAAI*, 2019.

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

[69] Julien Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *NeurIPS*, 2013.

Appendix

A Derivation of the Weighting Scheme in Meta-Weight-Net

Recall the update equation of the parameters of Meta-Weight-Net as follows:

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{m} \sum_{i=1}^{m} \nabla_\Theta L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}}. \tag{9}$$

The computation of Eq. (9) in the paper by backpropagation can be understood by the following derivation:

$$\frac{1}{m} \sum_{i=1}^{m} \nabla_\Theta L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}(\Theta)}\Big|_{\hat{\mathbf{w}}^{(t)}} \sum_{j=1}^{n} \frac{\partial \hat{\mathbf{w}}^{(t)}(\Theta)}{\partial \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)} \frac{\partial \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}$$

$$= \frac{-\alpha}{n * m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}} \sum_{j=1}^{n} \frac{\partial L_j^{train}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}$$

$$= \frac{-\alpha}{n} \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^{T} \frac{\partial L_j^{train}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right) \frac{\partial \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}. \tag{10}$$

Let

$$G_{ij} = \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^{T} \frac{\partial L_j^{train}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}}, \tag{11}$$

by substituting Eq. (10) into Eq. (9), we can get:

$$\Theta^{(t+1)} = \Theta^{(t)} + \frac{\alpha\beta}{n} \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} G_{ij} \right) \frac{\partial \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}. \tag{12}$$

B Convergence Proof of Our Method

This section provides the proofs for Theorems 1 and 2 in the paper.

Suppose that we have a small amount of meta (validation) dataset with M samples $\{(x_i^{(m)}, y_i^{(m)}), 1 \le i \le M\}$ with clean labels, and the overall meta loss is,

$$\mathcal{L}^{meta}(\mathbf{w}^*(\Theta)) = \frac{1}{M} \sum_{i=1}^{M} L_i^{meta}(\mathbf{w}^*(\Theta)), \tag{13}$$

where $\mathbf{w}^*$ is the parameter of the classifier network, and Θ is the parameter of the Meta-Weight-Net. Let's suppose we have another N training data, $\{(x_i, y_i), 1 \le i \le N\}$, where $M \ll N$, and the overall training loss is,

$$\mathcal{L}^{train}(\mathbf{w}; \Theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{V}(L_i^{train}(\mathbf{w}); \Theta) L_i^{train}(\mathbf{w}). \tag{14}$$

Lemma 1. *Suppose the meta loss function is Lipschitz smooth with constant L, and $\mathcal{V}(\cdot)$ is differential with a δ-bounded gradient and twice differential with its Hessian bounded by $\mathcal{B}$, and the loss function ℓ have ρ-bounded gradients with respect to training/meta data. Then the gradient of Θ with respect to meta loss is Lipschitz continuous.*

Proof. The gradient of Θ with respect to meta loss can be written as:

$$\nabla_\Theta L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}}$$

$$= \frac{-\alpha}{n} \sum_{j=1}^{n} \left(\frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^{T} \frac{\partial L_j^{train}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right) \frac{\partial \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}. \tag{15}$$

Let $\mathcal{V}_j(\Theta) = \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)$ and G_{ij} being defined in Eq.(11). Taking gradient of Θ in both sides of Eq.(15), we have

$$\nabla_{\Theta^2}^2 L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}} = \frac{-\alpha}{n} \sum_{j=1}^{n} \left[\frac{\partial}{\partial\Theta}(G_{ij})\Big|_{\Theta^{(t)}} \frac{\partial\mathcal{V}_j(\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}} + (G_{ij}) \frac{\partial^2\mathcal{V}_j(\Theta)}{\partial\Theta^2}\Big|_{\Theta^{(t)}} \right].$$

For the first term in the right hand side, we have that

$$\left\| \frac{\partial}{\partial\Theta}(G_{ij})\Big|_{\Theta^{(t)}} \frac{\partial\mathcal{V}_j(\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}} \right\| \leq \delta \left\| \frac{\partial}{\partial\hat{\mathbf{w}}}\left(\frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial\Theta}\Big|_{\Theta^{(t)}} \right)^T\Big|_{\hat{\mathbf{w}}^{(t)}} \frac{\partial L_j^{train}(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right\|$$

$$= \delta \left\| \frac{\partial}{\partial\hat{\mathbf{w}}}\left(\frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}} \frac{-\alpha}{n}\sum_{k=1}^{n} \frac{\partial L_k^{train}(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial\mathcal{V}_k(\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}} \right)^T\Big|_{\hat{\mathbf{w}}^{(t)}} \frac{\partial L_j^{train}(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right\|$$

$$= \delta \left\| \left(\frac{\partial^2 L_i^{meta}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}^2}\Big|_{\hat{\mathbf{w}}^{(t)}} \frac{-\alpha}{n}\sum_{k=1}^{n} \frac{\partial L_k^{train}(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial\mathcal{V}_k(\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}} \right)^T\Big|_{\hat{\mathbf{w}}^{(t)}} \frac{\partial L_j^{train}(\mathbf{w})}{\partial\mathbf{w}} \right\| \leq \alpha L \rho^2 \delta^2, \quad (16)$$

since $\left\| \frac{\partial^2 L_i^{meta}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}^2}\Big|_{\hat{\mathbf{w}}^{(t)}} \right\| \leq L, \left\| \frac{\partial L_j^{train}(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right\| \leq \rho, \left\| \frac{\partial\mathcal{V}_j(\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}} \right\| \leq \delta$. And for the second term we have

$$\left\| (G_{ij}) \frac{\partial^2\mathcal{V}_j(\Theta)}{\partial\Theta^2}\Big|_{\Theta^{(t)}} \right\| = \left\| \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^T \frac{\partial L_j^{train}(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial^2\mathcal{V}_j(\Theta)}{\partial\Theta^2}\Big|_{\Theta^{(t)}} \right\| \leq \mathcal{B}\rho^2, \quad (17)$$

since $\left\| \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^T \right\| \leq \rho, \left\| \frac{\partial^2\mathcal{V}_j(\Theta)}{\partial\Theta^2}\Big|_{\Theta^{(t)}} \right\| \leq \mathcal{B}$. Combining the above two inequalities Eq.(16) (17), we have

$$\left\| \nabla_{\Theta^2}^2 L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}} \right\| \leq \alpha\rho^2(\alpha L\delta^2 + \mathcal{B}). \quad (18)$$

Define $L_V = \alpha\rho^2(\alpha L\delta^2 + \mathcal{B})$, based on Lagrange mean value theorem, we have:

$$\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta_1)) - \nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta_2))\| \leq L_V\|\Theta_1 - \Theta_2\|, \quad for\ all\ \Theta_1, \Theta_2, \quad (19)$$

where $\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta_1)) = \nabla_\Theta L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta_1}$. $\qquad\qquad\square$

Theorem 3. *Suppose the meta loss function is Lipschitz smooth with constant L, and $\mathcal{V}(\cdot)$ is differential with a δ-bounded gradient and twice differential with its Hessian bounded by $\mathcal{B}$, and the loss function ℓ have ρ-bounded gradients with respect to training/meta data. Let the learning rate α_t satisfies $\alpha_t = \min\{1, \frac{k}{T}\}$, for some $k > 0$, such that $\frac{k}{T} < 1$, and $\beta_t, 1 \leq t \leq N$ is a monotone descent sequence, $\beta_1 = \min\{\frac{1}{L}, \frac{c}{\sigma\sqrt{T}}\}$ for some $c > 0$, such that $\frac{\sigma\sqrt{T}}{c} \geq L$. Then Meta-Weight-Net can achieve $\mathbb{E}[\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2] \leq \epsilon$ in $\mathcal{O}(1/\epsilon^2)$ steps. More specifically,*

$$\min_{0 \leq t \leq T} \mathbb{E}[\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2] \leq \mathcal{O}(\frac{C}{\sqrt{T}}), \quad (20)$$

where C is some constant independent of the convergence process, σ is the variance of drawing uniformly mini-batch sample at random.

Proof. The update of Θ in each iteration is as follows:

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta\frac{1}{m}\sum_{i=1}^{m} \nabla_\Theta L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}}. \quad (21)$$

This can be written as:

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta_t\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\Big|_{\Xi_t}. \quad (22)$$

Since the mini-batch Ξ_t is drawn uniformly from the entire data set, we can rewrite the update equationp as:

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta_t[\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) + \xi^{(t)}], \quad (23)$$

14

where $\xi^{(t)} = \nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\big|_{\Xi_t} - \nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))$. Note that $\xi^{(t)}$ are i.i.d random variable with finite variance, since Ξ_t are drawn i.i.d with a finite number of samples. Furthermore, $\mathbb{E}[\xi^{(t)}] = 0$, since samples are drawn uniformly at random. Observe that

$$\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))$$
$$= \left\{\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}))\right\} + \left\{\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\right\}. \quad (24)$$

By Lipschitz smoothness of meta loss function, we have

$$\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}))$$
$$\leq \langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})), \hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})\rangle + \frac{L}{2}\|\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})\|_2^2$$

Since $\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}) = -\alpha_t\frac{1}{n}\sum_{i=1}^{n}\mathcal{V}(L_i^{train}(\mathbf{w}^{(t+1)});\Theta^{(t+1)})\nabla_{\mathbf{w}}L_i^{train}(\mathbf{w})\big|_{\mathbf{w}^{(t+1)}}$ according to Eq.(3),(5), we have

$$\|\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}))\| \leq \alpha_t\rho^2 + \frac{L\alpha_t^2}{2}\rho^2 = \alpha_t\rho^2(1 + \frac{\alpha_t L}{2}) \quad (25)$$

Since $\left\|\frac{\partial L_j^{train}(\mathbf{w})}{\partial\mathbf{w}}\big|_{\mathbf{w}^{(t)}}\right\| \leq \rho$, $\left\|\frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^T\right\| \leq \rho$.

By Lipschitz continuity of $\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))$ according to Lemma 1, we can obtain the following:

$$\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \leq \langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \Theta^{(t+1)} - \Theta^{(t)}\rangle + \frac{L}{2}\|\Theta^{(t+1)} - \Theta^{(t)}\|_2^2$$
$$= \langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), -\beta_t[\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) + \xi^{(t)}]\rangle + \frac{L\beta_t^2}{2}\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) + \xi^{(t)}\|_2^2$$
$$= -(\beta_t - \frac{L\beta_t^2}{2})\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2 + \frac{L\beta_t^2}{2}\|\xi^{(t)}\|_2^2 - (\beta_t - L\beta_t^2)\langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \xi^{(t)}\rangle.$$

Thus Eq.(24) satifies

$$\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \leq \alpha_t\rho^2(1 + \frac{\alpha_t L}{2}) - (\beta_t - \frac{L\beta_t^2}{2})$$
$$\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2 + \frac{L\beta_t^2}{2}\|\xi^{(t)}\|_2^2 - (\beta_t - L\beta_t^2)\langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})), \xi^{(t)}\rangle. \quad (26)$$

Rearranging the terms, we can obtain

$$(\beta_t - \frac{L\beta_t^2}{2})\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2 \leq \alpha_t\rho^2(1 + \frac{\alpha_t L}{2}) + \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}))$$
$$+ \frac{L\beta_t^2}{2}\|\xi^{(t)}\|_2^2 - (\beta_t - L\beta_t^2)\langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})), \xi^{(t)}\rangle. \quad (27)$$

Summing up the above inequalities and rearranging the terms, we can obtain

$$\sum_{t=1}^{T}(\beta_t - \frac{L\beta_t^2}{2})\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2 \leq \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)})(\Theta^{(1)}) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(T+1)}(\Theta^{(T+1)}))$$
$$+ \sum_{t=1}^{T}\alpha_t\rho^2(1 + \frac{\alpha_t L}{2}) - \sum_{t=1}^{T}(\beta_t - L\beta_t^2)\langle\nabla\mathcal{L}^{meta}(\Theta^{(t)}), \xi^{(t)}\rangle + \frac{L}{2}\sum_{t=1}^{T}\beta_t^2\|\xi^{(t)}\|_2^2$$
$$\leq \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^{T}\alpha_t\rho^2(1 + \frac{\alpha_t L}{2}) - \sum_{t=1}^{T}(\beta_t - L\beta_t^2)\langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \xi^{(t)}\rangle + \frac{L}{2}\sum_{t=1}^{T}\beta_t^2\|\xi^{(t)}\|_2^2, \quad (28)$$

Taking expectations with respect to $\xi^{(N)}$ on both sides of Eq. 28, we can then obtain:

$$\sum_{t=1}^{T}(\beta_t - \frac{L\beta_t^2}{2})\mathbb{E}_{\xi^{(N)}}\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2 \leq \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^{T}\alpha_t\rho^2(1 + \frac{\alpha_t L}{2}) + \frac{L\sigma^2}{2}\sum_{t=1}^{T}\beta_t^2, \quad (29)$$

15

since $\mathbb{E}_{\xi^{(N)}}\langle\nabla\mathcal{L}^{meta}(\Theta^{(t)}),\xi^{(t)}\rangle = 0$ and $\mathbb{E}[\|\xi^{(t)}\|_2^2] \leq \sigma^2$, where σ^2 is the variance of $\xi^{(t)}$. Furthermore, we can deduce that

$$\min_t\mathbb{E}[\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2] \leq \frac{\sum_{t=1}^T(\beta_t - \frac{L\beta_t^2}{2})\mathbb{E}_{\xi^{(N)}}\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2}{\sum_{t=1}^T(\beta_t - \frac{L\beta_t^2}{2})}$$

$$\leq \frac{1}{\sum_{t=1}^T(2\beta_t - L\beta_t^2)}\left[2\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^T\alpha_t\rho^2(2+\alpha_t L) + L\sigma^2\sum_{t=1}^T\beta_t^2\right]$$

$$\leq \frac{1}{\sum_{t=1}^T 2\beta_t}\left[2\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^T\alpha_t\rho^2(2+\alpha_t L) + L\sigma^2\sum_{t=1}^T\beta_t^2\right]$$

$$\leq \frac{1}{2T\beta_1}\left[2\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \alpha_1\rho^2 T(2+L) + L\sigma^2 T\beta_1^2\right]$$

$$= \frac{\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{T}\frac{1}{\beta_1} + \frac{\alpha_1\rho^2(2+L)}{\beta_1} + \frac{LT\sigma^2}{2}\beta_1$$

$$= \frac{\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{T}\max\{L,\frac{\sigma\sqrt{T}}{c}\} + \min\{1,\frac{k}{T}\}\max\{L,\frac{\sigma\sqrt{T}}{c}\}\rho^2(2+L) + L\sigma^2\frac{c}{\sigma\sqrt{T}}$$

$$\leq \frac{\sigma\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{c\sqrt{T}} + \frac{k\sigma\rho^2(2+L)}{c\sqrt{T}} + \frac{L\sigma c}{\sqrt{T}} = \mathcal{O}(\frac{1}{\sqrt{T}}). \tag{30}$$

Therefore, we can conclude that our algorithm can always achieve $\min_{0\leq t\leq T}\mathbb{E}[\|\nabla\mathcal{L}^{meta}(\Theta^{(t)})\|_2^2] \leq \mathcal{O}(\frac{1}{\sqrt{T}})$ in T steps, and this finishes our proof of Theorem 3. $\square$

Lemma 2. *(Lemma A.5 in [69]) Let $(a_n)_{n\leq 1}, (b_n)_{n\leq 1}$ be two non-negative real sequences such that the series $\sum_{i=1}^\infty a_n$ diverges, the series $\sum_{i=1}^\infty a_n b_n$ converges, and there exists $K > 0$ such that $|b_{n+1} - b_n| \leq Ka_n$. Then the sequences $(b_n)_{n\leq 1}$ converges to 0.*

Theorem 4. *Suppose the loss function ℓ is Lipschitz-smooth with constant L, and have ρ-bounded gradients with respect to training/meta data. Let the learning rate α_t satisfies $\alpha_t = \min\{1,\frac{k}{T}\}$, for some $k > 0$, such that $\frac{k}{T} < 1$, and $\beta_t, 1 \leq t \leq N$ is a monotone descent sequence. Then*

$$\lim_{t\to\infty}\mathbb{E}[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] = 0. \tag{31}$$

Proof. It is easy to conclude that α_t satisfy $\sum_{t=0}^\infty\alpha_t = \infty, \sum_{t=0}^\infty\alpha_t^2 < \infty$. Recall the update of $\mathbf{w}$ in each iteration as follows:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha\frac{1}{n}\sum_{i=1}^n\mathcal{V}(L_i^{train}(\mathbf{w}^{(t)});\Theta^{(t+1)})\nabla_{\mathbf{w}}L_i^{train}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}}. \tag{32}$$

It can be written as:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha_t\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})|_{\Psi_t}, \tag{33}$$

where $\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta) = \frac{1}{n}\sum_{i=1}^n\mathcal{V}(L_i^{train}(\mathbf{w}^{(t)});\Theta)\nabla_{\mathbf{w}}L_i^{train}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}}$. Since the mini-batch Ψ_t is drawn uniformly at random, we can rewrite the update equation as:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha_t[\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}) + \psi^{(t)}], \tag{34}$$

where $\psi^{(t)} = \nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})|_{\Psi_t} - \nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})$. Note that $\psi^{(t)}$ is i.i.d. random variable with finite variance, since Ψ_t are drawn i.i.d. with a finite number of samples. Furthermore, $\mathbb{E}[\psi^{(t)}] = 0$, since samples are drawn uniformly at random, and $\mathbb{E}[\|\psi^{(t)}\|_2^2] \leq \sigma^2$.

The objective function $\mathcal{L}^{train}(\mathbf{w};\Theta)$ defined in Eq. 14 can be easily checked to be Lipschitz-smooth with constant L, and have ρ-bounded gradients with respect to training data. Therefore, we have:

$$\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+1)}) - \mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})$$

$$\leq \langle \nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}), \mathbf{w}^{(t+1)} - \mathbf{w}^{(t)} \rangle + \frac{L}{2}\|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\|_2^2$$

$$= \langle \nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}), -\alpha_t[\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}) + \psi^{(t)}]\rangle + \frac{L\alpha_t^2}{2}\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}) + \psi^{(t)}\|_2^2$$

$$= -(\alpha_t - \frac{L\alpha_t^2}{2})\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2 + \frac{L\alpha_t^2}{2}\|\psi^{(t)}\|_2^2 - (\alpha_t - L\alpha_t^2)\langle\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}),\psi^{(t)}\rangle. \quad (35)$$

Taking expectation of both sides of (35) and since $\mathbb{E}[\psi^{(t)}] = 0$, we have

$$\mathbb{E}[\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+1)})] - \mathbb{E}[\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})] \leq -\alpha_t\mathbb{E}[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] + \frac{L\alpha_t^2}{2}\{\mathbb{E}[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] + \mathbb{E}[\|\psi^{(t)}\|_2^2]\}$$

Summing up the above inequalities over $t = 1, ..., \infty$ in both sides, we obtain

$$\sum_{t=1}^{\infty}\alpha_t\mathbb{E}[\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] \leq \sum_{t=1}^{\infty}\frac{L\alpha_t^2}{2}\{\mathbb{E}[\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] + \mathbb{E}[\|\psi^{(t)}\|_2^2]\} + \mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(1)};\Theta^{(2)})] - \lim_{T\to\infty}\mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t+1)})]$$

$$\leq \sum_{t=1}^{\infty}\frac{L\alpha_t^2}{2}\{\rho^2 + \sigma^2\} + \mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(1)};\Theta^{(2)})] \leq \infty.$$

The above inequality implies that $\sum_{t=1}^{\infty}\alpha_t\mathbb{E}[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] < \infty$, since $\sum_{t=0}^{\infty}\alpha_t^2 < \infty$. By Lemma 2, to substantiate $\lim_{t\to\infty}\mathbb{E}[\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] = 0$, since $\sum_{t=0}^{\infty}\alpha_t = \infty$, it only needs to prove:

$$\left|\mathbb{E}[\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)})\|_2^2] - \mathbb{E}[\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2]\right| \leq C\alpha_k, \quad (36)$$

for some constant C. Based on the inequality:

$$|(\|a\| + \|b\|)(\|a\| - \|b\|)| \leq \|a + b\|\|a - b\|, \quad (37)$$

we then have:

$$\left|\mathbb{E}[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)})\|_2^2] - \mathbb{E}[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2]\right|$$

$$= \left|\mathbb{E}\left[(\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)})\|_2 + \|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2)(\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)})\|_2 - \|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2)\right]\right|$$

$$\leq \mathbb{E}\left[\left|\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+1)})\|_2 + \|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t)})\|_2\right|\left|(\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)})\|_2 - \|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2)\right|\right]$$

$$\leq \mathbb{E}\left[\left\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)}) + \nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\right\|_2 \left\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)}) - \nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\right\|_2\right]$$

$$\leq \mathbb{E}\left[(\left\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)})\right\|_2 + \left\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\right\|_2)\left\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+2)}) - \nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\right\|_2\right]$$

$$\leq 2L\rho\mathbb{E}\left[\|(\mathbf{w}^{(t+1)},\Theta^{(t+2)}) - (\mathbf{w}^{(t)},\Theta^{(t+1)})\|_2\right]$$

$$\leq 2L\rho\alpha_t\beta_t\mathbb{E}\left[\left\|\left(\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}) + \psi^{(t)}, \nabla\mathcal{L}^{meta}(\Theta^{(t+1)}) + \xi^{(t+1)}\right)\right\|_2\right]$$

$$\leq 2L\rho\alpha_t\beta_t\mathbb{E}\left[\sqrt{\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}) + \psi^{(t)}\|_2^2} + \sqrt{\|\nabla\mathcal{L}^{meta}(\Theta^{(t+1)}) + \xi^{(t+1)}\|_2^2}\right]$$

$$\leq 2L\rho\alpha_t\beta_t\sqrt{\mathbb{E}\left[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)}) + \psi^{(t)}\|_2^2\right] + \mathbb{E}\left[\|\nabla\mathcal{L}^{meta}(\Theta^{(t+1)}) + \xi^{(t+1)}\|_2^2\right]}$$

$$\leq 2L\rho\alpha_t\beta_t\sqrt{\mathbb{E}\left[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2\right] + \mathbb{E}\left[\|\psi^{(t)}\|_2^2\right] + \mathbb{E}\left[\|\xi^{(t+1)}\|_2^2\right] + \mathbb{E}\left[\|\nabla\mathcal{L}^{meta}(\Theta^{(t+1)})\|_2^2\right]}$$

$$\leq 2L\rho\alpha_t\beta_t\sqrt{2\sigma^2 + 2\rho^2}$$

$$\leq 2\sqrt{2(\sigma^2 + \rho^2)}L\rho\beta_1\alpha_t. \quad (38)$$

According to the above inequality, we can conclude that our algorithm can achieve

$$\lim_{t\to\infty}\mathbb{E}[\|\nabla\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t+1)})\|_2^2] = 0. \quad (39)$$

The proof is completed. $\square$

C Experimentalp Details on Meta-Weight-Net

To match the original training step size, in our experiment, we can consider normalizing the weights of all examples in a training batch so that they sum up to one. In other words, we choose to have a hard constraint within the set $\|\mathcal{V}(\ell;\Theta)\|_1 = 1$, and the normalized weight

$$\eta_i^{(t)} = \frac{\mathcal{V}^{(t)}(L_i;\Theta)}{\sum_j \mathcal{V}^{(t)}(L_j;\Theta) + \delta(\sum_i \mathcal{V}^{(t)}(L_j;\Theta))}, \quad (40)$$

where the function of $\delta(\cdot)$ is to prevent the degeneration case when all $\mathcal{V}^{(t)}(L_j;\Theta)$ in a mini-batch are zeros, i.e. $\delta(a) = \tau$. τ denotes a constant grater than 0, if $a = 0$; and equal to 0 otherwise. With batch normalization, we effectively cancel the learning rate of Meta-Weight-Net, and it works well with a fixed learning rate.

D MLP architecture of Meta-Weight-Net

We actually have tried different MLP architecture settings in experiments. The right table depicts some representative results under 6 different structures, with different depths and widths. It can be seen that varying MLP settings have unsubstantial effects to the final result. We thus prefer to use the simple and shallow one.

Table 5: Test accuracy on CIFAR-10 and CIFAR-100 of different MW-Nets.

architcture	Imbalance (factor 100)		Uniform noise (40%)		Flip noise (40%)	
	CIFAR10	CIFAR100	CIFAR10	CIFAR100	CIFAR10	CIFAR100
1-50-1	73.50	41.87	89.01	67.63	87.38	57.83
1-100-1	75.21	42.09	89.27	67.73	87.54	58.64
1-200-1	74.70	41.72	89.58	67.84	87.74	58.41
1-100-100-1	75.01	41.97	89.09	66.48	87.28	57.39
1-10-10-1	74.71	41.94	89.10	66.53	87.58	57.11
1-10-10-10-1	74.96	42.31	88.82	66.67	87.36	57.29

E Complexity Analysis of The Proposed Algorithm

Our Meta-PGC algorithm can be roughly regarded as that requires an extra full forward and backward passes of the network on training data (step 5 in algorithm) and an extra full forward and backward passes of the network of meta data (step 6 in algorithm) in the presence of the normal classifier network' parameters update (step 7 in algorithm). Therefore compared to regular training, our method needs approximately $3 \times$ training time in each iteration. It is suggest to let batch size of meta data less than or equal to batch size of training data, which avoids GPU memory increase and speeds up the learning process.

F Robustness Towards Label Noise Overfitting Issue

Fig. 7 plots the tendency curves of the mini-batch training accuracy calculated on noisy training set in experiments, as well as those calculated simultaneously on clean test data during learning iterations. From the figure, we can easily find that the BaseModel can easily overfit to the noisy labels contained in the training set, whose test accuracy quickly degrades after the first learning rate decays. While our method and L2RW are less prone to such overfitting issue, and they retain the similar test accuracy until termination. Especially, throughout all our experiments, we find that our method can converge significantly faster than the BaseModel and L2RW methods[5], as clearly shown in Fig.7, and get the peak performance at around 40 epochs, as compared with 120 epochs required for the other two methods, as shown in Fig. 7(a). We thus only report our results at 40 epochs in the figure for other experiments.

G Generalization of Meta-Weighting-Net

Different from current re-weighting methods, our Meta-Weighting-Net can directly generalize to new tasks without need to retrain Meta-Weight-Net with extra meta data. To validate this, we use the trained Meta-Weighting-Net on CIFAR-10 dataset to help the target CIFAR-100 dataset. This process is reasonable because there are no common categories between CIFAR-10 and CIFAR-100

---

[5]Since our method is inspired by L2RW [1], we compare this method, as well as the BaseModel, for better illustration. The uniform noise experiment setting is the same as L2RW [1], and thus the iteration steps of BaseModel and L2RW follows the original setting.

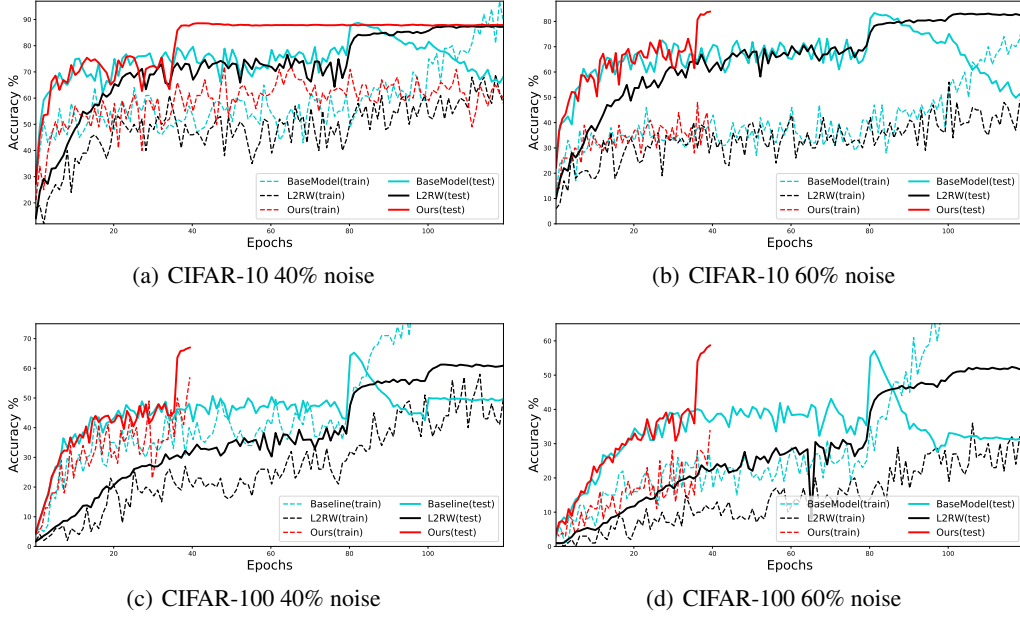|	
(a) CIFAR-10 40% noise	(b) CIFAR-10 60% noise
(c) CIFAR-100 40% noise	(d) CIFAR-100 60% noise

Figure 7: Training and test accuracy changing curves in **uniform noise** cases of CIFAR-10 and CIFAR-100 datasets. Solid and dotted curves denote the test and training accuracies, respectively. Our method and L2RW are less prone to overfit label noises, while our method can converge faster at around 40 epoch as shown in Fig. 7(a). We thus terminate our method in 40 epoch in other experiments.

dataset [6]. Following the similar setting as [49], we define the meta-training process as training the Meta-Weight-Net on CIFAR-10 with training, meta, test data partition, and the meta-test process as applying the learned Meta-Weighting-Net to the target CIFAR-100 dataset with training, test data without meta data partition [7].

As shown in Table. 6, *Transferred Meta-Weight-Net* represents that we meta-trained Meta-Weighting-Net on CIFAR-10 datatset, and then applied it to direct output sample weights on CIFAR-100 dataset without meta data learning. Compared with directly using meta data to train the Meta-Weighting-Net on CIFAR-100 dataset (*Meta-Weight-Net*), *Transferred Meta-Weight-Net* obtained comparable performance on class imbalance and corrupted labels experiments, respectively, and has superior performance than current meta-learning method[8] (MentorNet [21] and L2RW [1]).

Table 6: Test accuracy (%) of Transferred Meta-Weight-Net from CIFAR-10 to CIFAR-100 on class imbalance and corrupted experiments compared with Meta-Weight-Net, MentorNet and L2RW using meta-data on CIFAR-100 dataset ('—' means MentorNet does not handle with class imbalance case).

Experiment Setting	Class Imbalance						Uniform Noise			Flip Noise		
	200	100	50	20	10	1	0	0.4	0.6	0	0.2	0.4
L2RW	33.38	40.23	44.44	51.64	53.73	64.11	72.99	60.79	48.15	64.11	57.47	50.98
MentorNet	—	—	—	—	—	—	73.26	61.39	36.87	70.24	61.97	52.66
Meta-Weight-Net	37.91	42.09	46.74	54.37	58.46	70.37	78.76	67.73	58.75	70.11	64.22	58.64
Transferred Meta-Weight-Net	35.82	41.37	46.24	53.32	58.23	70.35	78.28	67.57	58.45	69.71	63.97	57.86

---

[6]http://www.cs.toronto.edu/ kriz/cifar.html

[7]The parameters setting follows the paper's setting

[8]Since L2RW is not able to output an explicit weight function, so that it cannot be generalized to implement sample weighting on new tasks, we use meta-data to training it for fair comparison.

H Confusion Matrices for Class Imbalance and Corrupted Labels

We demonstrate confusion matrices of Baseline and our algorithm on long-tailed CIFAR-10 dataset for class imbalance and corrupted labels experiments, as shown in Fig. 2-4.
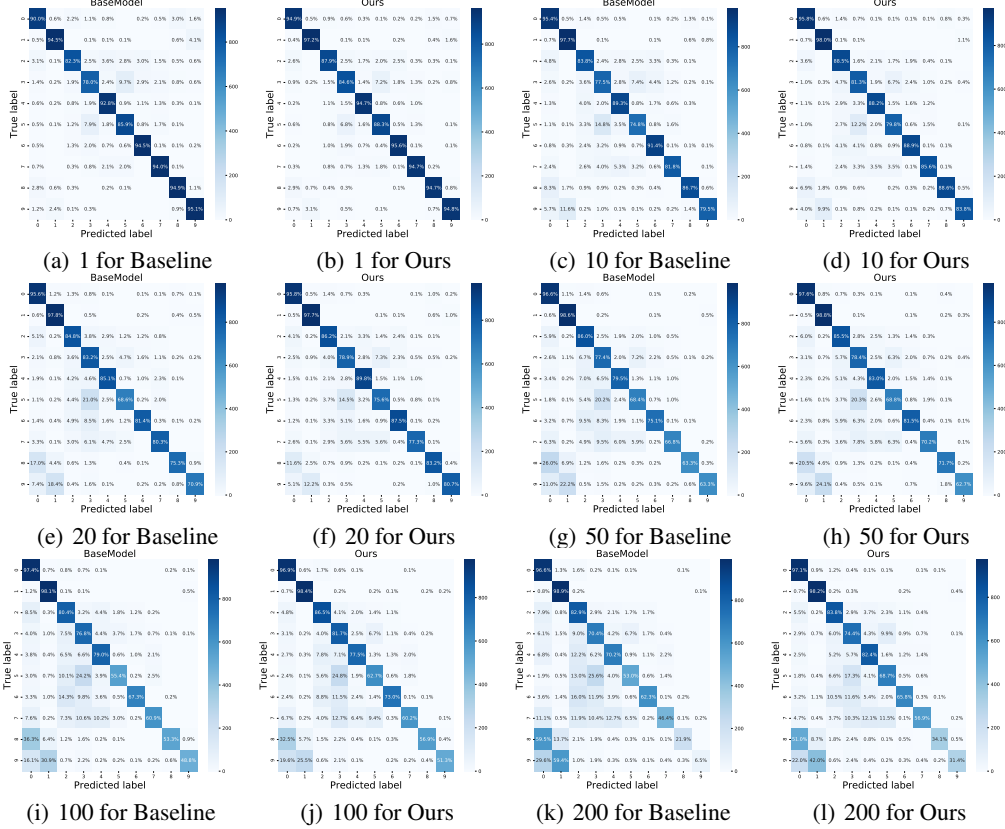

(a) 1 for Baseline	(b) 1 for Ours	(c) 10 for Baseline	(d) 10 for Ours
(e) 20 for Baseline	(f) 20 for Ours	(g) 50 for Baseline	(h) 50 for Ours
(i) 100 for Baseline	(j) 100 for Ours	(k) 200 for Baseline	(l) 200 for Ours

Figure 8: Confusion matrices on long-tailed CIFAR-10 with imbalance factors ranging from 1 to 200.

| (a) 40% for Baseline | (b) 40% for Ours | (c) 60% for Baseline | (d) 60% for Ours |
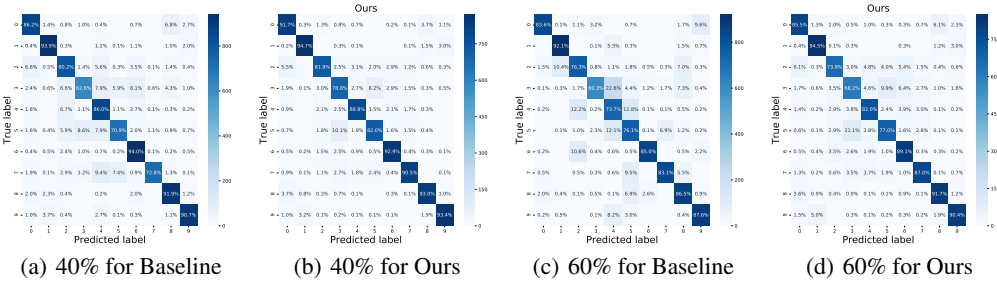
Figure 9: Confusion matrices on CIFAR-10 dataset with varying noise rates under **uniform noise**.

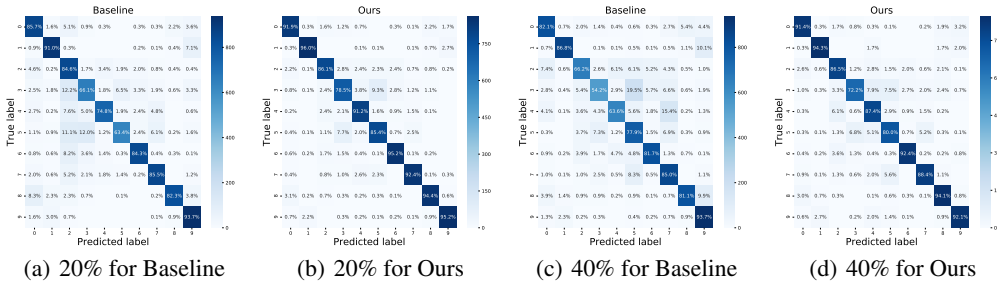| (a) 20% for Baseline | (b) 20% for Ours | (c) 40% for Baseline | (d) 40% for Ours |

Figure 10: Confusion matrices on CIFAR-10 dataset with varying noise rates under **flip noise**.

I Convergence Verification for The Training loss and Meta Loss

To validate the convergence results obtained in Theorem 1 and 2 in the paper, we plot the changing tendency curves of training and meta losses with the number of epochs in our experiments, as shown in Fig. 11 - 13. The convergence tendency can be easily observed in the figures, substantiating the properness of the theoretical results in two theorems.
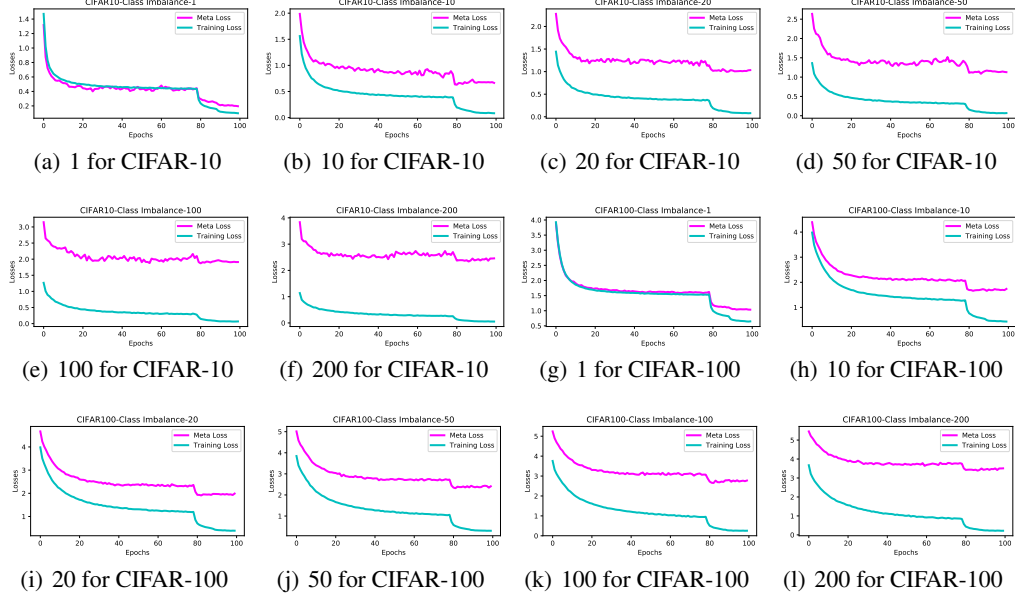

(a) 1 for CIFAR-10 (b) 10 for CIFAR-10 (c) 20 for CIFAR-10 (d) 50 for CIFAR-10

(e) 100 for CIFAR-10 (f) 200 for CIFAR-10 (g) 1 for CIFAR-100 (h) 10 for CIFAR-100

(i) 20 for CIFAR-100 (j) 50 for CIFAR-100 (k) 100 for CIFAR-100 (l) 200 for CIFAR-100

Figure 11: Training and meta loss tendency curves on long-tailed CIFAR with imbalance factors ranging from 1 to 200.
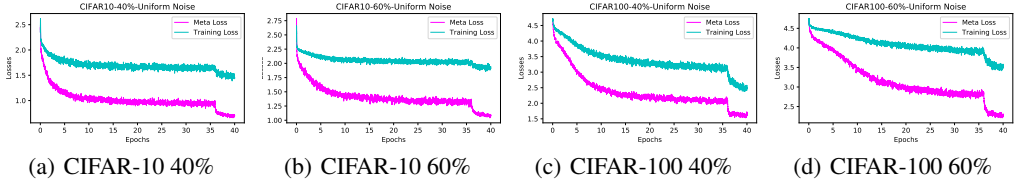

(a) CIFAR-10 40% (b) CIFAR-10 60% (c) CIFAR-100 40% (d) CIFAR-100 60%

Figure 12: Training and meta loss tendency curves on CIFAR dataset with varying noise rates under **uniform noise**.

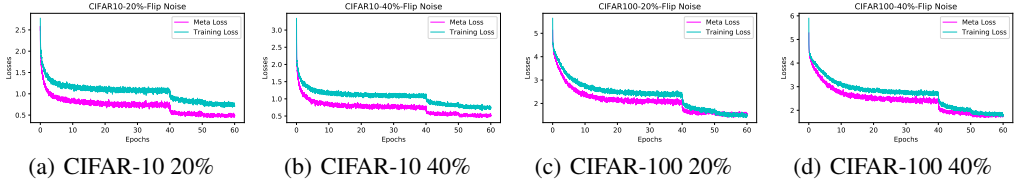(a) CIFAR-10 20% (b) CIFAR-10 40% (c) CIFAR-100 20% (d) CIFAR-100 40%

Figure 13: Training and meta loss tendency curves on CIFAR dataset with varying noise rates under **flip noise**.