CMW-Net: Learning a Class-Aware Sample Weighting Mapping for Robust Deep Learning

Jun Shu, Xiang Yuan, Deyu Meng, and Zongben Xu

Abstract—Modern deep neural networks (DNNs) can easily overfit to biased training data containing corrupted labels or class imbalance. Sample re-weighting methods are popularly used to alleviate this data bias issue. Most current methods, however, require to manually pre-specify the weighting schemes as well as their additional hyper-parameters relying on the characteristics of the investigated problem and training data. This makes them fairly hard to be generally applied in practical scenarios, due to their significant complexities and inter-class variations of data bias situations. To address this issue, we propose a meta-model capable of adaptively learning an explicit weighting scheme directly from data. Specifically, by seeing each training class as a separate learning task, our method aims to extract an explicit weighting function with sample loss and task/class feature as input, and sample weight as output, expecting to impose adaptively varying weighting schemes to different sample classes based on their own intrinsic bias characteristics. Synthetic and real data experiments substantiate the capability of our method on achieving proper weighting schemes in various data bias cases, like the class imbalance, feature-independent and dependent label noise scenarios, and more complicated bias scenarios beyond conventional cases. Besides, the task-transferability of the learned weighting scheme is also substantiated, by readily deploying the weighting function learned on relatively smaller-scale CIFAR-10 dataset on much larger-scale full WebVision dataset. A performance gain can be readily achieved compared with previous state-of-the-art ones without additional hyper-parameter tuning and meta gradient descent step. The general availability of our method for multiple robust deep learning issues, including partial-label learning, semi-supervised learning and selective classification, has also been validated. Code for reproducing our experiments is available at https://github.com/xjtushujun/CMW-Net.

Index Terms—Meta Learning, sample re-weighting, noisy labels, class imbalance, semi-supervised learning, partial-label learning.

1 Introduction

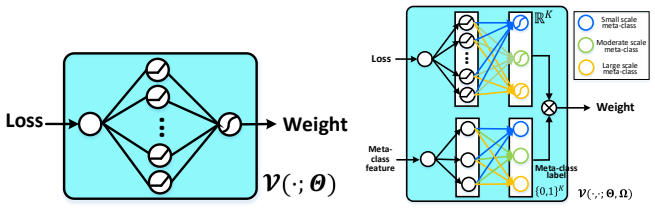
Deep neural networks (DNNs), equipped with highly parameterized structures for modeling complex input patterns, have recently obtained impressive performance on various applications, e.g., computer vision [1], natural language processing [2], speech processing [3], etc. These successes largely attribute to many large-scale paired sample-label datasets expected to properly and sufficiently simulate the testing/evaluating environments. However, in most real applications, collecting such large-scale supervised datasets is notoriously costly, and always highly dependent on a rough crowdsourcing system or search engine. This often makes the training datasets error-prone, with unexpected data bias from the real testing distributions.

This distribution mismatch issue could have many different forms. For example, the collected training sets are often class imbalanced [4], [5]. Actually, real-world datasets are usually depicted as skewed distributions. Specifically, the frequency distribution of visual categories in our daily life is generally long-tailed, with a few common classes and many more rare ones. This often leads to a mismatch between collected datasets with long-tailed class distributions for training a machine learning model and our expectation on the model to perform well on all classes. Another popular data bias is the noisy label case [6], [7], [8], [9]. Even the most celebrated datasets collected from a crowdsourcing system with expert knowledge [10], like ImageNet, have been demonstrated to contain harmful examples with unreliable labels [11], [12], [13]. To mitigate the high labeling cost, it has received increasing attention to collect web images by search engines [14]. Though cheaper and easier to obtain training data, it often yields inevitable noisy labels due to the error-prone automatic tagging system [15], [16].

The overparameterized DNNs tend to suffer significantly from overfitting on these biased training data, then conducting their poor performance in generalization. This robust deep learning issue has been theoretically illustrated in multiple literatures [17], [18] and gradually attracted more attention in the field. Recently, various methods have been proposed to deal with such biased training data. Readers can refer to [19], [20], [21], [22], [23], [24] for an overall review. In this paper, we focus on the sample re-weighting approach, which is a commonly used strategy against such data bias issue and has been widely investigated started at 1950s [25].

The sample re-weighting approach [9], [26] attempts to assign a weight to each example and minimize the corresponding weighted training loss to learn a classifier model. The example weights are typically calculated based on the training loss. More specifically, the learning methodology of sample re-weighting is to design a weighting function mapping from training loss to sample weight, and then iterates between calculating weights from current sample loss values and minimizing weighted training loss for classifier updating (that's why the method is called "re-weighting"). However, there exist two entirely contrary ideas for constructing such a loss-weight mapping. In class imbalanced problems, the function is generally set as monotonically increasing, aiming to enforce the learning to more emphasize samples with larger loss values since they are more like to be the minority

• *Jun Shu, Xiang Yuan, Deyu Meng (corresponding author) and Zongben Xu are with School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi'an Jiaotong University, Shaanxi, P.R.China.*
Email: xjtushujun,relojeffrey@gmail.com, dymeng,zbxu@mail.xjtu.edu.cn.



(a) MW-Net (b) CMW-Net

Fig. 1. The architectures of (a) MW-Net and (b) CMW-Net.

class. Typical methods include Boosting and AdaBoost [27], [28], hard negative mining [29] and focal loss [30]. But in noisy label problems, the function is more commonly set as monotonically decreasing, i.e., taking samples with smaller loss values as more important ones, since they are more likely to be high-confident ones with clean labels. Typical methods include self-paced learning (SPL) [31], iterative reweighting [32] and multiple variants [33], [34], [35].

Although these pre-defined weighting schemes has substantiated to help improve the robustness of a learning algorithm on certain data bias scenarios, they still have evident deficiencies in practice. On the one hand, they need to manually preset a specific form of weighting function based on certain assumptions of training data. This, however, tends to be infeasible when we know insufficient knowledge underlying data or the label conditions are too complicated, like the case that the training set is both imbalanced and label-noisy. On the other hand, even when we properly specify certain weighting schemes, like focal loss [30] or SPL [31], 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 in real problems.

To alleviate the above issues, our earlier work attempts to parameterize the weighting function as an MLP (multilayer perceptron) network with one hidden layer called MW-Net [9], as depicted in Fig. 1(a), which is theoretically capable of dealing with such weighting function approximation problem [36]. Instead of assuming a pre-defined weighting scheme, MW-Net can automatically learn a suitable weighting strategy from data for the training data at hand. Experiments on datasets with class imbalanced or noisy labels show that the automatically learned weighting schemes are consistent with the properly defined ones as traditional.

However, when encountered with complicated data bias scenarios, especially those with heterogeneous bias situations, MW-Net inclines to significantly lose efficacy in such more practical cases. This issue is mainly attributed to the fact that MW-Net assumes one unique weighting function shared by all classes of training data, implying that different classes should possess consistent bias. Unfortunately, this strong assumption is often invalid. As shown in Fig. 2(a), we plot the empirical probability density function (pdf) of training loss [1] of each class for four kinds of common data bias types, including class imbalance, symmetric noise, asymmetric noise and more realistic feature-dependent noise [38]. It can be seen that only in symmetric noise case, each class is with an approximately homoscedastic training loss distribution for all classes. Thus MW-Net can properly learn a suitable weighting scheme and naturally distinguish clean and noisy samples in this instance, as shown in Fig.2(b) (second column). While for all other cases, MW-Net can not finely adapt to the heteroscedastic loss distributions across classes. Particularly, for asymmetric and feature-dependent noise cases, MW-Net easily learns a monotonically increasing weighting function for all classes and fails to entirely distinguish clean and noisy samples as shown in Fig.2(b) (third and fourth columns). The real-world biased datatsets (like WebVision [14]), however, always possess even more inter-class heterogenous bias configurations than these simulated biased ones. It is thus fairly insufficient and improper to employ only one single weighting function to deal with such complicated biased datasets.

This issue can be more intrinsically analyzed under the framework of meta-learning. From the task-distribution view, a meta-learning approach attempts to learn a task-agnostic learning algorithm from a family of training tasks, that is hopeful to be generalizable across tasks and enable new tasks to be learned better and more easily [39]. By taking every class of training samples as a separate learning task, MW-Net can also be seen as a meta-learning strategy, aiming to learn how to properly impose an explicit weighting function from a set of training classes/tasks. The properness of using such meta-learning regime, however, is built on the premise that all training tasks approximately follow a similar task distribution [40], [41], [42], [43]. In complicated data bias scenarios, however, such premise is evidently hampered by the heterogeneous bias situations across different classes, making MW-Net hardly fit a concise weighting rule generally suitable for all classes.

The issue will be more prominent for practical large-scale datasets, especially for those containing large number of training classes but also possessing many rare ones. Then the inter-class heterogeneity will be more significant and the data bias situation more complicated. An easy amelioration is to separately learn a weighting function for each training task/class to obtain a better flexibility. This easy learning manner, however, is not only impractical due to its required large computation burden, but also easily leads to overfitting and thus hardly extracts available weight schemes from highly insufficient training task information for each class. More importantly, such learning manner is deviated from the original motivation of meta-learning, e.g., to learn a general weighting function imposing methodology generalizable and transferable to new biased datasets. The learned weighting scheme is even infeasible to be utilized in new problems with different class numbers and characteristics due to their mismatched input information to the meta-model.

Against the aforementioned issues, in this study, we substantially reform MW-Net to make it performable in practical scenarios with complicated data biases. The core idea is to extract certain feature representations from all training classes/tasks to deliver their specific heterogeneous bias characteristics for discriminating similar training classes (to accumulate them as a meta-class), and take this meta-knowledge as the supplementary input information besides the sample loss into the weighting function. The purpose is to make such reformed MW-Net capable of distinguishing

1. Since the loss values of all samples have been considered and validated to be important and beneficial information for exploring proper sample weight assignment principle, its distribution largely delivers the underlying bias configurations underlying data [7], [8], [37].

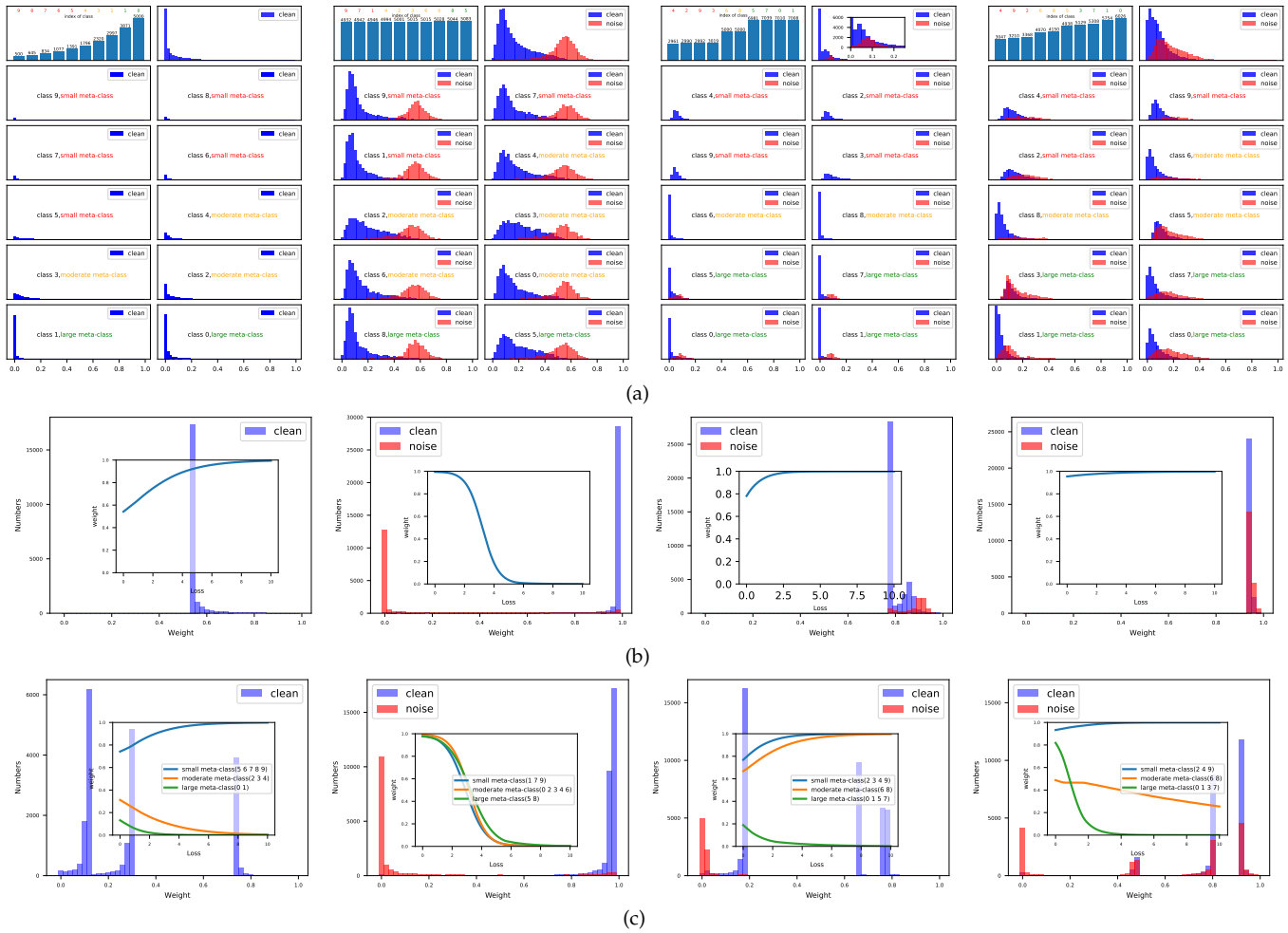


Fig. 2. (a) Empirical pdfs of cross-entropy loss of all 10 classes on CIFAR-10 dataset with different data bias cases. From left to right: Class imbalance (imbalanced factor 10), Symmetric noise (noise rate 40%), Asymmetric noise (noise rate 40%), Feature-dependent noise (Type-I + 30% Asymmetric). For each case, the classes are demonstrated in an ascending order of their contained sample numbers. (b) The weighting function extracted by MW-Net [9], alongside the histogram of all sample weights calculated by it, for each type of biased dataset in (a). (c) Three weighting functions extracted by CMW-Net (corresponding to three meta-classes with small, moderate, large data scales), alongside the histogram of all sample weights calculated by them, for each type of biased dataset in (a).

individual properties of different data classes, and adaptively ameliorating their imposed weighting function forms. We thus call this approach as the Class-aware Meta-Weight-Net, or CMW-Net for brevity.

By employing CMW-Net so proposed, the limitations of the MW-Net are expected to be essentially ameliorated. On the one hand, the involvement of the class feature makes CMW-Net capable of integrating similar training classes (i.e., a meta-class) to soundly train their shared weighting scheme while eliminating the unexpected interference from heterogeneouss classes. The method is thus hopeful to achieve better approximation capability to this specific meta-learning task even for datasets with complicated biases, as can be evidently observed Fig. 1(b) (more details are introduced in Sec. 4). On the other hand, this meta-class-knowledge embedded structure makes the learned weighting function possess consistent format across different training datasets, and thus enables the possibility of transferring the meta-learned weighting scheme to be readily used to new unseen biased datasets. The method is thus also with potentially stronger task-generalization capability. This is especially meaningful for those large-scale problems, since we can readily obtain the weighting-function-imposing methodology from relatively smaller-scale tasks, and then directly use it in larger-scale problems (corresponding to the meta-train and meta-test stages, respectively [43]), avoiding the extra time-consuming weighting function tuning process.

In a nutshell, the main contribution of this paper can be summarized as follows.

1) Through simply taking the scale level of each sample class as input task information to represent its meta-class feature, we realize an easy form for CMW-Net meta-model. Albeit simple, this task information is validated to be effective and capable of assembling sample classes with similar homoscedastic loss distributions, as shown in Fig.2(a), revealing the potential usefulness of such CMW-Net framework and possibility for further improving its performance.

2) The proposed CMW-Net is model-agnostic, and is substantiated to be performable in different complicated data bias cases, like class imbalance, symmetric, asymmetric and feature-dependent label noise ones, as shown in Fig.2(c). It is also verified to obtain competitive results with state-of-the-art (SOTA) methods on real-world biased datasets, like ANIMAL-10N [44], Webvision [14] and WebFG-496 [45].

3) We further make soft-label amelioration for the proposed CMW-Net model by integrating sample pseudo-label

knowledge estimated by model prediction, aiming to correct and reuse the suspected noisy samples into the model training. Attributed to the beneficial information contained among these samples, the performance of CWN-Net tends to be evidently improved, especially for noisy label cases.

4) We study the transferability of CMW-Net. The learned weighting scheme can be used in a plug-and-play manner, and can be directly deployed on unseen datasets, without extra hyperparameters required to be tuned.

5) We also evaluate easy generality of CMW-Net to other robust learning tasks, including partial-label learning [46], semi-supervised learning [47] and selective classification [48].

The paper is organized as follows. Sec. 2 discusses related work. Sec. 3 presents the proposed CMW-Net method as well as its learning algorithm and convergence analysis. Simulated and real-world experiments are demonstrated in Sec. 4 and Sec. 5, respectively. Sec. 6 evaluates the transferability of CMW-Net. Sec. 7 introduces the evaluation of CMW-Net to several related applications. The conclusion is finally made.

2 Related Work

Conventional Sample Weighting Methods. The idea of re-weighting examples can be dated back to importance sampling [25], aiming to assign weights to samples in order to match one distribution to another. Besides, the early attempts of dataset resampling [49], [50] or instance re-weight [51] pre-evaluate the sample weights using certain prior knowledge on the task or data. To make sample weights fit data more flexibly, more recent researchers focus more on pre-designing an explicit weighting function mapping from training loss to sample weight, and dynamically ameliorate weights during training process. There are mainly two manners to design such weighting function. One is to make it monotonically increasing, which is specifically effective in class imbalance case. Typical methods along this line include the boosting algorithm [27], [28], [52], hard example mining [29] and focal loss [30], which impose larger weights to ones with larger loss values. On the contrary, another series of methods specify the weighting function as monotonically decreasing, more popularly used in noisy label cases. Typical examples include SPL [31] and its extensions [33], [53], iterative reweighting [6], [35], paying more emphasis on easy samples with smaller losses. The evident limitation of these methods is that they all need to manually pre-specify the form of weighting function as well as its hyper-parameters based on users' prior expert knowledge on the investigated data and learning problem, raising their difficulty to be readily used in real applications. Meanwhile, presetting a certain form of weighting function suffers from the limited flexibility to make the model adaptable to the complicated training data biases, like those with inter-class bias-heterogenous distributions.

Meta Learning Methods for Sample Weighting. Inspired by meta-learning developments [15], [39], [41], [43], recently some methods have been proposed to adaptively learn sample weights from data to make the learning more automatic and reliable. Typical methods along this line include FWL [54], learning to teach [55], MentorNet [37], L2RW [26], and MW-Net [9]. Especially, MW-Net [9] adopts an MLP net to learn an explicit weighting scheme instead of conventional pre-defined weighting scheme. It has been substantiated that weighting function automatically extracted from data comply with those proposed in the hand-designed studies for class-imbalance or noisy labels [9]. As analyzed in Sec. 1, the effectiveness of the method, however, is built on the premise assumption that all training classes are with approximately homogeneous biases. However, real-world biased dataset are always inter-class heteroscedastic, and thus it tends to lose efficacy in more practical applications.

Other Methods for Class Imbalance. Except for sample re-weighting methods, there exist other learning paradigms for handling class imbalance. Typically, [56], [57] try to transfer the knowledge learned from major classes to minor ones. [58] uses meta feature modulator to balance the contribution per class during the training phase. The metric learning based methods, e.g., triple-header loss [59] and range loss [60], have also been developed to effectively exploit the tailed data to improve the generalization. Furthermore, [61] applies domain adaptation on learning tail class representation.

Other Methods for Corrupted Labels. For handling noisy label issue, many methods have also been designed by making endeavor to correct noisy labels to their true ones to more sufficiently discover and reuse the beneficial knowledge underlying these corrupted data. The typical strategies include supplementing an extra label correction step [7], [62], [63], [64], designing a robust loss function [6], [65], [66], [67], [68], [69], revising the loss function via loss correction [53], [70], [71], [72], and so on. Please refer to references [20], [21], [22], [23], [24] for a more overall review.

3 Class-Aware Meta-Weight-Net

3.1 Sample Re-weighting Methodology

Consider a classification problem with biased training set $\mathcal{D}^{tr} = \{x_i, y_i\}_{i=1}^N$, where x_i denotes the i-th training sample, $y_i \in \{0,1\}^C$ is the one-hot encoding label corresponding to x_i, and N is the number of the entire training data. $f(x;\mathbf{w})$ denotes the classifier with $\mathbf{w}$ representing its model parameters. In current applications, $f(x,\mathbf{w})$ is always set with a DNN architecture. We thus also adopt DNN as our prediction model, and call it a classifier network for convenience in the following. Generally, the optimal model parameter $\mathbf{w}^*$ can be extracted by minimizing the following training loss calculated on the training set:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{N}\sum_{i=1}^{N} \ell(f(x_i;\mathbf{w}), y_i), \tag{1}$$

where $\ell(f(x;\mathbf{w}), y)$ denotes the training loss on training sample (x, y). In this study, we adopt the commonly adopted cross-entropy (CE) loss $\ell(f(x;\mathbf{w}), y) = -y^T \log(f(x;\mathbf{w}))$, where $f(x;\mathbf{w})$ denotes the network output (especially, $f(x;\mathbf{w}) \in \mathbb{R}^c$ when using Sigmoid activation function in the end layer of the network). For notation convenience, we denote $L_i^{tr}(\mathbf{w}) = \ell(f(x_i;\mathbf{w}), y_i)$ in the following.

In the presence of biased training data, sample re-weighting methods aim to enhance the robustness of network training by imposing a weight $v_i \in [0,1]$ on the i-th training sample loss. Then the optimal parameter $\mathbf{w}^*$ is calculated by minimizing the following weighted loss function:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} v_i \ell(f(x_i;\mathbf{w}), y_i). \tag{2}$$

To make sample weights fit data more flexibly, researchers mostly focused on pre-defining a weighting function mapping from training loss to sample weight, and dynamically ameliorate weights during training process [28], [30], [31]. More details can refer to literatures provided in related work.

3.2 Meta-Weight-Net

As aforementioned, most conventional sample re-weighting studies need to manually pre-specify the form of weighting function as well as their hyper-parameters based on cetain expert knowledge for the investigated problem. This naturally raises their difficulty in readily using them in real applications. Meanwhile, such weighting function presetting manner suffers from the limited flexibility to adapt complicated data bias cases, like applications simultaneously containing class imbalance and noisy label abnormalities in their certain classes. To address above issues, MW-Net [9] is proposed to use an MLP to deliver a suitable weighting function from data. The architecture of the MW-Net (see Fig. 1(a)), denoted as $V(\ell;\theta)$, is naturally succeeded from the previous sample re-weighting approaches, by setting its input as training loss and output as sample weight, with θ as its network parameter. Just following standard MLP set, 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 $[0,1]$. This weight net is known with a strong fitting capability to represent a wide range of weighting function forms, like those monotonically increasing or decreasing ones as conventional manually specified ones [36]. The MW-Net thus ideally includes many conventional sample weighting schemes as its special cases.

The parameters contained in MW-Net can be optimized in a meta learning manner [39], [41], [43]. Specifically, with a small amount of unbiased meta-data set $\mathcal{D}^{meta} = \{x_i^{(meta)}, y_i^{(meta)}\}_{i=1}^{M}$ (i.e., with clean labels and balanced data class distribution), representing the meta-knowledge of ground-truth sample-label distribution, where M is the number of meta-samples, the optimal parameter θ^* of MW-Net can be obtained by minimizing the following bi-level optimization problem:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \frac{1}{M}\sum_{i=1}^{M} L_i^{meta}(\mathbf{w}^*(\theta)), \\ s.t.\ \mathbf{w}^*(\theta) &= \arg\min_{\mathbf{w}} \sum_{i=1}^{N} V(L_i^{tr}(\mathbf{w});\theta) L_i^{tr}(\mathbf{w}), \end{aligned} \tag{3}$$

where $L_i^{meta}(\mathbf{w}^*(\theta)) = \ell\Big(f(x_i^{(meta)};\mathbf{w}^*(\theta)), y_i^{(meta)}\Big)$. Experimental results on datasets with inter-class homogeneous bias situations, like all classes with similar imbalance rate for class imbalance or similar noise rate for noisy labels case, have shown that the learned weighting schemes are consistent with empirical pre-defined ones as conventional methods [9].

3.3 Class-aware Meta-Weight-Net

The main limitation of MW-Net lies in its specification of only one unique weighting scheme to handle data biases over all training classes, which largely ignores the inter-class variation of bias situations in practical datasets, especially for those with large number of training classes. This problem can be easily observed in the asymmetric and feature-dependent noise cases as depicted in Fig.2(b). It is evident that when the biased datasets are with inter-class heteroscedastic loss distributions, MW-Net tends to largely lose its efficacy. This motivates us to reform MW-Net to make it possess adaptability of specifying proper weighting schemes to different classes based on their own internal bias characteristics.

To this aim, we propose to integrate meta-class-feature knowledge into the input of the original MW-Net as a beneficial compensation besides the original sample loss input. The purpose is to enable the output weight of a sample correlated with its included training class/task, so as to make it possibly adaptable against class bias variations. We thus call the new weighting model as Class-aware Meta-Weight-Net (CMW-Net for brevity). Such amelioration is expected to accumulate training classes with approxmiately homogeneous bias types to extract a possible faithful sample-weighting scheme shared by them, while suppressing the unexpected interference by other heterogeneous ones.

In this study, we attempt to take the scale level of each training class to represent its meta-class feature. Albeit simple, such feature does be able to deliver helpful class/task pattern underlying its bias types. For instance, as demonstrated in Fig. 2(a) and 2(c) (especially, please see class imbalance, asymmetric noise and feature-dependent noisy label cases), for training classes with relatively small number of samples, the class imbalance bias tends to more possibly occur, and it is thus more rational to set the sample-weighting function as monotonically increasing to emphasize those informative marginal samples with larger loss values. Yet for those with relatively large number of training samples, the weighting function should be more rationally set as monotonically decreasing to take samples with smaller loss values as more important ones, since in this case we have sufficient training samples and should focus more on those high-confident ones with probable clean labels.

Specifically, we design the architecture of CMW-Net as the integration of two branches, as depicted in Fig. 1(b). Denote N_i $(i = 1,\cdots,N)$ as the number of samples contained in the training class to which the i-th sample x_i belongs. Then one branch of CMW-Net can be expressed as $\mathcal{C}(N_i;\Omega) \in \{0,1\}^K$, by taking N_i as its input to represent the meta-class knowledge, and including a hidden layer containing K nodes, attached with K-levels of scales $\Omega = \{\mu_k\}_{k=1}^{K}$ sorted in an ascending order (i.e., $\mu_1 < \mu_2 < \cdots < \mu_K$). The output of this branch is a K-dimensional one-hot vector (i.e., meta-class label), whose 1-element is located at its k-th dimension corresponding to the nearest μ_k to the input N_i.

The other branch can be represented as $\mathbf{V}(L_i^{tr}(\mathbf{w});\Theta) \in \mathbb{R}^K$, built as an MLP architecture with the loss value of the i-th sample as its input, containing one hidden layer and a K-dimensional output[2]. Different from 1-dimensional weight output of MW-Net, this network contains K output weights, corresponding to its K different weighting schemes imposed on samples located in different meta-classes. The sharing hidden layer among these meta-classes extracts the

2. In all our experiments, we just simply set the hidden layer containing 100 nodes with ReLU activation function, and specify the output node with Sigmoid activation function, to guarantee the output of each meta-class located in the interval of $[0,1]$.

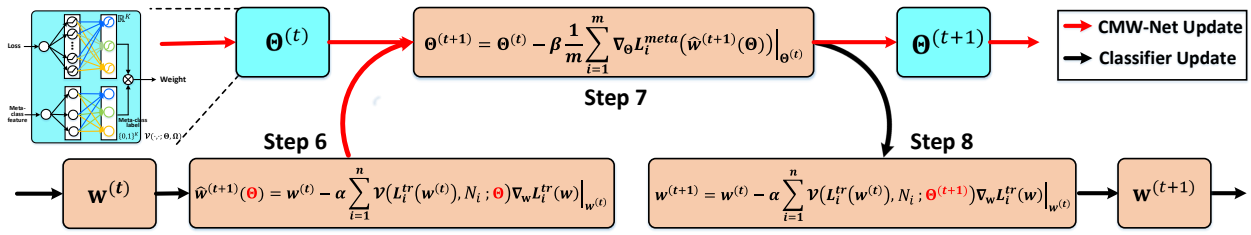


Fig. 3. Main flowchart of the proposed CMW-Net meta-training algorithm (steps 6-8 in Algorithm 1).

correlation among weighting principles of different meta-classes, which helps reduce the risk of overfitting.

Then the CMW-Net weighting function is formulated as:

$$\mathcal{V}(L_i^{tr}(\mathbf{w}), N_i; \Theta, \Omega) = \mathbf{V}(L_i^{tr}(\mathbf{w}); \Theta) \otimes \mathcal{C}(N_i; \Omega), \quad (4)$$

where $\otimes$ denotes the dot product between two vectors. Through the modulation of the meta-class information, CMW-Net is expected to learn a class-aware weighting function by accumulating classes with homogeneous bias situations, and allow different sample classes possessing different weighting schemes complying their own internal bias characteristics.

Now, the objective function of CWM-Net can be written as the following bi-level optimization problem:

$$\{\Theta^*, \Omega^*\} = \arg\min_{\Theta,\Omega} \frac{1}{M}\sum_{i=1}^{M} L_i^{meta}(\mathbf{w}^*(\Theta, \Omega)), \quad (5)$$

$$\mathbf{w}^*(\Theta, \Omega) = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} \mathcal{V}(L_i^{tr}(\mathbf{w}), N_i; \Theta, \Omega) L_i^{tr}(\mathbf{w}). \quad (6)$$

Note that CMW-Net is degenerated to the original MW-Net if we take $K = 1$.

3.4 Learning Algorithm of CMW-Net

3.4.1 Meta-training: learning CMW-Net from training data

There are two groups of hyper-parameters, including Θ and Ω, required to be optimized to attain the CMW-Net model. Therein, the optimization of the scale parameters Ω corresponds to an integer programming problem and thus hard to design an efficient algorithm for getting its global optimum. We thus adopt a two-stage process to first predetermine a rational specification of Ω^*, and then focus the computation on optimizing other parameters in the problem. In specific, the standard K-means algorithm [73] is employed on the sample numbers within all training classes (including C positive integers) to obtain cluster centers $\Omega = \{\mu_k\}_{k=1}^{K}$ sorted in an ascending order. Throughout all our experiments, we simply set $K = 3$ to make our trained CMW-Net able to distinguish small, moderate, and large-scale meta-classes for different datasets. All our experiments show consistently and stably fine performance under such simple setting. This also implies that there remains a large room for further performance enhancement of our model by utilizing more elegant optimization techniques and designing more comprehensive class features, which will be further investigated in our future research.

Then our aim is to solve the bi-level optimization of Eqs. (5) and (6) to obtain optimal Θ^* and $\mathbf{w}^*$. To make notation concise, we directly neglect Ω in Eqs. (5) and (6) in the following. Note that exact solutions to Eqs. (5) and (6) require to solve the optimal $\mathbf{w}^*$ whenever Θ gets updated. This is both analytically infeasible and computationally expensive. Following previous works [9], [26], we adopt one step of stochastic gradient descent (SGD) update for $\mathbf{w}$ to online approximate the optimal classifier for a given Θ, which guarantees the efficiency of the algorithm.

Formulating learning manner of classifier network. To optimize Eq. (6), in each iteration a mini-batch of training samples $\{(x_i, y_i)\}_{i=1}^{n}$ is sampled, where n is the mini-batch size. Then the classifier parameter can be updated by moving the current $\mathbf{w}^{(t)}$ along the descent direction of Eq. (6) on the mini-batch training data as the following expression:

$$\hat{\mathbf{w}}^{(t+1)}(\Theta) = \mathbf{w}^{(t)} - \alpha \sum_{i=1}^{n} \mathcal{V}(L_i^{tr}(\mathbf{w}^{(t)}), N_i; \Theta) \nabla_{\mathbf{w}} L_i^{tr}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}}, \quad (7)$$

where α is the learning rate for the classifier network f.

Updating parameters of CMW-Net: Based on the classifier updating formulation $\hat{\mathbf{w}}^{(t+1)}(\Theta)$ from Eq.(7), the parameter Θ of the CMW-Net can then be readily updated guided by Eq.(5), i.e., moving the current parameter $\Theta^{(t)}$ along the objective gradient of Eq.(5). Similar to the updating step for $\mathbf{w}$, the stochastic gradient descent (SGD) is also adopted. That is, the update is calculated on a sampled mini-batch of meta-data $\{(x_i^{meta}, y_i^{meta})\}_{i=1}^{m}$, expressed as

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{m}\sum_{i=1}^{m} \nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))\Big|_{\Theta^{(t)}}, \quad (8)$$

where β is the learning rate for CMW-Net. Notice that Θ in $\hat{\mathbf{w}}^{t+1}(\Theta)$ here is a variable instead of a quantity, which makes the gradient in Eq. (8) able to be computed.

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.,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \sum_{i=1}^{n} \mathcal{V}(L_i^{tr}(\mathbf{w}^{(t)}), N_i; \Theta^{(t+1)}) \nabla_{\mathbf{w}} L_i^{tr}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}}. \quad (9)$$

Note that we derive with plain SGD here. This, however, also holds for most variants of SGD, like Adam [74]. The CMW-Net learning algorithm can then be summarized in Algorithm 1, and Fig.3 illustrates its main implementation process (steps 6-8). All computations of gradients can be efficiently implemented by automatic differentiation techniques and generalized to any deep learning architectures of the classifier. The algorithm can be easily implemented using popular deep learning frameworks like PyTorch [75]. It is easy to see that both the classifier and CMW-Net gradually ameliorate their parameters during the learning process based on their values calculated in the last step, and the weights thus tends to be updated in a stable manner.

Algorithm 1 The CMW-Net Meta-training Algorithm

Input: Training dataset $\mathcal{D}^{tr}$, meta-data set $\mathcal{D}^{meta}$, batch size n, m, max iterations T.
Output: Classifier parameter $\mathbf{w}^{(*)}$, CMW-Net parameter $\Theta^{(*)}$

1: Apply K-means on the sample numbers of all training classes to obtain $\Omega = \{\mu_k\}_{k=1}^{K}$ sorted in an ascending order.
2: Initialize classifier network parameter $\mathbf{w}^{(0)}$ and CMW-Net parameter $\Theta^{(0)}$.
3: **for** $t = 0$ **to** $T-1$ **do**
4: $\{x, y\} \leftarrow$ SampleMiniBatch($\tilde{\mathcal{D}}^{tr}, n$).
5: $\{x^{meta}, y^{meta}\} \leftarrow$ SampleMiniBatch($\mathcal{D}^{meta}, m$).
6: Formulate the learning manner of classifier network $\hat{\mathbf{w}}^{(t+1)}(\Theta)$ by Eq. (7).
7: Update parameter $\Theta^{(t+1)}$ of CMW-Net by Eq. (8).
8: Update parameter $\mathbf{w}^{(t+1)}$ of classifier by Eq. (9).
9: **end for**

3.4.2 Analysis on intrinsic learning mechanism of CMW-Net

The updating step of Eq. (8) can be equivalently rewritten as (derivations are presented in supplementary material):

$$\Theta^{(t+1)} = \Theta^{(t)} + \alpha\beta \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} G_{ij} \right) \frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}, \tag{10}$$

where $G_{ij} = \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}}$. Neglecting the coefficient $\frac{1}{m}\sum_{i=1}^{m} G_{ij}$, it is easy to see that each term in the above sum orients to the ascend gradient of the weight function $\mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)$. The coefficient imposed on the j-th gradient term, $\frac{1}{m}\sum_{i=1}^{m} G_{ij}$, represents the similarity between the gradient of the j-th training sample computed on the training loss and the average gradient of the mini-batch meta data calculated on meta loss. This means that if the learning gradient of a training sample is similar to that of the meta samples, then it inclines to be considered as in-distribution and CMW-Net tends to produce a higher sample weight for it. Conversely, samples with gradient different from that of the meta set incline to be suppressed. This understanding is consistent with the intrinsic working mechanism underlying the well-known MAML [41], [76].

3.4.3 Meta-test: transferring CMW-Net to unseen tasks

After the meta-training stage, the learned CMW-Net with parameter $\Theta^{(*)}$ can then be transferred to assign proper sample weights on unseen biased datasets. Specifically, for a query dataset $\mathcal{D}^q = \{x_i^q, y_i^q\}_{i=1}^{N^q}$, we first need to implement K-means on sample numbers of all classes to obtain its cluster centers $\Omega^q = \{\mu_k^q\}_{k=1}^{K}$ as meta-class feature. Then the learned CMW-Net can be directly used to imposing sample weights to the classifier learning of the problem by solving:

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \sum_{i=1}^{N^q} \mathcal{V}(L_i^q(\mathbf{u}), N_i^q; \Theta^*, \Omega^q) L_i^q(\mathbf{u}), \tag{11}$$

where $L_i^q(\mathbf{u}) = \ell\left(f(x_i^q; \mathbf{u}), y_i^q\right)$, and N_i^q represents the number of samples contained in the class to which x_i^q belongs. Then we can directly solve Eq.(11) with the learned Θ^* to obtain classifier $\mathbf{u}^*$. The overall algorithm is summarized in Algorithm 2.

Algorithm 2 The CMW-Net Meta-test Algorithm

Input: Training dataset $\mathcal{D}^q$, batch size n', max iterations T' and meta-learned CMW-Net with parameter Θ^*.
Output: Classifier parameter $\mathbf{u}^*$.

1: Apply K-means on sample numbers of all training classes to obtain $\Omega^q = \{\mu_k^q\}_{k=1}^{K}$ sorted in an ascending order.
2: Initialize classifier network parameter $\mathbf{u}^{(0)}$.
3: **for** $t = 0$ **to** $T'-1$ **do**
4: Update classifier $\mathbf{u}^{(t+1)}$ by solving Eq. (11).
5: **end for**

3.5 Convergence of the CMW-Net Learning Algorithm

Next we attempt to establish a convergence result of our method for calculating Eqs. (5) and (6) in a bi-level optimization manner. In particular, we theoretically show that our method converges to critical points of both the meta loss (Eq.(5)) and training loss (Eq.(6)) under some mild conditions in Theorem 1 and 2, respectively. The proofs are presented in the supplementary material (SM for brevity).

Theorem 1. *Suppose the loss function ℓ is Lipschitz smooth with constant L, and CMW-Net $\mathcal{V}(\cdot,\cdot;\Theta)$ 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 $\alpha_t, \beta_t, 1 \le t \le T$ be monotonically decreasing sequences, and satisfy $\alpha_t = \min\{\frac{1}{L}, \frac{c_1}{\sqrt{T}}\}, \beta_t = \min\{\frac{1}{L}, \frac{c_2}{\sqrt{T}}\}$, for some $c_1, c_2 > 0$, such that $\frac{\sqrt{T}}{c_1} \ge L, \frac{\sqrt{T}}{c_2} \ge L$. Meanwhile, they satisfy $\sum_{t=1}^{\infty}\alpha_t = \infty, \sum_{t=1}^{\infty}\alpha_t^2 < \infty, \sum_{t=1}^{\infty}\beta_t = \infty, \sum_{t=1}^{\infty}\beta_t^2 < \infty$. Then CMW-Net can then achieve $\mathbb{E}[\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\|_2^2] \le \epsilon$ in $\mathcal{O}(1/\epsilon^2)$ steps. More specifically,*

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

where C is some constant independent of the convergence process.

Theorem 2. *Under the conditions of Theorem 1, CMW-Net can achieve $\mathbb{E}[\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\|_2^2] \le \epsilon$ in $\mathcal{O}(1/\epsilon^2)$ steps, where $\mathcal{L}^{tr}(\mathbf{w};\Theta) = \sum_{i=1}^{N}\mathcal{V}(L_i^{tr}(\mathbf{w}), N_i;\Theta)L_i^{tr}(\mathbf{w})$. More specifically,*

$$\min_{0\le t\le T} \mathbb{E}\left[\left\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2\right] \le \mathcal{O}(\frac{C}{\sqrt{T}}),$$

where C is some constant independent of the convergence process.

3.6 Enhancing CMW-Net with Soft Label Supervision

In the typical bias case that some training samples are with corrupted labels, the sample weighting strategy tends to largely neglect the function of these samples by imposing small or even zero weights on them. This manner, however, inclines to regrettably waste the beneficial information essentially contained in these samples. Some recent researches have thus been presented to possibly correct the noisy labels and reuse them in training. One popular option is to extract a pseudo soft label z on a sample x through the clue of the classifier's estimation during the training iterations, and then set the training loss as a convex weighting combination of loss terms computed with the suspected noisy label y and the pseudo-label z [7], [8], [77], [78], i.e.,

$$\ell_S(f(x;\mathbf{w}), y) = v\ell(f(x;\mathbf{w}), y) + (1-v)\ell(f(x;\mathbf{w}), z), \tag{12}$$

where $v \in [0,1]$ denotes the sample weight. By setting the loss as the cross-entropy, the loss (12) can be rewritten as:

$$\ell_S(f(x;\mathbf{w}),y) = -\left(vy+(1-v)z\right)^T \log(f(x;\mathbf{w})). \quad (13)$$

It can then be understood as setting a corrected soft label $vy+(1-v)z$ to ameliorate the original label y to make it more reliably reused and avoid roughly suppressing or throwing off the sample from training as conventional.

We then shortly introduce the current research on how to set the sample weight v in the above (12) or (13). The early attempts often adopted a manual manner for setting this hyper-parameter, e.g., the v is empirically set as $v = 0.8$ for all samples in [77]. Evidently, such a fixed and constant weight specification could not sufficiently convey the variant knowledge of training samples with different contents of corruption and reliability. Afterwards, some methods try to dynamically assign individually weights for different samples. Typically, SELFIE [78] iteratively selects clean samples by assigning weights $v = 1$ on them, and neglect doubtful noisy samples by setting their weights as $v = 0$ in (13). [7] ameliorates this hard weighting manner as soft, by fitting a two-component mixture model per epoch to estimate the probability of a sample being clean or noisy, and then use this probability to assign a soft weight for the corresponding sample. Recently, DivideMix [8] improves [7] by adopting a Gaussian mixture model to estimate v on its per-sample loss distribution and using this clue to set a soft weight v.

However, all above methods require to exploit a separate early-learning stage [79] to heuristically pre-determine the sample weights v, while certainly ignore the beneficial feedback from the classifier during the learning process. We thus can naturally introduce our CMW-Net method to automatically explore a weighting scheme by making it trained together with the classifier in a meta-learning manner. Specifically, we just need to easily revise the training objective of CMW-Net in Eq.(6) as (called CMW-Net-SL):

$$\begin{aligned}\mathbf{w}^*(\Theta) = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} [&\mathcal{V}(L_i^{tr}(\mathbf{w}),N_i;\Theta)L_i^{tr}(\mathbf{w}) \\ &+ (1-\mathcal{V}(L_i^{tr}(\mathbf{w}),N_i;\Theta))L_i^{Pse}(\mathbf{w})], \quad (14)\end{aligned}$$

where $L_i^{Pse}(\mathbf{w}) = \ell(f(x_i;\mathbf{w}),z_i)$. Taking a similar process as Sec. 3.4.2, we have

$$\begin{aligned}&\Theta^{(t+1)} = \Theta^{(t)} + \alpha\beta\times \\ &\sum_{j=1}^{n}\left[\frac{1}{m}\sum_{i=1}^{m}(G_{ij}-G'_{ij})\right]\frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}),N_j;\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}, \quad (15)\end{aligned}$$

where $G'_{ij} = \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|^T_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{Pse}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}}$. Compared with CMW-Net, it is seen that CMW-Net-SL produces another term G'_{ij} to control the learning of the meta-learner. Specifically, if $\frac{1}{m}\sum_{i=1}^{m}(G_{ij}-G'_{ij}) > 0$, it means that the similarity between learning gradient of a training sample with original label and the meta samples is larger than that of a training sample with pseudo-label, and then it will be considered as a relatively clean label and CMW-Net tends to produce a higher sample weight to it. Otherwise, it inclines to be considered as a relatively noisy label and CMW-Net will suppress the influence of original labeled sample while produce more confidence on pseudo-labeled one.

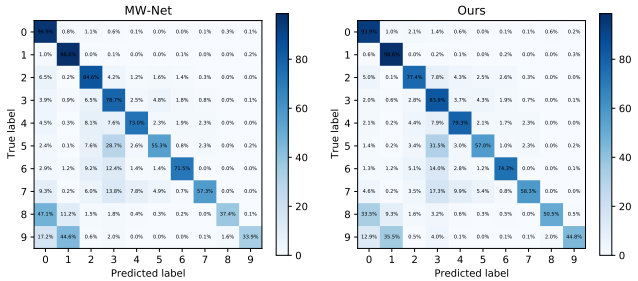


Fig. 4. Confusion matrices obtained by (left) MW-Net and (right) CMW-Net on CIFAR-10-LT (with imbalance factor 200).

In our experiments, we directly apply pseudo-label generating strategy in ELR [79], which has been verified to be effective in tasks like semi-supervised learning [47], [80] and robust learning [7], [79], to produce pseudo-labels in our CMW-Net-SL algorithm. Note that the meta-train and meta-test algorithms of CMW-Net-SL are similar to Algorithms 1 and 2 except that the training loss is revised from (5) to (14). More detailed algorithm description is provided in the SM.

4 Learning with Synthetic Biased Data

4.1 Class Imbalance Experiments

Datasets. We use long-tailed versions of CIFAR-10 and CIFAR-100 datasets (CIFAR-10-LT and CIFAR-100-LT) as in [4]. They contain the same categories as the original CIFAR dataset [82], but are created by reducing the number of training samples per class according to an exponential function $n = n_i\mu^i$, where i denotes 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.

Baselines. The comparison methods include: 1) Empirical risk minimization (ERM): all examples have the same weights. By default, we use standard cross-entropy loss; 2) Focal loss [30] and 3) CB loss [4]: represent SOTA predefined sample re-weighting techniques; 4) LDAM loss [81]: dynamically tune the margins between classes according to their degrees of dominance in the training set; 5) L2RW [26]: adaptively assign sample weights by meta-learning; 6) MW-Net [9]: learn an explicit weighting function by meta-learning; 7) MCW [61]: also use a meta-learning framework, while consider an elegantly designed class-wise weighting scheme, validated to be specifically effective for class imbalance bias. More implementation details are specified in SM.

Results. Table 1 shows the test errors of all competing methods by taking ResNet-32 as the classifier model on CIFAR-10-LT and CIFAR-100-LT with different imbalance factors. It can be observed that: 1) Our algorithm outperforms other competing methods on the datasets, showing its robustness in such biased data; 2) CMW-Net evidently outperforms MW-Net in each experiment. Especially, the performance gain tends to be more evident under larger imbalanced factors. Fig. 4 shows confusion matrices produced by the results of MW-Net and CMW-Net on CIFAR-10-LT with imbalance factor 200[3]. Compared with MW-Net, it is seen that CMW-Net improves the accuracies on tail classes and meanwhile maintains good performance on head classes. 3) Although

3. The confusion matrix is calculated by applying the trained classifier to the corresponding testing set included with the CIFAR-10 dataset.

TABLE 1
Test top-1 error (%) comparison of different competing methods with ResNet-32 classifier on CIFAR-10-LT and CIFAR-100-LT under different imbalance settings. * indicates results reported in [61].

Dataset Name	CIFAR-10-LT						CIFAR-100-LT					
Imbalance factor	200	100	50	20	10	1	200	100	50	20	10	1
ERM	34.32	29.64	25.19	17.77	13.61	7.53	65.16	61.68	56.15	48.86	44.29	29.50
Focal loss [30]	34.71	29.62	23.29	17.24	13.34	6.97	64.38	61.59	55.68	48.05	44.22	28.85
CB loss [4]	31.11	27.63	21.95	15.64	13.23	7.53	64.44	61.23	55.21	48.06	42.43	29.37
LDAM loss [81]*	-	26.65	-	-	13.04	-	60.40	-	-	-	43.09	-
L2RW [26]	33.49	25.84	21.07	16.90	14.81	10.75	66.62	59.77	55.56	48.36	46.27	35.89
MW-Net [9]	32.80	26.43	20.90	15.55	12.45	7.19	63.38	58.39	54.34	46.96	41.09	29.90
MCW [61] with CE loss*	29.34	23.59	19.49	13.54	11.15	**7.21**	**60.69**	56.65	51.47	44.38	40.42	-
CMW-Net with CE loss	**27.80**	**21.15**	**17.26**	**12.45**	**10.97**	8.30	60.85	**55.25**	**49.73**	**43.06**	**39.41**	30.81
MCW [61] with LDAM loss*	**25.10**	20.00	17.77	15.63	12.60	10.29	60.47	55.92	**50.84**	47.62	42.00	-
CMW-Net with LDAM loss	25.57	**19.95**	**17.66**	**13.08**	**11.42**	**7.04**	**59.81**	**55.87**	51.14	**45.26**	**40.32**	**29.19**

TABLE 2
Performance comparison of different competing methods in test accuracy (%) on CIFAR-10 and CIFAR-100 with symmetric and asymmetric noise. The average accuracy and standard deviation over 3 trials are reported.

Datasets	Noise	Symmetric Noise				Asymmetric Noise			
		0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
CIFAR-10	ERM	86.98 ± 0.12	77.52 ± 0.41	73.63 ± 0.85	53.82 ± 1.04	83.60 ± 0.24	77.85 ± 0.98	69.69 ± 0.72	55.20 ± 0.28
	Forward [70]	87.99 ± 0.36	83.25 ± 0.38	74.96 ± 0.65	54.64 ± 0.44	91.34 ± 0.28	89.87 ± 0.61	87.24 ± 0.96	81.07 ± 1.92
	GCE [6]	89.99 ± 0.16	87.31 ± 0.53	82.15 ± 0.47	57.36 ± 2.08	89.75 ± 1.53	87.75 ± 0.36	67.21 ± 3.64	57.46 ± 0.31
	M-correction [7]	93.80 ± 0.23	92.53 ± 0.11	90.30 ± 0.34	86.80 ± 0.11	92.15 ± 0.18	91.76 ± 0.57	87.59 ± 0.33	67.78 ± 1.22
	DivideMix [8]	95.70 ± 0.31	95.00 ± 0.17	94.23 ± 0.23	**92.90 ± 0.31**	93.96 ± 0.21	91.80 ± 0.78	80.14 ± 0.45	59.23 ± 0.38
	L2RW [26]	89.45 ± 0.62	87.18 ± 0.84	81.57 ± 0.66	58.59 ± 1.84	90.46 ± 0.56	89.76 ± 0.53	88.22 ± 0.71	85.17 ± 0.31
	MW-Net [9]	90.46 ± 0.52	86.53 ± 0.57	82.98 ± 0.34	64.41 ± 0.92	92.69 ± 0.24	90.17 ± 0.11	68.55 ± 0.76	58.29 ± 1.33
	CMW-Net	91.09 ± 0.54	86.91 ± 0.37	83.33 ± 0.55	64.80 ± 0.72	93.02 ± 0.25	92.70 ± 0.32	91.28 ± 0.40	87.50 ± 0.26
	CMW-Net-SL	**96.20 ± 0.33**	**95.29 ± 0.14**	**94.51 ± 0.32**	92.10 ± 0.76	**95.48 ± 0.29**	**94.51 ± 0.52**	**94.18 ± 0.21**	**93.07 ± 0.24**
CIFAR-100	ERM	60.38 ± 0.75	46.92 ± 0.51	31.82 ± 1.16	8.29 ± 3.24	61.05 ± 0.11	50.30 ± 1.11	37.34 ± 1.80	12.46 ± 0.43
	Forward [70]	63.71 ± 0.49	49.34 ± 0.60	37.90 ± 0.76	9.57 ± 1.01	64.97 ± 0.47	52.37 ± 0.71	44.58 ± 0.60	15.84 ± 0.62
	GCE [6]	68.02 ± 1.05	64.18 ± 0.30	54.46 ± 0.31	15.61 ± 0.97	66.15 ± 0.44	56.85 ± 0.72	40.58 ± 0.47	15.82 ± 0.63
	M-correction [7]	73.90 ± 0.14	70.10 ± 0.14	59.50 ± 0.35	48.20 ± 0.23	71.85 ± 0.19	70.83 ± 0.48	60.51 ± 0.52	16.06 ± 0.33
	DivideMix [8]	76.90 ± 0.21	75.20 ± 0.12	72.00 ± 0.33	**59.60 ± 0.21**	76.12 ± 0.44	73.47 ± 0.63	45.83 ± 0.83	16.98 ± 0.40
	L2RW [26]	65.32 ± 0.42	55.75 ± 0.81	41.16 ± 0.85	16.80 ± 0.22	65.93 ± 0.17	62.48 ± 0.56	51.66 ± 0.49	12.40 ± 0.61
	MW-Net [9]	69.93 ± 0.40	65.29 ± 0.43	55.59 ± 1.07	27.63 ± 0.56	69.80 ± 0.34	64.88 ± 0.63	56.89 ± 0.95	17.05 ± 0.52
	CMW-Net	70.11 ± 0.19	65.84 ± 0.50	56.93 ± 0.38	28.36 ± 0.67	71.07 ± 0.56	66.15 ± 0.51	58.21 ± 0.78	17.41 ± 0.16
	CMW-Net-SL	**77.84 ± 0.12**	**76.25 ± 0.67**	**72.61 ± 0.92**	55.21 ± 0.31	**77.73 ± 0.37**	**75.69 ± 0.68**	**61.54 ± 0.72**	**18.34 ± 0.21**

LDAM loss already has the capacity of mitigating the long-tailed issue by penalizing hard examples, our method can further boost its performances. 4) Owing to its class-wise weighting scheme, MCW also attains good performance in these experiments. Yet CMW-Net still performs better in most cases. Considering its adaptive weighting-scheme-setting capability and general usability in wider range of biased issues, it should be rational to say that CMW-Net is effective.

To understand the weighing scheme learned by CMW-Net, we also depict the weighting functions learned by the CMW-Net in Fig.2(c) (first column). It is seen that compared with MW-Net shown in Fig.2(b) (first column), CMW-Net produces three weighting functions corresponding to small, moderate and large-scale meta-classes. The overall tendency complies with conventional empirical setting for such class-wise weight functions, like MCW [4], [61], i.e., assigning weights inversely related to the class sizes. Specifically, the learned weights of the tail classes are more prominent than those of the head ones, implying that samples in tail classes should be more emphasized in training to alleviate the class imbalanced bias issue. This also explains the consistently better performance of CMW-Net as compared with MW-Net.

4.2 Feature-independent Label Noise Experiment

Datasets. We study two types of label noise following previous works [70], [71]: 1) Symmetric noise: randomly replace sample labels for a percentage of the training data with all possible labels. 2) Asymmetric noise: try to mimic the structure of real-life label noise, where labels are only replaced by similar classes. Two benchmark datasets are employed: CIFAR-10 and CIFAR-100 [82].

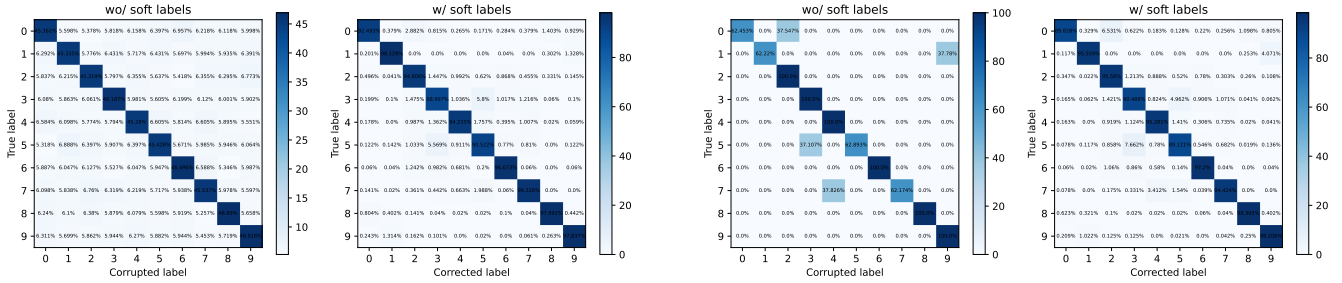
Baselines. The comparison methods include: 1) ERM; 2) Forward [70]: correct the prediction by the label transition matrix; 3) GCE [6]: behave as a robust loss to handle the noisy labels; 4) M-correction [7]; 5) DivideMix [8]: represent the SOTA method for handling noisy label bias, by dividing training data into clean and noisy ones through a loss threshold and designing different label amelioration strategies on them through two diverged networks to co-train the classifier; 6) L2RW [26] and 7) MW-Net [9]: represent the sample re-weighting methods by meta-learning. More experimental details are listed in SM.

Establishing Meta dataset. Motivated by [7] and [8], we selected meta data at each epoch according to the training loss. Specifically, we explore to create the meta dataset based on the high-quality clean samples as well as its high-quality pseudo labels from the training set (with lowest losses) as an unbiased estimator of the clean data-label distribution. To make the meta dataset balanced, we selected 10 images per class in each epoch iteration. In this case, the performance of meta dataset can be served as an indicator to measure how much extent CMW-Net is trained to filter noisy samples and generalized to clean test distribution.

Such meta dataset may lack of diversity pattern to characterize the latent clean data-label distribution. To alleviate this issue, we further explore to utilize mixup technique [84] to enrich the variety of the meta data distribution while possibly maintain its unbiasedness. The hyperparameter of convex combination in the technique is randomly sampled from a Beta distribution $Beta(1, 1)$. Our extensive experiments have verified the effectiveness of using such generated meta

TABLE 3
Test accuracy (%) of all competing methods on CIFAR-10 and CIFAR-100 under different feature-dependent noise types and levels. The average accuracy and standard deviation over 3 trials are reported.

Datasets	Noise	ERM	LRT [63]	GCE [6]	MW-Net [9]	PLC [83]	CMW-Net	CMW-Net-SL
CIFAR-10	Type-I (35%)	78.11 ± 0.74	80.98 ± 0.80	80.65 ± 0.39	82.20 ± 0.40	82.80 ± 0.27	82.27 ± 0.33	**84.23 ± 0.17**
	Type-I (70%)	41.98 ± 1.96	41.52 ± 4.53	36.52 ± 1.62	38.85 ± 0.67	42.74 ± 2.14	42.23 ± 0.69	**44.19 ± 0.69**
	Type-II (35%)	76.65 ± 0.57	80.74 ± 0.25	77.60 ± 0.88	81.28 ± 0.56	81.54 ± 0.47	81.69 ± 0.57	**83.12 ± 0.40**
	Type-II (70%)	45.57 ± 1.12	81.08 ± 0.35	40.30 ± 1.46	42.15 ± 1.07	46.04 ± 2.20	46.30 ± 0.77	**48.26 ± 0.88**
	Type-III (35%)	76.89 ± 0.79	76.89 ± 0.79	79.18 ± 0.61	81.57 ± 0.73	81.50 ± 0.50	81.52 ± 0.38	**83.10 ± 0.34**
	Type-III (70%)	43.32 ± 1.00	44.47 ± 1.23	37.10 ± 0.59	42.43 ± 1.27	45.05 ± 1.13	43.76 ± 0.96	**45.15 ± 0.91**
CIFAR-100	Type-I (35%)	57.68 ± 0.29	56.74 ± 0.34	58.37 ± 0.18	62.10 ± 0.50	60.01 ± 0.43	62.43 ± 0.38	**64.01 ± 0.11**
	Type-I (70%)	39.32 ± 0.43	45.29 ± 0.43	40.01 ± 0.71	44.71 ± 0.49	45.92 ± 0.61	46.68 ± 0.64	**47.62 ± 0.44**
	Type-II (35%)	57.83 ± 0.25	57.25 ± 0.68	58.11 ± 1.05	63.78 ± 0.24	63.68 ± 0.29	64.08 ± 0.26	**64.13 ± 0.19**
	Type-II (70%)	39.30 ± 0.32	43.71 ± 0.51	37.75 ± 0.46	44.61 ± 0.41	45.03 ± 0.50	50.01 ± 0.51	**51.99 ± 0.35**
	Type-III (35%)	56.07 ± 0.79	56.57 ± 0.30	57.51 ± 1.16	62.53 ± 0.33	63.68 ± 0.29	63.21 ± 0.23	**64.47 ± 0.15**
	Type-III (70%)	40.01 ± 0.18	44.41 ± 0.19	40.53 ± 0.60	45.17 ± 0.77	44.45 ± 0.62	47.38 ± 0.65	**48.78 ± 0.62**



(a) Symmetry Noise (60%) (b) Asymmetry Noise (60%)

Fig. 5. Confusion matrices obtained by CMW-Net and CMW-Net-SL on CIFAR-10 with (a) Symmetry Noise 60% and (b) Asymmetry Noise 60%.

dataset from training data. Such property makes such meta-learning strategy applicable to real-world biased dataset, since it is always not easy to collect an additional clean meta dataset in practice. We also use such meta-data-generation strategy in the following noisy labels experiments as well as the real-world biased dataset, where an expected clean meta-dataset is always unavailable.

Results. Table 2 evaluates the performance of our method on CIFAR-10 and CIFAR-100 with different levels of symmetric and asymmetric label noise. We report the averaged test accuracy over the last 10 epochs. It is seen that CMW-Net evidently outperforms MW-Net in all cases. For symmetric noise, the training loss distribution is usually homoscedastic, as depicted in Fig.2(b) (second column), and thus CMW-Net learns three similar weighting schemes as that extracted by MW-Net, as shown in Fig.2(c). For asymmetric noise, while MW-Net hardly adapts to the heterogeneous loss distribution across different classes, CMW-Net finely produces weighting schemes conditioned on different meta-classes. Specifically, as shown in Fig.2(c), weighting functions of small and moderate-scale meta-classes tend to more emphasize those informative marginal samples since they don't contain replaced labels from other classes. While for large-scale meta-class, with many corrupted labeled data, CMW-Net tends to impose smaller weights on samples with relatively large losses to suppress the effect of these noisy labels. This shows that the learned weighting scheme by CMW-Net are adaptable to the internal bias patterns of different classes, and thus naturally leads to its superiority over MW-Net.

By introducing soft label amelioration, CMW-Net-SL can further enhance the performance of CMW-Net, as clearly shown in Table 2. This is natural since CMW-Net-SL is able to adaptively refurbish noisy labeled samples rather than roughly trash them from training, and thus a more sufficient exploration on beneficial knowledge from training data could be obtained. Fig. 5 shows the confusion matrices obtained by CMW-Net and CMW-Net-SL in 60% label noise rate case. It is seen that CMW-Net-SL evidently improves the prediction accuracy, especially for classes with heavy corrupted labels.

Note that by involving label amelioration through using pseudo-prediction information as our method, the DivideMix method also performs well in most symmetric noise experiments, especially, slightly better than CMW-Net-SL in 80% noise rate. However, the superiority of our method is still significant in all asymmetric label noise cases. This can be rationally explained by that DivideMix uses a consistent loss threshold for distinguishing clean and noisy samples, which, however, is certainly deviated from the insight of inter-class heteroscedastic loss distributions underlying this type of data bias. As can be observed in Fig.2(c), since clean training classes are simultaneously tail classes, the loss values of some training samples in these classes are possibly larger than those of head classes, especially for their contained noisy samples. Thus DivideMix tends to mistakenly recognize certain amount of clean/noisy samples, which then results in its performance degradation. Comparatively, the class-aware capability possessed by CMW-Net-SL enables the method more properly treat heteroscedastic loss distributions across different classes, and thus obtain more accurate weighting functions specifically suitable for them, which then naturally leads to its relatively superior performance.

4.3 Feature-dependent Label Noise Experiment

We then evaluate the capability of our method against the feature-dependent label noise, which is more approximate to the real-world bias scenarios [38], [83].

Datasets. We follow the PMD noise generation scheme proposed in [83]. Let $\eta_{y_1}(x) = P(y = y_1|x)$ be the true posterior label distribution for the sample x. The noise label is generated by replacing the most confident label $u_x = \arg\max_y \eta_y(x)$ of each training sample x to its second confident category s_x with conditional probability

TABLE 4
Test accuracy (%) of all competing methods on CIFAR-10 and CIFAR-100 under different feature dependent (35%) and independent (30%) noise types and levels. The average accuracy and standard deviation over 3 trials are reported.

Datasets	Noise	ERM	LRT [63]	GCE [6]	MW-Net [9]	PLC [83]	CMW-Net	CMW-Net-SL
CIFAR-10	Type-I + Symmetric	75.26 ± 0.32	75.97 ± 0.27	78.08 ± 0.66	76.39 ± 0.42	79.04 ± 0.50	78.42 ± 0.47	**82.00 ± 0.36**
	Type-I + Asymmetric	75.21 ± 0.64	76.96 ± 0.45	76.91 ± 0.56	76.54 ± 0.56	78.31 ± 0.41	77.14 ± 0.38	**80.69 ± 0.47**
	Type-II + Symmetric	74.92 ± 0.63	75.94 ± 0.58	75.69 ± 0.21	76.57 ± 0.81	80.08 ± 0.37	76.77 ± 0.63	**80.96 ± 0.23**
	Type-II + Asymmetric	74.28 ± 0.39	77.03 ± 0.62	75.30 ± 0.81	75.35 ± 0.40	77.63 ± 0.30	77.08 ± 0.52	**80.94 ± 0.14**
	Type-III + Symmetric	74.00 ± 0.38	75.66 ± 0.57	77.00 ± 0.12	76.28 ± 0.82	80.06 ± 0.47	77.16 ± 0.30	**81.58 ± 0.55**
	Type-III + Asymmetric	75.31 ± 0.34	77.19 ± 0.74	75.70 ± 0.91	75.82 ± 0.77	77.54 ± 0.70	76.49 ± 0.88	**80.48 ± 0.48**
CIFAR-100	Type-I + Symmetric	48.86 ± 0.56	45.66 ± 1.60	52.90 ± 0.53	57.70 ± 0.32	60.09 ± 0.15	59.17 ± 0.42	**60.87 ± 0.56**
	Type-I + Asymmetric	45.85 ± 0.93	52.04 ± 0.15	52.69 ± 1.14	56.61 ± 0.71	56.40 ± 0.34	57.42 ± 0.81	**61.35 ± 0.52**
	Type-II + Symmetric	49.32 ± 0.36	43.86 ± 1.31	53.61 ± 0.46	54.08 ± 0.18	60.01 ± 0.63	59.16 ± 0.18	**61.00 ± 0.41**
	Type-II + Asymmetric	46.50 ± 0.95	52.11 ± 0.46	51.98 ± 0.37	58.53 ± 0.45	61.43 ± 0.33	58.99 ± 0.91	**61.35 ± 0.57**
	Type-III + Symmetric	48.94 ± 0.61	42.79 ± 1.78	52.07 ± 0.35	55.29 ± 0.57	60.14 ± 0.97	58.48 ± 0.79	**60.21 ± 0.48**
	Type-III + Asymmetric	45.70 ± 0.12	50.31 ± 0.39	50.87 ± 1.12	58.43 ± 0.60	54.56 ± 1.11	58.83 ± 0.57	**60.52 ± 0.53**

TABLE 5
Comparison of different competing methods on Animal-10N dataset. Results for baseline methods are copied from [83]

Method	Test Accuracy	Method	Test Accuracy
ERM	79.4 ± 0.14	ActiveBias [85]	80.5 ± 0.26
Co-teaching [86]	80.2 ± 0.13	SELFIE [78]	81.8 ± 0.09
PLC [83]	83.4 ± 0.43	MW-Net [9]	80.7 ± 0.52
CMW-Net	80.9 ± 0.48	CMW-Net-SL	**84.7 ± 0.28**

TABLE 6
Comparison of different competing methods on mini WebVision dataset. Results for baseline methods are copied from [8]. * denotes results trained with Inception-ResNet-v2.

Methods	WebVision		ILSVRC12	
	top1	top5	top1	top5
Forward* [70]	61.12	82.68	57.36	82.36
MentorNet* [37]	63.00	81.40	57.80	79.92
Co-teaching* [86]	63.58	85.20	61.48	84.70
Interative-CV* [87]	65.24	85.34	61.60	84.98
MW-Net [9]	69.34	87.44	65.80	87.52
CMW-Net	70.56	88.76	66.44	87.68
DivideMix* [8]	77.32	91.64	75.20	90.84
ELR* [79]	77.78	91.68	70.29	89.76
DivideMix [8]	76.32	90.65	74.42	91.21
CMW-Net-SL	78.08	92.96	75.72	92.52
DivideMix with C2D [88]	79.42	92.32	**78.57**	93.04
CMW-Net-SL+C2D	**80.44**	**93.36**	77.36	**93.48**

$\tau_{u_x,s_x} = P(\tilde{y} = u_x | y = s_x, x)$. We use three types of τ_{u_x,s_x} designed in [83] as follows:

$$\begin{aligned}
\text{Type-I} : \tau_{u_x,s_x} &= -\frac{1}{2}\left[\eta_{u_x}(x) - \eta_{s_x}(x)\right]^2 + \frac{1}{2}, \\
\text{Type-II} : \tau_{u_x,s_x} &= 1 - \left[\eta_{u_x}(x) - \eta_{s_x}(x)\right]^3, \\
\text{Type-III} : \tau_{u_x,s_x} &= 1 - \frac{1}{3}\Big[\left[\eta_{u_x}(x) - \eta_{s_x}(x)\right]^3 + \\
&\qquad \left[\eta_{u_x}(x) - \eta_{s_x}(x)\right]^2 + \left[\eta_{u_x}(x) - \eta_{s_x}(x)\right]\Big].
\end{aligned}$$

Besides, we also consider the hybrid noise consisting of both feature-dependent noise and symmetric as well as asymmetric noise as in Sec. 4.2. We use CIFAR-10 and CIFAR-100 benchmarks with such simulated label noise.

Baselines. Following the benchmark in [83], we compare the following baselines: 1) ERM; 2) LRT [63]; 3) GCE [6]; 4) MW-Net [9] and 5) PLC [83], which represents the SOTA method specifically designed for addressing heterogeneous feature-dependent label noise. All these methods are generic and handle label noise without assuming the noise structures.

Results. Table 3 lists the performance of different competing methods under three types of feature-dependent noise at noise levels 35% and 70%. It is seen that our method achieves the best performance on all cases. Table 4 further shows the results on datasets corrupted with a combination of feature dependent and independent noises, where feature-independent noise is overlayed on the feature-dependent one and thus bias patterns are more complicated. The superiority of the proposed method can still be easily observed.

The generation mechanism of such feature-dependent noise results in noisy samples near the decision boundary [83], which are harder to distinguish and more likely to be mislabeled. As shown in Fig.2(a) (fourth column), the feature-dependent noisy samples tend to further deteriorate the loss distribution compared with only asymmetric noise independent to data. From Fig.2(c) (fourth column), it can be seen that the proposed method can still finely distinguish most of clean and noisy samples (some noisy samples are wrongly assigned to high weights due to they are samples near the decision boundary). As compared, from Fig.2(b) (fourth column), it can be observed that MW-Net totally fails to distinguish clean and noisy samples, and assigns high weights to all samples, naturally leading to its performance degeneration. Note that the PLC method [83], which is specifically designed for feature-dependent label noise data, also achieves fine results. The main idea of this method is to progressively correct noisy labels and refine the model for those relatively reliable samples with high confidence, measured by a dynamically specified threshold gradually deceased in iterations. Considering its general availability to wider range of data bias cases and relatively more concise meta-learning framework, it should be rational to say that the proposed method is effective.

5 Learning with Real Biased Data

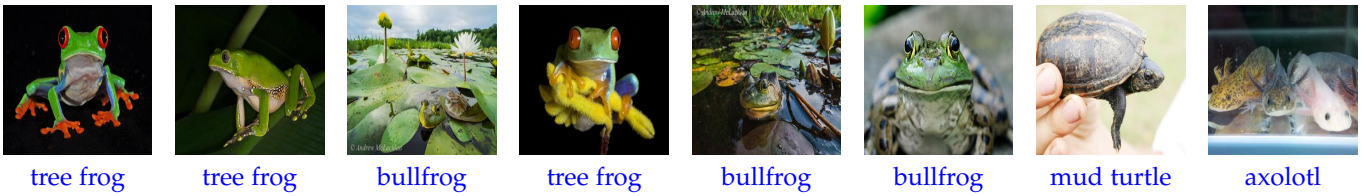
5.1 Learning with Real-world Noisy Datasets

Datasets. We adopt two real-world datasets, ANIMAL-10N [44] and WebVision [78]. Animal-10N contains 55,000 human labeled online images for 10 confusing animal classes, all with approximately similar noisy label distributions (8% noisy samples). Following previous works [78], 50,000 images are exploited for training while the left for testing. For ease of comparison to previous works [37], [87], we consider the mini WebVision dataset which contains the top 50 classes from the Google image subset of WebVision. The performance evaluation is implemented on both the validation sets of mini WebVision [14] and the corresponding class samples of ImageNet [89]. ResNet-50 is adopted as the classifier network. More implementation details are specified in SM.

Results. Tables 5 and 6 compare the test performance of all competing methods trained on the Animal-10N and

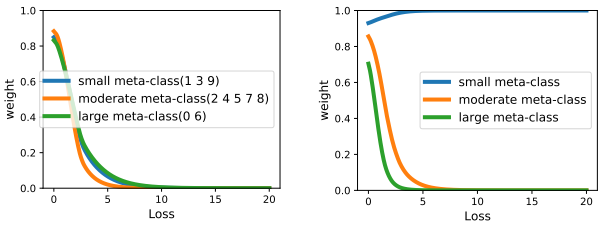
(a) Typical noisy labeled samples corrected by our method from Animal-10N [78]. The original training label is cat.

(b) Typical noisy labeled samples corrected by our method from Animal-10N [78]. The original training label is lynx.



(c) Typical noisy labeled samples corrected by our method from mini-WebVision [14]. The original training label is tailed frog.

Fig. 6. Examples of randomly selected samples with noisy labels corrected by our method. The original training labels and generated pseudo-labels by model are shown in red and blue, respectively. More comprehensive examples are depicted in the SM.



(a) ANIMAL-10N (b) mini WebVision

Fig. 7. Weighting schemes learned by CMW-Net on real biased datasets.

TABLE 7
performance comparison of classification accuracy (%) on WebFG-496.

Methods	Web-Bird	Web-Aircraft	Web-Car	Average
ERM	66.56	64.33	67.42	66.10
Decoupling [90]	70.56	75.97	75.00	73.84
Co-teaching [86]	73.85	72.76	73.10	73.24
Peer-learning [45]	76.48	74.38	78.52	76.46
MW-Net	75.60	72.93	77.33	75.29
CMW-Net	75.72	73.72	77.42	75.62
CMW-Net-SL	**77.41**	**76.48**	**79.70**	**77.86**

mini WebVision datasets, respectively. For the Animal-10N dataset, we compare 4 methods that have reported performance on this dataset. Compared with sample selection methods ActiveBias [85] and Co-teaching [86], our CMW-Net attains better performance, showing its better screening capability for useful samples. Under soft-label amelioration, our method achieves a further performance gain over recent label correction methods SELFIE [78] and PLC [83]. Figs. 6(a) and 6(b) visualize typical noisy examples selected by CMW-Net as well as its generated pseudo-labels. Though they are a pair of easily confused categories (cat, lynx), our method can still extract their wrong labels and correct them as the true ones. Fig. 7(a) further shows the weighting functions learned by CMW-Net, complying with the class balance and inter-class noise homogeneity property of this dataset.

Table 6 also shows the superiority of CMW-Net compared to other competing methods without involving soft labels. By introducing soft labels, our CMW-Net-SL achieves superior performance to recent SOTA methods, DivideMix and ELR. Furthermore, by combining the self-supervised pretraining technique proposed in the C2D method [88], we further boost the performance. In Fig. 6, we show some typical noisy examples corrected by the proposed method, showing its capability of recovering of these easily confused samples.

Fig. 7(b) plots the learned weighting functions by CMW-Net, revealing certain helpful data bias insight. For small-scale meta-class, the corresponding weighting function is with larger weights and shows increasing tendency. It is beneficial to more emphasize their contained rare samples especially those marginal informative ones for alleviating their possibly encountered class-imbalance bias issue. Yet for moderate and larger-scale meta-classes containing relatively abundant training data, the weighting functions are with monotonically decreasing shapes to suppress the negative effect brought by their contained noisy samples. Such more comprehensive and faithful exploration and encoding for data bias situations naturally leads to the better performance of CMW-Net than conventional sample weighting strategies.

5.2 Webly Supervised Fine-Grained Recognition

We further run our method on a benchmark WebFG-496 dataset proposed in [45], consisting of three sub-datasets: Web-aircraft, Web-bird, Web-car, which contain 13,503 images with 100 types of airplanes, 18,388 images with 200 species of birds, and 21,448 images with 196 categories of cars, respectively. The aim is to use web images to train a fine-grained recognition model. The data bias of this dataset is validated to be complicated, with both label noise and class imbalance patterns, as well as certain inter-class variance [45]. Experimental results are shown in Table 7. It is seen that CMW-Net-SL evidently improves other reported SOTA performance [45]. This further validates the effectiveness of our method for such real dataset with complex data biases. More implementation details are given in SM.

TABLE 8
Long-tail recognition accuracy of different competing methods by using ResNet-10 as the classifier on ImageNet-LT [5] . Results for baselines are copied from [5].

Methods	Accuracy			
	Many	Medium	Few	Overall
ERM	40.9	10.7	0.4	20.9
Lifted Loss [91]	35.8	30.4	17.9	30.8
Focal loss [30]	36.4	29.9	16	30.5
Range Loss [60]	35.8	30.3	17.6	30.7
OLTR [5]	43.2	35.1	18.5	35.6
OLTR [5] + CMW-Net	**47.2**	**39.2**	**19.7**	**39.5**

6 Transferability of CMW-Net

As aforementioned, a potential usefulness of the meta-learned weighing scheme by CMW-Net is that it is model-agnostic and hopefully equipped into other learning algorithms in a plug-and-play manner. To validate such transferable capability of CMW-Net, we attempt to transfer meta-learned CMW-Net on relatively smaller dataset to significantly larger-scale ones. In specific, we use CMW-Net trained on CIFAR-10 with feature-dependent label noise (i.e.,35% Type-I + 30% Asymmetric) as introduced in Sec. 4.3 since it finely simulates the real-world noise configuration. The extracted weighting function is depicted in Fig.2(c) (fourth column). We deploy it on two large-scale real-world biased dataset, ImageNet-LT [5] and full WebVision [14].

Table 8 shows the performance on ImageNet-LT. By readily equipping our learned CMW-Net upon the SOTA OLTR algorithm [5] on this dataset, it can be seen that around 4% higher overall accuracy can be readily obtained. Besides, performance on full WebVision is compared in Table 9. It is interesting to see that by directly integrating the learned CMW-Net into the simple ERM algorithm, the performance can be largely improved, even slightly superior to the SOTA HAR method on this data. This implies the potential usefulness of CMW-Net to more practical large-scale problems with complicated data bias situations, by intrinsically reducing the labor and computation costs of specifying proper weighting scheme for a learning algorithm. More experimental details are presented in SM.

7 Extensional Applications

We then evaluate the generality of our proposed adaptive sample weighting strategy in more robust learning tasks, including partial-label learning and semi-supervised learning. The experiments on the selective classification task are introduced in SM due to page limitation.

7.1 Partial-Label Learning

7.1.1 Problem Formulation

Partial-label learning (PLL) [46] aims to deal with the problem where each instance is provided with a set of candidate labels, only one of which is the correct label. Denote $\mathcal{X} \subset \mathbb{R}^d$ as the input space, $\mathcal{Y} := \{1, \cdots, C\}$ as the label space, where C is the number of all training classes. Denote the partially labeled dataset as $\mathcal{D}_{PLL} = \{(x_i, Y_i)\}_{i=1}^N$, where $Y_i \in \mathcal{Y}$ is the candidate label set of x_i. The goal of PLL is to find latent ground-truth label y for each of x_is through observing their partial label sets. The basic definition of PLL is that true label y of an instance x must be in its candidate label set Y.

TABLE 9
Validation accuracy of different competing methods by using InceptionResNet-v2 as the classifier on full WebVision validation set. Results for baselines are copied from [92].

Methods	top1	top5	Methods	top1	top5
ERM	69.7	87.0	MentorMix [93]	74.3	90.5
MentorNet [37]	70.8	88.0	HAR [92]	75.0	90.6
Curriculumnet [16]	72.1	89.2	ERM + CMW-Net	**75.4**	**91.5**

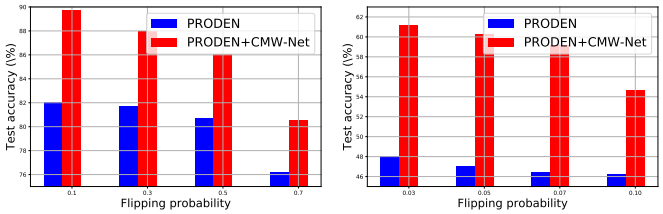


Fig. 8. Accuracy comparisons on PRODEN w/o CMW-Net strategy over (left) CIFAR-10 and (right) CIFAR100 under partial label learning setting.

The PLL risk estimator is then defined over $P(x, Y)$ as:

$$\mathcal{R}_{PLL}(f) = \mathbb{E}_{P(x,Y)}[\ell_{PLL}(f(x), Y)], \tag{16}$$

where $\ell_{PLL}(\cdot, \cdot)$ is the loss function and $f(\cdot)$ is the classifier.

To estimate Eq.(16), it usually treats all the candidate labels equally [46], i.e., $\ell_{PLL}(f(x), Y) = \frac{1}{|Y|}\sum_{y \in Y} \ell(f(x), y)$. Considering that only the true label contributes to retrieving the classifier, PRODEN [94] defines the PLL loss as the minimal loss over the candidate label set:

$$\ell_{PLL}(f(x), Y) = \min_{y \in Y} \ell(f(x), y).$$

They further relax the $\min$ operator of the above equality by the dynamic weights as follows:

$$\ell_{PLL}(f(x), Y) = \sum_{y \in Y} w_y \ell(f(x), y),$$

where all w_ys consist of a one-hot vector, expected to reflect the confidence of the label $y \in Y$ being the true label.

7.1.2 CMW-Net Amelioration and Experiments

PRODEN [94] represents the recent SOTA method against such PLL task, through progressively identifying the true labels from the partial label sets, and refining them in turn to ameliorate the classifier. Then by taking the samples with predicted labels as training data, which still contain many wrong annotations, the CMW-Net method (i.e., Algorithm 1 by using meta-data establishing technique introduced in Sec. 4.2) can be readily employed to further improve the classifier.

Following PRODEN [94], two sets of partial label datasets are generated from CIFAR-10 and CIFAR-100, respectively, under different flipping probabilities. As shown in Fig. 8, it is seen that CMW-Net can significantly enhance the performance of the baseline method in both test cases, showing its potential usability in this PLL task. More experimental settings and results are presented in SM.

7.2 Semi-Supervised Learning

7.2.1 Problem Formulation

To reduce the annotation cost for supervised learning, an alternative strategy is to train the classifier with small labeled set as well as a large amount of unlabeled samples. This constitutes the main aim of semi-supervised learning (SSL). Let $D = \{D_L, D_U\}$ denote the entire dataset, including a small labeled dataset $D_L = \{(x_i, y_i)\}_{i=1}^L$ and a large scale

TABLE 10
Performance comparison of Fixmatch w/o CMW-Net on CIFAR-10, CIFAR-100 and ImageNet datasets in test error over 3 trials. The baselines results of CIFAR are copied from [47], and those of ImageNet are copied from [95].

	CIFAR-10			CIFAR-100			ImageNet (10% labels)	
Method	40 labels	250 labels	4000 labels	400 labels	2500 labels	10000 labels	top-1	top5
FixMatch (RA) [47]	13.81 ± 3.37	5.07 ± 0.65	4.26 ± 0.05	48.85 ± 1.75	28.29 ± 0.11	22.60 ± 0.12	32.9	13.3
FixMatch (RA) + CMW-Net	**9.60 ± 0.62**	**4.73 ± 0.15**	**4.25 ± 0.03**	**47.70 ± 1.14**	**27.43 ± 0.12**	**22.55 ± 0.09**	**30.8**	**11.3**

unlabeled dataset $D_U = \{(x_i)\}_{i=1}^{U}$, and $L \ll U$. Formally, SSL aims to solve the following optimization problem [96]:

$$\min_{\mathbf{w}} \sum_{(x_l,y_l)\in D_L} \mathcal{L}_S(x_l, y_l; \mathbf{w}) + \alpha \sum_{x_u \in D_U} \mathcal{L}_U(x_u; \mathbf{w}),$$

where $\mathcal{L}_S$ denotes the supervised loss, e.g., cross-entropy for classification, and $\mathcal{L}_U$ denotes the unsupervised loss, e.g., consistency loss [97] or a regularization term [98]. $\mathbf{w}$ denotes the model parameters and $\alpha > 0$ denotes the compromise parameter balancing two terms.

Generally, different specifications of the unsupervised loss $\mathcal{L}_U$ lead to different SSL algorithms. One commonly used strategy is the Pseudo Labeling approach [99], which aims to sufficiently use labelled data to predict the labels of the unlabeled data, and take these pseudo-labeled data as labeled ones in training (reflected in the term $\mathcal{L}_U$). Recently, the SOTA SSL methods, like VAT [98], MixMatch [100], UDA [97], and FixMatch [47] make good progress to enhance the pseudo labeling capability by using sample augmentation techniques, through encouraging consistency under different augmented data [97]. We take the recent SOTA Fixmatch [47] as a typical example. Denote $\alpha(x_l)$ and $\mathcal{A}(x_u)$ as augmentation operators imposed on labeled and unlabeled samples, respectively. Then the FixMatch model can be written as:

$$\begin{aligned}\min_{\mathbf{w}} & \sum_{(x_l,y_l)\in D_L} \ell(f(\alpha(x_l); \mathbf{w}), y_l) \\ & + \alpha \sum_{x_u \in D_U} \mathbf{1}(\max(z_u \geq \tau) \ell(f(\mathcal{A}(x_u); \mathbf{w}), y_u), \quad (17)\end{aligned}$$

where y_u is the pseudo label on x_u, calculated by $y_u = \arg\max_j z_{uj}, z_u = f(\alpha(x_u); \mathbf{w})$ in iteration. τ is a scalar hyperparameter denoting the threshold above which we retain a pseudo-label. Note that $\mathbf{1}(\max(z_u \geq \tau)$ corresponds to a hard weighting scheme with manually specified hyperparameter τ. Albeit attaining good performance, the above FixMatch model is still with limitation that its hard-thresholding weighting scheme treats all unlabeled (augmented) samples equally, and its involved hyper-parameter τ is often not easily and adaptably specified against different tasks. The method thus still has room for further performance enhancement.

7.2.2 CMW-Net Amelioration and Experiments

To better distinguish clean and noisy pseudo-labels, we can easily substitute the original hard weighting scheme as CMW-Net, to make sample weights capable of more sufficiently reflecting noise extents and adaptable to training data/task. Then the problem (17) can be ameliorated as:

$$\begin{aligned}\min_{\mathbf{w}} & \sum_{(x_l,y_l)\in D_L} \ell(f(\alpha(x_l); \mathbf{w}), y_l) \\ & + \alpha \sum_{(x_u)\in D_U} \mathcal{V}(L_u^{tr}(\mathbf{w}), N_i; \Theta) \ell(f(\mathcal{A}(x_u); \mathbf{w}), y_u).\end{aligned}$$

The algorithm can also be readily designed by integrating the updating step for the meta-parameter Θ into the original algorithm of Fixmatch (the labeled data are naturally used as meta data), so as to make the weighting scheme iteratively extracted together with the classifier parameter $\mathbf{w}$ in an automatic and more likely intelligent manner.

We conduct experiments on several standard SSL image classification benchmarks, including CIFAR-10, CIFAR-100 [82] and ImageNet dataset [89]. Results are shown in Table 10. It is evident that our CMW-Net consistently helps improve the performance of FixMatch, especially under smaller labeled data resources, showing its potential application prospects on this task. More experimental settings and results are presented in SM.

8 Conclusion and Discussion

In this study, we have proposed a novel meta-model, called CMW-Net, for adaptively extracting an explicit sample weighing scheme directly from training data. Compared with current sample weighing approaches, CMW-Net is validated to possess better flexibility against complicated data bias situations with inter-class heterogeneity. Assisted by additional soft pseudo-label information, the proposed method achieves competitive (mostly superior) performance under various data bias cases, including class imbalance, feature independent or dependent label noise, and more practical real-world data bias scenarios, beyond those SOTA methods specifically designed on these robust learning tasks. The extracted weighting schemes can always help faithfully reveal bias insights underlying training data, making the good effect of the method rational and interpretable. Two potential application prospects of CMW-Net are specifically illustrated and substantiated. One is its fine task-transferability of the learned weighting scheme, implying a possible efficiency-speedup methodology for handling robust learning tasks under big data, through avoiding its time-consuming and laborsome weighting function tuning process. The other is its wide range of possible extensional applications for other robust learning tasks, e.g., partial-label learning, semi-supervised learning and selective classification.

In our future investigation, we'll apply the proposed adaptive sample weighting strategies to more robust learning tasks to further validate its generality. Attributed to its relatively concise modeling manner, it is also hopeful to develop deeper and more comprehensive statistical learning understanding for revealing its intrinsic generalization capability across different tasks. Besides, we'll try to build more wider range of connections of our method to previous techniques on exploring data insights, like importance weighting [101]. More sufficient and comprehensive meta-class representation will also be further investigated in our future research.

APPENDIX A
TECHNICAL DETAILS IN SECTION 3

A.1 Derivation of the Weighting Scheme in CMW-Net

We first derive the equivalent forms of the updating steps for CMW-Net and CMW-Net-SL parameters Θ, as expressed in Eqs. (10) and (15), in the main text, respectively.

Recall the update equation of the CWM-Net parameters as follows:

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{m} \sum_{i=1}^{m} \nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))\Big|_{\Theta^{(t)}}. \tag{18}$$

The gradient can be calculated by the following derivation:

$$\begin{aligned}
&\frac{1}{m} \sum_{i=1}^{m} \nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))\Big|_{\Theta^{(t)}} \\
=&\frac{1}{m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))}{\partial \hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial \hat{\mathbf{w}}^{(t+1)}(\Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}} \\
=&\frac{1}{m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))}{\partial \hat{\mathbf{w}}^{(t+1)}(\Theta)} \sum_{j=1}^{n} \frac{\partial \hat{\mathbf{w}}^{(t+1)}(\Theta)}{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)} \frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}.
\end{aligned} \tag{19}$$

Let

$$G_{ij} = \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}}, \tag{20}$$

and by substituting Eqs. (20) and (19) into Eq. (18), we can get:

$$\Theta^{(t+1)} = \Theta^{(t)} + \alpha\beta \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} G_{ij} \right) \frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}. \tag{21}$$

This corresponds to Eq. (10) in the main text.

1) For the CMW-Net, since

$$\hat{\mathbf{w}}^{(t+1)}(\Theta) = \mathbf{w}^{(t)} - \alpha \sum_{i=1}^{n} \mathcal{V}(L_i^{tr}(\mathbf{w}^{(t)}), N_i; \Theta) \nabla_{\mathbf{w}} L_i^{tr}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}}, \tag{22}$$

thus we have

$$\begin{aligned}
&\frac{1}{m} \sum_{i=1}^{m} \nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))\Big|_{\Theta^{(t)}} \\
=&\frac{-\alpha}{m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \sum_{j=1}^{n} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}} \\
=&-\alpha \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right) \frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}.
\end{aligned}$$

2) For the CMW-Net-SL, since

$$\hat{\mathbf{w}}^{(t+1)}(\Theta) = \mathbf{w}^{(t)} - \alpha \sum_{i=1}^{n} \left\{ \mathcal{V}(L_i^{tr}(\mathbf{w}^{(t)}), N_i; \Theta) \nabla_{\mathbf{w}} L_i^{tr}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}} + \left(1 - \mathcal{V}(L_i^{tr}(\mathbf{w}^{(t)}), N_i; \Theta)\right) \nabla_{\mathbf{w}} L_i^{P_{se}}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}} \right\}, \tag{23}$$

where $L_i^{tr}(\mathbf{w}) = \ell(f(x_i; \mathbf{w}), y_i), L_i^{P_{se}}(\mathbf{w}) = \ell(f(x_i; \mathbf{w}), z_i)$, z_i is the pseudo-label for example x_i, we thus have

$$\frac{1}{m} \sum_{i=1}^{m} \nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))\Big|_{\Theta^{(t)}} \tag{24}$$

$$= \frac{-\alpha}{m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \sum_{j=1}^{n} \left[\frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} - \frac{\partial L_j^{P_{se}}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right] \frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}} \tag{25}$$

$$= -\alpha \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{i=1}^{m} \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \left[\frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} - \frac{\partial L_j^{P_{se}}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right] \right) \frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}. \tag{26}$$

Algorithm 3 Learning Algorithm for CMW-Net-SL Model

Input: Training data $\mathcal{D}^{tr}$, meta data $\mathcal{D}^{meta}$, batch size n, temporal ensembling momentum $\alpha \in [0,1)$, weight averaging momentum $\beta \in [0,1)$, mixup hyperparameter $\gamma > 0$,
Output: Classifier parameter $\mathbf{w}^*$

1: Initialize classifier network parameter $\mathbf{w}^{(0)}$. Initialize averaged predictions $\mathbf{z}^{(0)} = \mathbf{0}_{[N\times C]}$, and averaged weights (untrainable) $\mathbf{w}_{WA}^{(0)} = \mathbf{0}$.
2: **for** $t = 0$ **to** $T-1$ **do**
3: $\{x, y\} \leftarrow$ SampleMiniBatch($\mathcal{D}^{tr}, n$).
4: $\{x^{meta}, y^{meta}\} \leftarrow$ SampleMiniBatch($\mathcal{D}^{meta}, m$).
5: Generate mixing coefficient $\lambda \sim Beta(\gamma,\gamma), \lambda = \max(\lambda, 1-\lambda)$.
6: Calculate weight averaging: $\mathbf{w}_{WA}^{(t+1)} = \beta\mathbf{w}_{WA}^{(t)} + (1-\beta)\mathbf{w}^{(t)}$.
7: Calculate temporal ensembling: $\mathbf{z}^{(t+1)} = \alpha\mathbf{z}_i^{(t)} + (1-\alpha)f(x;\mathbf{w}_{WA}^{(t+1)})$.
8: Generate new index sequence idx = torch.randperm(n).
9: Generate $\tilde{x} = \lambda' x + (1-\lambda')x[\text{idx}]$, and let $\tilde{y} = y[\text{idx}], \tilde{z}^{(t+1)} = z^{(t+1)}[\text{idx}]$, $\ell_i = \ell(f(\tilde{x}_i;\mathbf{w}), y_i)$, $\tilde{\ell}_i = \ell(f(\tilde{x}_i;\mathbf{w}), \tilde{y}_i)$. Calculate N_i and $\tilde{N}_i$, representing the numbers of samples contained in the classes to which x_i and $x[\text{idx}]_i$ belong, respectively.
10: Formulate the learning manner of classifier network:

$$\hat{\mathbf{w}}^{(t+1)}(\Theta) = \mathbf{w}^{(t)} - \alpha\sum_{i=1}^{n}\{\lambda\left[\mathcal{V}(\ell_i, N_i;\Theta)\nabla_{\mathbf{w}}\ell(f(\tilde{x}_i;\mathbf{w}), y_i)\Big|_{\mathbf{w}^{(t)}} + (1-\mathcal{V}(\ell_i, N_i;\Theta))\nabla_{\mathbf{w}}\ell(f(\tilde{x}_i;\mathbf{w}), \mathbf{z}_i^{(t+1)})\Big|_{\mathbf{w}^{(t)}}\right] + (1-\lambda)\left[\mathcal{V}(\tilde{\ell}_i, \tilde{N}_i;\Theta)\nabla_{\mathbf{w}}\ell(f(\tilde{x}_i;\mathbf{w}), \tilde{y}_i)\Big|_{\mathbf{w}^{(t)}} + (1-\mathcal{V}(\tilde{\ell}_i, \tilde{N}_i;\Theta))\nabla_{\mathbf{w}}\ell(f(\tilde{x}_i;\mathbf{w}), \tilde{\mathbf{z}}_i^{(t+1)})\Big|_{\mathbf{w}^{(t)}}\right]\}.$$

11: Update parameters of CMW-Net $\Theta^{(t+1)}$ by

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta\frac{1}{m}\sum_{i=1}^{m}\nabla_{\Theta}\ell\left(f(x_i^{(meta)};\hat{\mathbf{w}}^{(t+1)}(\Theta)), y_i^{(meta)}\right)\Big|_{\Theta^{(t)}}.$$

12: Update parameters of classifier $\mathbf{w}^{(t+1)}$ by

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha\sum_{i=1}^{n}\{\lambda\left[\mathcal{V}(\ell_i, N_i;\Theta^{(t+1)})\nabla_{\mathbf{w}}\ell(f(\tilde{x}_i;\mathbf{w}), y_i)\Big|_{\mathbf{w}^{(t)}} + (1-\mathcal{V}(\ell_i, N_i;\Theta^{(t+1)}))\nabla_{\mathbf{w}}\ell(f(\tilde{x}_i;\mathbf{w}), \mathbf{z}_i^{(t+1)})\Big|_{\mathbf{w}^{(t)}}\right] + (1-\lambda)\left[\mathcal{V}(\tilde{\ell}_i, \tilde{N}_i;\Theta^{(t+1)})\nabla_{\mathbf{w}}\ell(f(\tilde{x}_i;\mathbf{w}), \tilde{y}_i)\Big|_{\mathbf{w}^{(t)}} + (1-\mathcal{V}(\tilde{\ell}_i, \tilde{N}_i;\Theta^{(t+1)}))\nabla_{\mathbf{w}}\ell(f(\tilde{x}_i;\mathbf{w}), \tilde{\mathbf{z}}_i^{(t+1)})\Big|_{\mathbf{w}^{(t)}}\right]\}.$$

13: **end for**

Let

$$G_{ij} = \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t+1)}(\Theta)}^{T}\frac{\partial L_j^{tr}(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}}, G'_{ij} = \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t+1)}(\Theta)}^{T}\frac{\partial L_j^{P_{se}}(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}}, \tag{27}$$

by substituting Eqs. (24) and (27) into Eq. (18), we can get:

$$\Theta^{(t+1)} = \Theta^{(t)} + \alpha\beta\sum_{j=1}^{n}\left[\frac{1}{m}\sum_{i=1}^{m}(G_{ij} - G'_{ij})\right]\frac{\partial\mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j;\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}. \tag{28}$$

This corresponds to Eq. (15) in the main text.

A.2 Complete Learning Algorithm of CMW-Net-SL

Recently, some works are presented to extract pseudo soft labels on samples through the clue of the classifier's estimation during the training iterations, and then use such beneficial information to improve the robustness of classifier training especially in the presence of noisy labels. The utilized techniques include temporal ensembling [80], weight averaging, mixup [84], and others. In our experiments, we just directly apply the strategy utilized in ELR [79] and DivideMix [8], which has been verified to be effective in tasks like semi-supervised learning [47], [80] and robust learning [7], [79], to produce pseudo-labels in our CMW-Net-SL algorithm. The complete algorithm is summarized in the above Algorithm 3.

A.3 Convergence Proof of Proposed CMW-Net Learning Algorithm

This section provides the proofs of 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,

$$\mathcal{L}^{meta}(\mathbf{w}^*(\Theta)) = \frac{1}{M}\sum_{i=1}^{M}L_i^{meta}(\mathbf{w}^*(\Theta)), \tag{29}$$

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

$$\mathcal{L}^{tr}(\mathbf{w}; \Theta) = \sum_{i=1}^{N} \mathcal{V}(L_i^{tr}(\mathbf{w}), N_i; \Theta) L_i^{tr}(\mathbf{w}), \tag{30}$$

where $\sum_{i=1}^{N} \mathcal{V}(L_i^{tr}(\mathbf{w}), N_i; \Theta) = 1$.

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

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

$$\nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))\Big|_{\Theta^{(t)}} = -\alpha \sum_{j=1}^{n} \left(\frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right) \frac{\partial \mathcal{V}(L_j^{tr}(\mathbf{w}^{(t)}), N_j; \Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}}. \tag{31}$$

Let $\mathcal{V}_j(\Theta) = \mathcal{V}(L_j^{train}(\mathbf{w}^{(t)}); \Theta)$ and G_{ij} be defined in Eq.(20). Taking gradient of Θ in both sides of Eq.(31), we have

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

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

$$\begin{aligned}
&\left\| \frac{\partial}{\partial \Theta}(G_{ij})\Big|_{\Theta^{(t)}} \frac{\partial \mathcal{V}_j(\Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}} \right\| \\
\leq& \delta \left\| \frac{\partial}{\partial \Theta}(G_{ij})\Big|_{\Theta^{(t)}} \right\| = \delta \left\| \frac{\partial}{\partial \hat{\mathbf{w}}} \left(\frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \Theta}\Big|_{\Theta^{(t)}} \right)\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right\| \\
=& \delta \left\| \frac{\partial}{\partial \hat{\mathbf{w}}} \left(\frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} (-\alpha) \sum_{k=1}^{n} \frac{\partial L_k^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial \mathcal{V}_k(\Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}} \right)\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right\| \\
=& \delta \left\| \left(\frac{\partial^2 L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}^2}\Big|_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} (-\alpha) \sum_{k=1}^{n} \frac{\partial L_k^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial \mathcal{V}_k(\Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}} \right)\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right\| \\
\leq& \alpha n L \rho^2 \delta^2,
\end{aligned} \tag{32}$$

since $\left\| \frac{\partial^2 L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}^2}\Big|_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \right\| \leq L$, $\left\| \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right\| \leq \rho$, $\left\| \frac{\partial \mathcal{V}_j(\Theta)}{\partial \Theta}\Big|_{\Theta^{(t)}} \right\| \leq \delta$. And for the second term we have

$$\left\| (G_{ij}) \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial \Theta^2}\Big|_{\Theta^{(t)}} \right\| = \left\| \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \frac{\partial L_j^{tr}(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial \Theta^2}\Big|_{\Theta^{(t)}} \right\| \leq \mathcal{B}\rho^2, \tag{33}$$

since $\left\| \frac{\partial L_i^{meta}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|^{T}_{\hat{\mathbf{w}}^{(t+1)}(\Theta)} \right\| \leq \rho$, $\left\| \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial \Theta^2}\Big|_{\Theta^{(t)}} \right\| \leq \mathcal{B}$. Combining the above two inequalities Eqs.(32) and (33), we then have

$$\left\| \nabla^2_{\Theta^2} L_i^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}} \right\| \leq \alpha \rho^2 (n \alpha L \delta^2 + \mathcal{B}). \tag{34}$$

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

$$\|\nabla \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta_1)) - \nabla \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta_2))\| \leq L_V \|\Theta_1 - \Theta_2\|, for\ all\ \ \Theta_1, \Theta_2, \tag{35}$$

where $\nabla \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta_1)) = \nabla_{\Theta} L_i^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))\big|_{\Theta_1}$. □

Theorem 1. *Suppose the loss function ℓ is Lipschitz smooth with constant L, and CMW-Net $\mathcal{V}(\cdot, \cdot; \Theta)$ is differential with a δ-bounded gradient and twice differential with its Hessian bounded by $\mathcal{B}$, and the loss function ℓ has ρ-bounded gradient with respect to training/meta data. Let the learning rate $\alpha_t, \beta_t, 1 \leq t \leq T$ be monotonically descent sequences, and satisfy $\alpha_t = \min\{\frac{1}{L}, \frac{c_1}{\sqrt{T}}\}$, $\beta_t = \min\{\frac{1}{L}, \frac{c_2}{\sqrt{T}}\}$, for some $c_1, c_2 > 0$, such that $\frac{\sqrt{T}}{c_1} \geq L$, $\frac{\sqrt{T}}{c_2} \geq L$. Meanwhile, they satisfy $\sum_{t=1}^{\infty} \alpha_t = \infty$, $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$, $\sum_{t=1}^{\infty} \beta_t = \infty$, $\sum_{t=1}^{\infty} \beta_t^2 < \infty$. Then CMW-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,*

$$\min_{0 \leq t \leq T} \mathbb{E}\left[\left\| \nabla \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 \right] \leq \mathcal{O}(\frac{C}{\sqrt{T}}), \tag{36}$$

where C is some constant independent of the convergence process.

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

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

Under the sampled mini-batch Ξ_t, the updating equation can be rewritten as:

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

Since the mini-batch is drawn uniformly from the entire data set, the above update equation can be written as:

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

where $\xi^{(t)} = \nabla \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\big|_{\Xi_t} - \nabla \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))$. 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

$$\begin{aligned} &\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\ =&\left\{\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\} + \left\{\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\right\}. \end{aligned} \tag{37}$$

For the first term, by Lipschitz continuity of $\nabla_{\Theta}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta))$ according to Lemma 1, we can deduce that:

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

For the second term, by Lipschitz smoothness of the meta loss function $\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}))$, we have

$$\begin{aligned} &\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\ \leq&\left\langle \nabla_{\mathbf{w}}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t)}) \right\rangle + \frac{L}{2}\|\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t)})\|_2^2. \end{aligned}$$

Since

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

where Ψ_t denotes the mini-batch drawn randomly from the training dataset in the t-th iteration, $\nabla \mathcal{L}^{train}(\mathbf{w}^{(t)}; \Theta^{(t)}) = \sum_{i=1}^{n} \mathcal{V}(L_i^{tr}(\mathbf{w}^{(t)}), N_i; \Theta^{(t)}) \nabla_{\mathbf{w}^{(t)}} L_i^{tr}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}}$, and $\sum_{i=1}^{n} \mathcal{V}(L_i^{tr}(\mathbf{w}^{(t)}), N_i; \Theta^{(t+1)}) = 1$. Since the mini-batch Ψ_t is drawn uniformly at random, we can rewrite the update equation as:

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

where $\psi^{(t)} = \nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)})|_{\Psi_t} - \nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)})$. Note that $\psi^{(t)}$ are i.i.d. random variables with finite variance, since Ψ_t are drawn i.i.d. with a finite number of samples, and thus $\mathbb{E}[\psi^{(t)}] = 0$, $\mathbb{E}[\|\psi^{(t)}\|_2^2] \leq \sigma^2$. Thus we have

$$\begin{aligned} &\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\ \leq&\left\langle \nabla_{\mathbf{w}}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), -\alpha_t[\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)}) + \psi^{(t)}] \right\rangle + \frac{L}{2}\left\|\alpha_t[\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)}) + \psi^{(t)}]\right\|_2^2 \\ =&\frac{L\alpha_t^2}{2}\left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)})\right\|_2^2 - \alpha_t\left\langle \nabla_{\mathbf{w}}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)}) \right\rangle + \frac{L\alpha_t^2}{2}\left\|\psi^{(t)}\right\|_2^2 \\ &+ L\alpha_t^2\left\langle \nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)}), \psi^{(t)} \right\rangle - \alpha_t\left\langle \nabla_{\mathbf{w}}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \psi^{(t)} \right\rangle \\ \leq&\frac{L\alpha_t^2\rho^2}{2} + \alpha_t\rho\left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)})\right\|_2 + \frac{L\sigma^2\alpha_t^2}{2} + L\alpha_t^2\left\langle \nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)}), \psi^{(t)} \right\rangle - \alpha_t\left\langle \nabla_{\mathbf{w}}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \psi^{(t)} \right\rangle. \end{aligned}$$

The last inequality holds since $\left\langle \nabla_{\mathbf{w}}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)}) \right\rangle \leq \left\|\nabla \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\right\|_2 \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)})\right\|_2$. Thus Eq.(37) satifies

$$\begin{aligned} &\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\ \leq&\frac{L\alpha_t^2\rho^2}{2} + \alpha_t\rho\left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)})\right\|_2 + \frac{L\sigma^2\alpha_t^2}{2} + L\alpha_t^2\left\langle \nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)}; \Theta^{(t)}), \psi^{(t)} \right\rangle - \alpha_t\left\langle \nabla_{\mathbf{w}}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \psi^{(t)} \right\rangle \\ &- (\beta_t - \frac{L\beta_t^2}{2})\left\|\nabla_{\Theta}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2 + \frac{L\beta_t^2}{2}\|\xi^{(t)}\|_2^2 - (\beta_t - L\beta_t^2)\langle \nabla_{\Theta}\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \xi^{(t)}\rangle. \end{aligned}$$

Rearranging the terms, and taking expectations with respect to $\xi^{(t)}$ and $\psi^{(t)}$ on both sides, we can obtain

$$\begin{aligned}
&(\beta_t - \frac{L\beta_t^2}{2}) \left\|\nabla_\Theta \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2 \\
\leq &\frac{L\alpha_t^2\rho^2}{2} + \alpha_t\rho \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2 + \frac{L\sigma^2\alpha_t^2}{2} + \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) + \frac{L\beta_t^2}{2}\sigma^2,
\end{aligned}$$

since $\mathbb{E}[\xi^{(t)}] = 0, \mathbb{E}[\psi^{(t)}] = 0$ and $\mathbb{E}[\|\xi^{(t)}\|_2^2] \leq \sigma^2$. Summing up the above inequalities, we can obtain

$$\begin{aligned}
&\sum_{t=1}^T (\beta_t - \frac{L\beta_t^2}{2}) \left\|\nabla_\Theta \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2 \\
\leq &\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)})(\Theta^{(1)}) - \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(T+1)})(\Theta^{(T+1)}) + \frac{L(\sigma^2+\rho^2)}{2}\sum_{t=1}^T \alpha_t^2 + \rho\sum_{t=1}^T \alpha_t \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L}{2}\sum_{t=1}^T \beta_t^2\sigma^2 \\
\leq &\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)})(\Theta^{(1)}) + \frac{L(\sigma^2+\rho^2)}{2}\sum_{t=1}^T \alpha_t^2 + \rho\sum_{t=1}^T \alpha_t \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L}{2}\sum_{t=1}^T \beta_t^2\sigma^2.
\end{aligned}$$

Furthermore, we can deduce that

$$\begin{aligned}
&\min_t \mathbb{E}\left[\left\|\nabla_\Theta \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2\right] \\
\leq &\frac{\sum_{t=1}^T (\beta_t - \frac{L\beta_t^2}{2})\mathbb{E}\left\|\nabla_\Theta \mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2}{\sum_{t=1}^T \left(\beta_t - \frac{L\beta_t^2}{2}\right)} \\
\leq &\frac{1}{\sum_{t=1}^T (2\beta_t - L\beta_t^2)}\left[\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)})(\Theta^{(1)}) + \frac{L(\sigma^2+\rho^2)}{2}\sum_{t=1}^T \alpha_t^2 + \rho\sum_{t=1}^T \alpha_t \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L}{2}\sum_{t=1}^T \beta_t^2\sigma^2\right] \\
\leq &\frac{1}{\sum_{t=1}^T \beta_t}\left[2\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)})(\Theta^{(1)}) + L(\sigma^2+\rho^2)\sum_{t=1}^T \alpha_t^2 + \rho\sum_{t=1}^T \alpha_t \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L}{2}\sigma^2\sum_{t=1}^T \beta_t^2\right] \\
\leq &\frac{1}{T\beta_T}\left[2\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)})(\Theta^{(1)}) + L(\sigma^2+\rho^2)\sum_{t=1}^T \alpha_t^2 + \rho\sum_{t=1}^T \alpha_t \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L}{2}\sigma^2\sum_{t=1}^T \beta_t^2\right] \\
= &\frac{2\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(1)})(\Theta^{(1)}) + L(\sigma^2+\rho^2)\sum_{t=1}^T \alpha_t^2 + \rho\sum_{t=1}^T \alpha_t \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L}{2}\sigma^2\sum_{t=1}^T \beta_t^2}{T}\max\{L, \frac{\sqrt{T}}{c}\} \\
= &\mathcal{O}(\frac{C}{\sqrt{T}}).
\end{aligned}$$

The third inequality holds since $\sum_{t=1}^T (2\beta_t - L\beta_t^2) = \sum_{t=1}^T \beta_t(2 - L\beta_t) \geq \sum_{t=1}^T \beta_t$, and the final equality holds since $\lim_{T\to\infty}\sum_{t=1}^T \alpha_t^2 < \infty, \lim_{T\to\infty}\sum_{t=1}^T \beta_t^2 < \infty, \lim_{T\to\infty}\sum_{t=1}^T \alpha_t \left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 < \infty$. Thus we can conclude that our algorithm can always achieve $\min_{0\leq t\leq T}\mathbb{E}[\|\nabla \mathcal{L}^{meta}(\Theta^{(t)})\|_2^2] \leq \mathcal{O}(\frac{C}{\sqrt{T}})$ in T steps, and this finishes our proof of Theorem 1. □

Theorem 2. *Suppose that the loss function ℓ is Lipschitz smooth with constant L, and CMW-Net $\mathcal{V}(\cdot,\cdot;\Theta)$ is differential with a δ-bounded gradient and twice differential with its Hessian bounded by $\mathcal{B}$, and the loss function ℓ has ρ-bounded gradient with respect to training/meta data. Let the learning rate $\alpha_t, \beta_t, 1 \leq t \leq T$ be monotonically descent sequences, and satisfy $\alpha_t = \min\{\frac{1}{L}, \frac{c_1}{\sqrt{T}}\}, \beta_t = \min\{\frac{1}{L}, \frac{c_2}{\sqrt{T}}\}$, for some $c_1, c_2 > 0$, such that $\frac{\sqrt{T}}{c_1} \geq L, \frac{\sqrt{T}}{c_2} \geq L$. Meanwhile, they satisfy $\sum_{t=1}^\infty \alpha_t = \infty, \sum_{t=1}^\infty \alpha_t^2 < \infty, \sum_{t=1}^\infty \beta_t = \infty, \sum_{t=1}^\infty \beta_t^2 < \infty$. Then CMW-Net can achieve $\mathbb{E}[\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\|_2^2] \leq \epsilon$ in $\mathcal{O}(1/\epsilon^2)$ steps. More specifically,*

$$\min_{0\leq t\leq T} \mathbb{E}\left[\left\|\nabla \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2\right] \leq \mathcal{O}(\frac{C}{\sqrt{T}}) \tag{38}$$

where C is some constant independent of the convergence process.

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 equation of $\mathbf{w}$ in each iteration as follows:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha\sum_{i=1}^n \mathcal{V}(L_i^{tr}(\mathbf{w}), S_{y_i};\Theta^{(t+1)})\nabla_{\mathbf{w}} L_i^{tr}(\mathbf{w})\Big|_{\mathbf{w}^{(t)}}.$$

Under the sampled mini-batch Ψ_t from the training dataset, the updating equation can be rewritten as:

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

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

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

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

The objective function $\mathcal{L}^{tr}(\mathbf{w};\Theta)$ defined in Eq. (30) can be easily proved to be Lipschitz-smooth with constant L, and have ρ-bounded gradient with respect to training data. Observe that

$$\begin{aligned}&\mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t+1)}) - \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\\ =&\left\{\mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t+1)}) - \mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t)})\right\} + \left\{\mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t)}) - \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\}.\end{aligned} \tag{39}$$

For the first term, by Lipschitz smoothness of the training loss function $\mathcal{L}^{tr}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}))$, we have

$$\begin{aligned}&\mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t)}) - \mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\\ \le&\left\langle\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)}),\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\right\rangle + \frac{L}{2}\left\|\mathbf{w}^{(t+1)} - \mathbf{w}^{(t)}\right\|_2^2\\ =&\left\langle\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)}),-\alpha_t[\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)}) + \psi^{(t)}]\right\rangle + \frac{L\alpha_t^2}{2}\left\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)}) + \psi^{(t)}\right\|_2^2\\ =&-(\alpha_t - \frac{L\alpha_t^2}{2})\left\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L\alpha_t^2}{2}\left\|\psi^{(t)}\right\|_2^2 - (\alpha_t - L\alpha_t^2)\left\langle\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)}),\psi^{(t)}\right\rangle.\end{aligned}$$

For the second term, we have

$$\begin{aligned}&\mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t+1)}) - \mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t)})\\ =&\frac{1}{n}\sum_{i=1}^{n}\left\{\mathcal{V}(\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)}),N_i;\Theta^{(t+1)}) - \mathcal{V}(\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)}),N_i;\Theta^{(t)})\right\}\mathcal{L}_i^{train}(\mathbf{w}^{(t+1)})\\ \le&\frac{1}{n}\sum_{i=1}^{n}\left\{\left\langle\frac{\partial\mathcal{V}(\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)}),N_i;\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}},\Theta^{(t+1)} - \Theta^{(t)}\right\rangle + \frac{\delta}{2}\left\|\Theta^{(t+1)} - \Theta^{(t)}\right\|_2^2\right\}\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)})\\ =&\frac{1}{n}\sum_{i=1}^{n}\left\{\left\langle\frac{\partial\mathcal{V}(\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)}),N_i;\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}},-\beta_t[\nabla_\Theta\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})) + \xi^{(t)}]\right\rangle + \frac{\delta\beta_t^2}{2}\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})) + \xi^{(t)}\right\|_2^2\right\}\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)})\\ =&\frac{1}{n}\sum_{i=1}^{n}\left\{\left\langle\frac{\partial\mathcal{V}(\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)}),N_i;\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}},-\beta_t[\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})) + \xi^{(t)}]\right\rangle + \frac{\delta\beta_t^2}{2}\left(\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2 + \|\xi^{(t)}\|_2^2 +\right.\right.\\ &\left.\left.2\left\langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})),\xi^{(t)}\right\rangle\right)\right\}L_i^{tr}(\mathbf{w}^{(t+1)}).\end{aligned}$$

Therefore, for Eq.(39), we have:

$$\begin{aligned}&\mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t+1)}) - \mathcal{L}^{tr}(\mathbf{w}^{(t+1)};\Theta^{(t)})\\ \le&\frac{1}{n}\sum_{i=1}^{n}\left\{\left\langle\frac{\partial\mathcal{V}(\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)}),N_i;\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}},-\beta_t[\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})) + \xi^{(t)}]\right\rangle + \frac{\delta\beta_t^2}{2}\left(\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2 + \|\xi^{(t)}\|_2^2 +\right.\right.\\ &\left.\left.2\left\langle\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)})),\xi^{(t)}\right\rangle\right)\right\}L_i^{tr}(\mathbf{w}^{(t+1)}) - (\alpha_t - \frac{L\alpha_t^2}{2})\left\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L\alpha_t^2}{2}\left\|\psi^{(t)}\right\|_2^2\\ &-(\alpha_t - L\alpha_t^2)\left\langle\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)}),\psi^{(t)}\right\rangle.\end{aligned}$$

Taking expectation of the both sides of the above inequality and based on $\mathbb{E}[\xi^{(t)}] = 0, \mathbb{E}[\psi^{(t)}] = 0$, we have

$$\begin{aligned}&\mathbb{E}[\mathcal{L}^{train}(\mathbf{w}^{(t+1)};\Theta^{(t+1)})] - \mathbb{E}[\mathcal{L}^{train}(\mathbf{w}^{(t)};\Theta^{(t)})]\\ &\le\mathbb{E}\frac{1}{n}\sum_{i=1}^{n}\left\{\left\langle\frac{\partial\mathcal{V}(\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)}),N_i;\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}},-\beta_t\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\rangle + \frac{\delta\beta_t^2}{2}\left(\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2 + \|\xi^{(t)}\|_2^2\right)\right\}\\ &L_i^{tr}(\mathbf{w}^{(t+1)}) - (\alpha_t - \frac{L\alpha_t^2}{2})\mathbb{E}\left\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 + \frac{L\alpha_t^2}{2}\mathbb{E}[\|\psi^{(t)}\|_2^2].\end{aligned}$$

Summing up the above inequalities over $t = 1, ..., T$ in both sides and rearranging the terms, we obtain

$$
\begin{aligned}
&\sum_{t=1}^{T}\left(\alpha_t - \frac{L\alpha_t^2}{2}\right)\mathbb{E}\left\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2 \\
\leq&\sum_{t=1}^{T}\beta_t\mathbb{E}\frac{1}{n}\sum_{i=1}^{n}\left\|L_i^{tr}(\mathbf{w}^{(t+1)})\right\|\left\|\frac{\partial\mathcal{V}(\mathcal{L}_i^{tr}(\mathbf{w}^{(t+1)}),N_i;\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}\right\|\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\| + \sum_{t=1}^{T}\frac{L\alpha_t^2}{2}\mathbb{E}[\|\psi^{(t)}\|_2^2] + \mathbb{E}[\mathcal{L}^{train}(\mathbf{w}^{(1)};\Theta^{(1)})] \\
&- \mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(T+1)};\Theta^{(T+1)})] + \sum_{t=1}^{T}\frac{\delta\beta_t^2}{2}\left\{\frac{1}{n}\sum_{i=1}^{n}\left\|L_i^{tr}(\mathbf{w}^{(t+1)})\right\|\left(\mathbb{E}\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2 + \mathbb{E}\left\|\xi^{(t)}\right\|_2^2\right)\right\} \\
\leq&\delta M\sum_{t=1}^{T}\beta_t\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\| + \sum_{t=1}^{T}\frac{L\alpha_t^2}{2}\sigma^2 + \mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(1)};\Theta^{(1)})] + \sum_{t=1}^{T}\frac{\delta\beta_t^2}{2}\left\{M(\rho^2+\sigma^2)\right\} < \infty.
\end{aligned}
$$

The last inequality holds since $\sum_{t=0}^{\infty}\alpha_t^2 < \infty, \sum_{t=0}^{\infty}\beta_t^2 < \infty, \sum_{t=1}^{T}\beta_t\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\| < \infty$, and $\frac{1}{n}\sum_{i=1}^{n}\|L_i^{train}(\mathbf{w}^{(t)})\| \leq M$, i.e., the sum of limited number of samples' losses is bounded. Thus we have

$$
\begin{aligned}
&\min_t\mathbb{E}\left[\left\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2\right] \\
\leq&\frac{\sum_{t=1}^{T}\left(\alpha_t - \frac{L\alpha_t^2}{2}\right)\mathbb{E}\left\|\nabla_\Theta\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\|_2^2}{\sum_{t=1}^{T}\left(\alpha_t - \frac{L\alpha_t^2}{2}\right)} \\
\leq&\frac{\delta M\sum_{t=1}^{T}\beta_t\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\| + \sum_{t=1}^{T}\frac{L\alpha_t^2}{2}\sigma^2 + \mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(1)};\Theta^{(1)})] + \sum_{t=1}^{T}\frac{\delta\beta_t^2}{2}\left\{M(\rho^2+\sigma^2)\right\}}{\sum_{t=1}^{T}\left(\alpha_t - \frac{L\alpha_t^2}{2}\right)} \\
\leq&\frac{\delta M\sum_{t=1}^{T}\beta_t\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\| + \sum_{t=1}^{T}\frac{L\alpha_t^2}{2}\sigma^2 + \mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(1)};\Theta^{(1)})] + \sum_{t=1}^{T}\frac{\delta\beta_t^2}{2}\left\{M(\rho^2+\sigma^2)\right\}}{\sum_{t=1}^{T}\alpha_t} \\
\leq&\frac{\delta M\sum_{t=1}^{T}\beta_t\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\| + \sum_{t=1}^{T}\frac{L\alpha_t^2}{2}\sigma^2 + \mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(1)};\Theta^{(1)})] + \sum_{t=1}^{T}\frac{\delta\beta_t^2}{2}\left\{M(\rho^2+\sigma^2)\right\}}{T\alpha_t} \\
\leq&\frac{\delta M\sum_{t=1}^{T}\beta_t\left\|\nabla\mathcal{L}^{meta}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t)}))\right\| + \sum_{t=1}^{T}\frac{L\alpha_t^2}{2}\sigma^2 + \mathbb{E}[\mathcal{L}^{tr}(\mathbf{w}^{(1)};\Theta^{(1)})] + \sum_{t=1}^{T}\frac{\delta\beta_t^2}{2}\left\{M(\rho^2+\sigma^2)\right\}}{T}\max\{L, \frac{\sqrt{T}}{c}\} \\
=&\mathcal{O}(\frac{C}{\sqrt{T}}).
\end{aligned}
$$

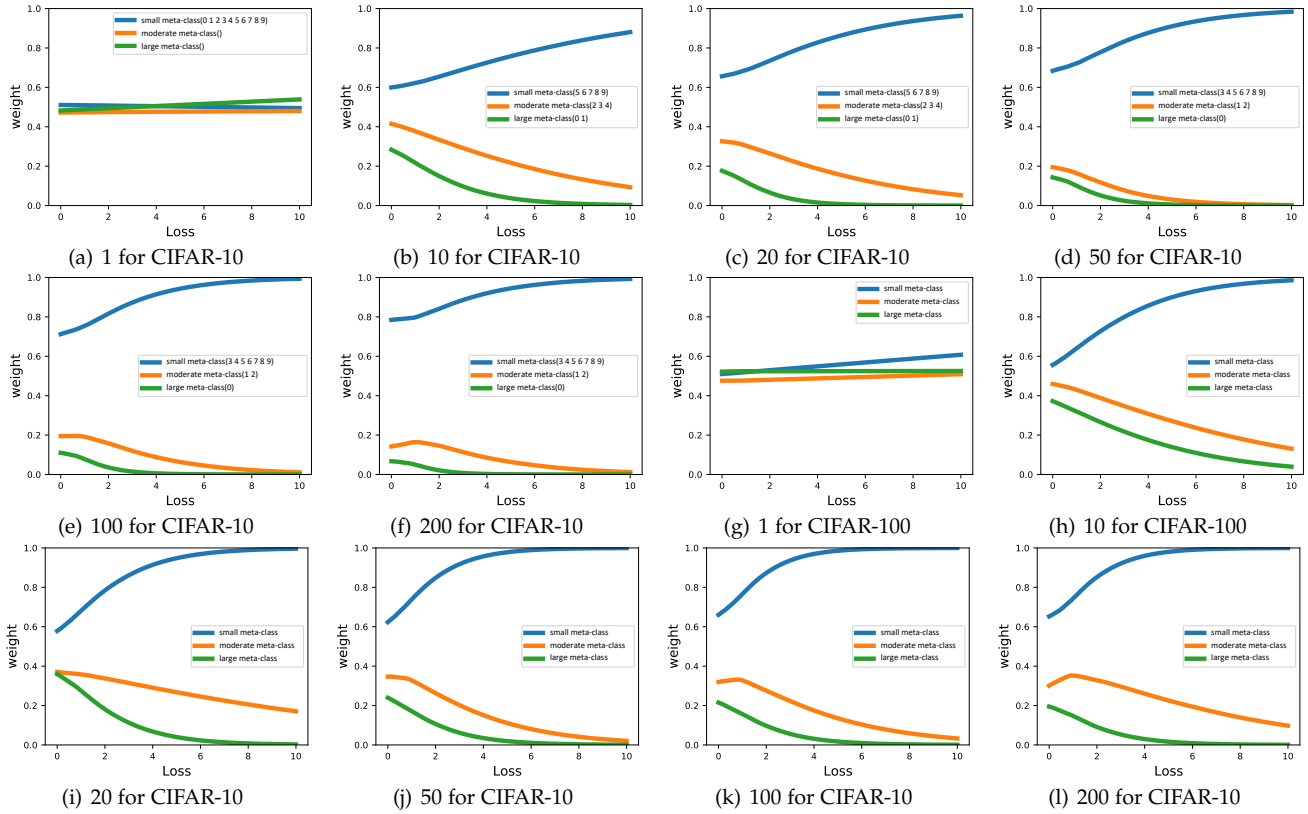
The third inequality holds since $\sum_{t=1}^{T}(2\alpha_t - L\alpha_t^2) = \sum_{t=1}^{T}\alpha_t(2 - L\alpha_t) \geq \sum_{t=1}^{T}\alpha_t$. Thus we can conclude that our algorithm can always achieve $\min_{0\leq t\leq T}\mathbb{E}\left[\left\|\nabla\mathcal{L}^{tr}(\mathbf{w}^{(t)};\Theta^{(t)})\right\|_2^2\right] \leq \mathcal{O}(\frac{C}{\sqrt{T}})$ in T steps, and this completes our proof of Theorem 2. □

A.4 Pytorch codes of our algorithm

The following is the Pytorch codes of our algorithm, which is essily completed based on the code of MW-Net. The main difference from MW-Net is to re-define the structure of meta-model (CMW-Net) and generate the meta-class labels in advance. The completed training code is avriable at https://github.com/xjtushujun/CMW-Net.

```
def norm_func(v_lambda):
norm_c = torch.sum(v_lambda)
if norm_c != 0:
v_lambda_norm = v_lambda / norm_c
else:
v_lambda_norm = v_lambda
return  v_lambda_norm


class share(MetaModule):
def __init__(self, input, hidden1, hidden2):
super(share, self).__init__()
self.layer = nn.Sequential( MetaLinear(input, hidden1), nn.ReLU(inplace=True) )

def forward(self, x):
output = self.layer(x)
return output
```

```
class task(MetaModule):
def __init__(self, hidden2, output, num_classes):
super(task, self).__init__()
self.layers = nn.ModuleList()
for i in range(num_classes):
self.layers.append(nn.Sequential( MetaLinear(hidden2, output), nn.Sigmoid() ))

def forward(self, x, num, c):
si = x.shape[0]
output = torch.tensor([]).cuda()
for i in range(si):
output = torch.cat(( output, self.layers[c[num[i]]]( x[i].unsqueeze(0) ) ),0)

return output


# The structure of CMW-Net
class VNet(MetaModule):
def __init__(self, input, hidden1, hidden2, output, num_classes):
super(VNet, self).__init__()
self.feature = share(input, hidden1, hidden2)
self.classfier = task(hidden2, output, num_classes)

def forward(self, x, num, c):
num = torch.argmax(num, -1)
output = self.classfier( self.feature(x), num, c )
return output

optimizer_a = torch.optim.SGD(model.params(), args.lr, momentum=args.momentum, nesterov=args.nesterov,
                                                    weight_decay=args.weight_decay)
optimizer_c = torch.optim.Adam(vnet.params(), 1e-3, weight_decay=1e-4)

# Generating meta-class labels
es = Kmeans(3)
es.fit(train_loader.dataset.targets)
c = es.labels_

for iters in range(num_iters):
adjust_learning_rate(optimizer_a, iters + 1)
model.train()
data, target = next(iter(train_loader))
data, target = data.to(device), target.to(device)
meta_model.load_state_dict(model.state_dict())
y_f_hat = meta_model(data)
cost = F.cross_entropy(y_f_hat, target, reduce=False)
cost_v = torch.reshape(cost, (len(cost), 1))
v_lambda = vnet(cost_v.data, target.data, c)
v_lambda_norm = norm_func(v_lambda)
l_f_meta = torch.sum(cost_v * v_lambda_norm)
meta_model.zero_grad()
grads = torch.autograd.grad(l_f_meta,(meta_model.params()),create_graph=True)
meta_model.update_params(lr_inner=meta_lr,source_params=grads)

data_meta,target_meta = next(iter(train_meta_loader))
data_meta,target_meta = data_meta.to(device),target_meta.to(device)
y_g_hat = meta_model(data_meta)
l_g_meta = F.cross_entropy(y_g_hat, target_meta)
optimizer_c.zero_grad()
l_g_meta.backward()
optimizer_c.step()

y_f = model(data)
cost_w = F.cross_entropy(y_f, target, reduce=False)
cost_v = torch.reshape(cost_w, (len(cost_w), 1))
with torch.no_grad():
w_new = vnet(cost_v,target, c)
w_v = norm_func(w_new)
l_f = torch.sum(cost_v * w_v)
optimizer_a.zero_grad()
l_f.backward()
optimizer_a.step()
```



(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-10 (j) 50 for CIFAR-10 (k) 100 for CIFAR-10 (l) 200 for CIFAR-10

Fig. 9. Weighting schemes learned by CMW-Net on CIFAR-10-LT and CIFAR-100-LT with imbalance factors ranging from 1 to 200.

APPENDIX B
MORE EXPERIMENTAL RESULTS AND EXPERIMENTAL SETTINGS IN SECTION 4

B.1 Class Imbalance Experiments

In this series of experiments, we use ResNet-32 [1] as the classifier network 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 100 after 160 and 180 epoch (for a total 200 epochs). The learning rate of CMW-Net is fixed as 10^{-4}, and the weight decay of CMW-Net is fixed as 10^{-5}. The batch size is set as 100 for all experiments. We randomly selected fixed images per class in every epoch from the training set as the meta-data set and the number of selected images for each class is the same as the number of the least class.

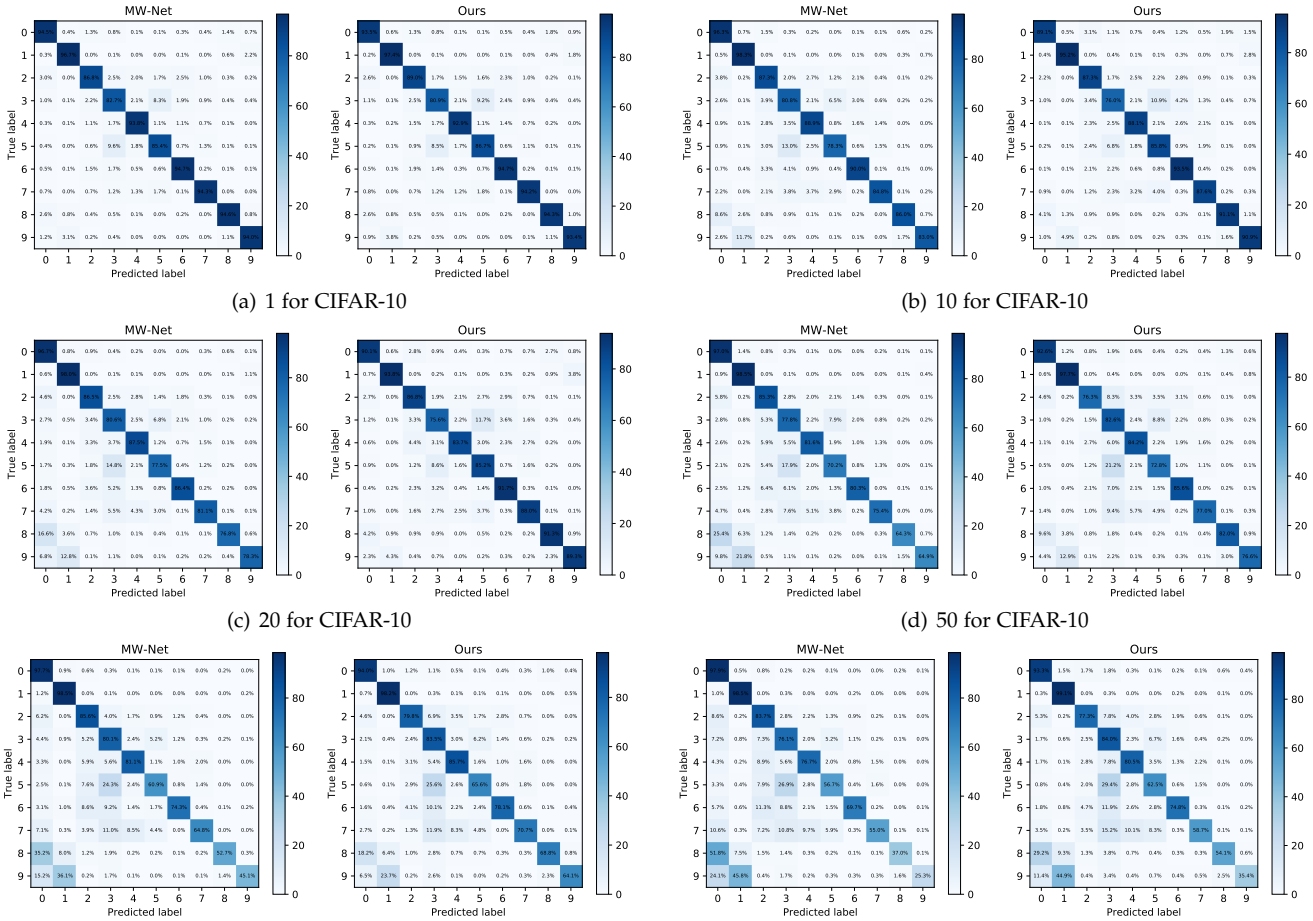
Fig. 9 shows the weighting schemes learned by CMW-Net on CIFAR-10-LT and CIFAR-100-LT, under different imbalance settings. It can be seen that our CMW-Net can adaptively learn proper weighting schemes according to different degrees of class imbalance. For example, when the dataset is balanced, CMW-Net tends to learn approximately similar weighting functions for all three meta-classes. When the degree of class imbalance becomes more significant, the weighting schemes extracted from different meta-classes tend to be more largely varied, showing their different internal bias characteristics.

In Fig. 10, we further plot the confusion matrices produced by the MW-Net and CMW-Net methods, respectively. As can be easily seen, CMW-Net consistently outperforms MW-Net, and CMW-Net more evidently improves the accuracies of MW-Net under larger class imbalance rate. Specifically, CMW-Net gets more prominent performance gain on tail classes, and meanwhile maintains the performance on head classes.

B.2 Feature-independent Label Noise Experiment

In this series of experiments, we adopted an 18-layer PreAct Resnet [102] as our classifier network, with softmax cross-entropy loss by SGD with a momentum 0.9, a weight decay 5×10^{-4}. For CMW-Net, we set the initial learning rate as 0.1 and the learning rate of classification network is divided by 10 after 80 and 100 epoch (for a total 120 epochs). For the CMW-Net-SL, we set the initial learning rate as 0.01 and the learning rate of classification network is divided by 10 after 150 epoch (for a total 300 epochs) following by Dividemix [8]. The batch size is specified as 128 for all experiments. We adopt Adam optimizer to optimize CMW-Net and the learning rate of CMW-Net is fixed as 10^{-3}, and the weight decay of CMW-Net is fixed as 10^{-4}. We repeat the experiments with 3 random trials and report the mean value and standard deviation.

Motivated by M-correction [7] and Dividemix [8], we selected the meta data at each epoch according to the training loss. Specifically, we explore to create the meta dataset dynamically along iteration, based on the high-quality clean samples as well as its high-quality pseudo labels from the training set (with lowest losses) as an unbiased estimator of the clean



(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

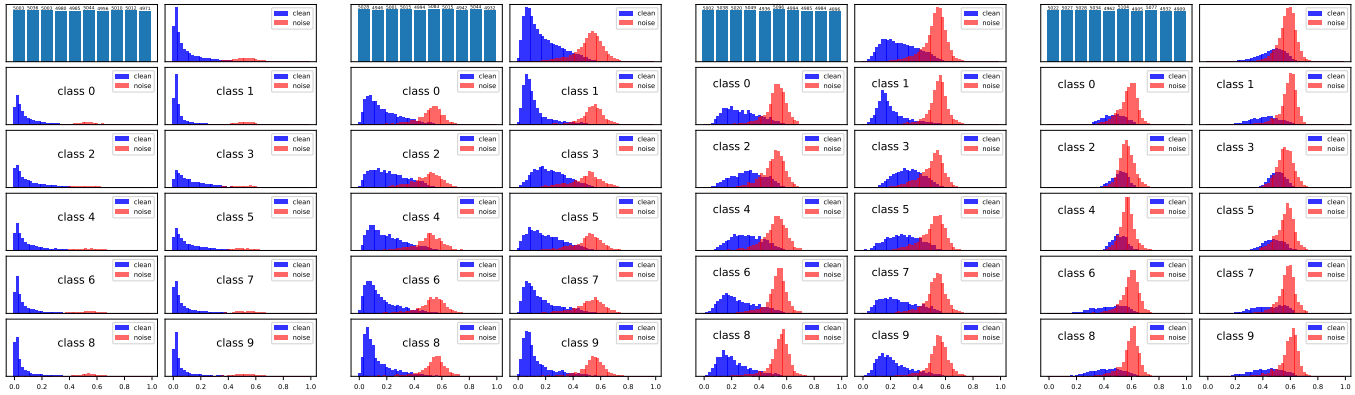
Fig. 10. Confusion matrices obtained by (left) MW-Net and (right) CMW-Net on CIFAR-10-LT with imbalance factors ranging from 1 to 200.

data-label distribution in each iteration of our algorithm. To make the meta dataset balanced, we selected 10 images per class. In this case, the performance of meta dataset can be served as an indicator of whether CMW-Net is trained to filter noisy samples and generalize to clean test distribution.

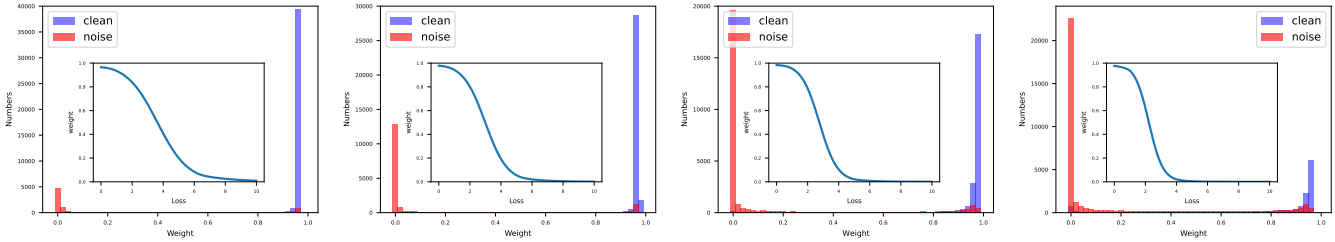
Such meta dataset generation strategy may lack of diversity patterns to characterize the latent clean data-label distribution. To overcome this, we explore to utilize mixup technique [84] to enrich the variety of our proposed meta dataset distribution while maintaining the unbiasedness in terms of clean test distribution. The hyperparameter of convex combination is randomly sampled from a Beta distribution $Beta(1, 1)$. Extensive experiments have verified the effectiveness of such created meta dataset from training dataset. Such property makes our meta-learning algorithm feasible to be applied to real-world biased datasets, where it is generally hard to collect an ideal high-quality extra clean meta dataset. We also use such meta dataset generation strategy all our noisy labels experiments as well as all real-world biased datasets.

Figs. 11 and 13 shows the empirical pdfs of cross-entropy loss for each class on CIFAR-10 dataset under symmetric and asymmetric noises with varying noise rates, respectively. The corresponding weighting functions and weight distributions over the training examples learned by MW-Net [9] and the proposed CMW-Net are also depicted. It can be easily observed that compared with MW-Net, CMW-Net has better flexibility to deal with both training data bias cases, even for inter-class heterogenous data biases. Specifically, the proposed CMW-Net can adaptively adjust its weighting schemes to adapt variations of noise rates, and behaves consistently with the underlying data biased patterns, naturally leading to its better performance on distinguishing clean and noisy images. Note that even for high noise rate (e.g., 80%) scenarios, our method still shows fine capability of distinguishing clean and noisy images.

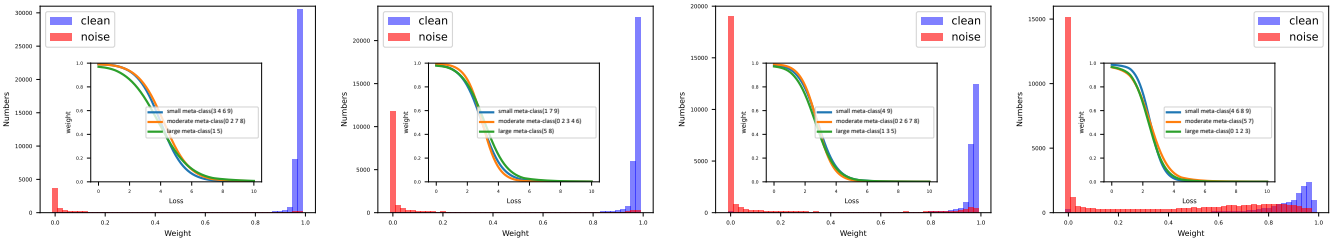
To further improve the learning effect of CMW-Net, we introduce additional soft label supervision to build the CMW-Net-SL strategy. Figs. 12 and 14 show the confusion matrices learned by two methods, respectively. Specifically, the confusion matrices obtained by CMW-Net almost correspond to the noise transition matrices, and those calculated by CMW-Net-SL contain the refurbished labels by soft labels. Although the noise rate was in relatively high levels (e.g., 60% and 80%), most of the diagonal entries had probability larger than 0.95, implying the effectiveness of CMW-Net-SL on its fine label correction ability. This side information is thus validated to be able to compensate beneficially to the sample reweighting learning, and ameliorate both weighting scheme extracting and robust classifier learning in a stable way.



(a) Empirical pdf of cross-entropy loss for each class on CIFAR-10 dataset with varying noise rates under symmetric noise.



(b) Weighting functions and histograms of all sample weights over all training examples learned by MW-Net under symmetric noise.



(c) Weighting functions and histograms of all sample weights over all training examples learned by CMW-Net under symmetric noise.

Fig. 11. (a) Empirical pdf of the cross-entropy loss calculated on all samples of each class on CIFAR-10 with varying noise rates (from left to right, the noise rates are 20%, 40%, 60%, 80%) under symmetric noise; (b)(c) The weighting functions and histograms of all sample weights over all training examples learned by MW-Net and CMW-Net.

B.3 Feature-dependent Label Noise Experiment

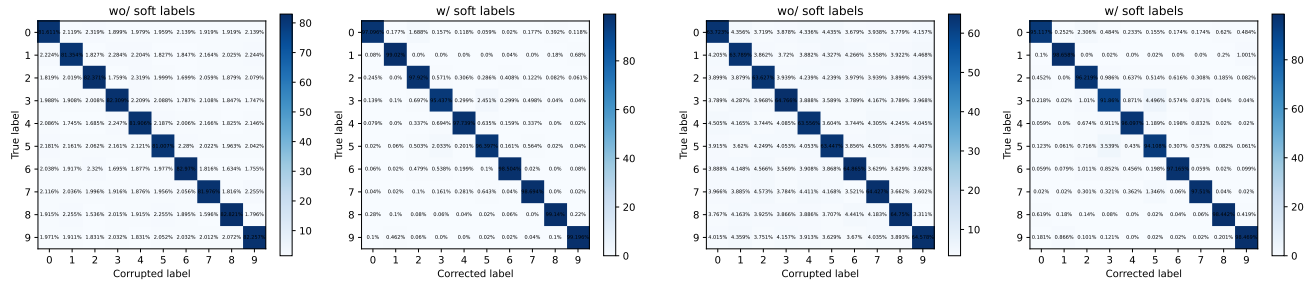
In this series of experiments, we use ResNet-34 [1] as the classifier network, with softmax cross-entropy loss by SGD with a momentum 0.9, a weight decay 5×10^{-4} and an initial learning rate 0.1. The learning rate of ResNet-34 is set as CosineAnnealingWarmRestarts [103]. The learning rate of CWN-Net is fixed as 10^{-3}, and the weight decay of CMW-Net is fixed as 10^{-4}. The batch size is 128 for all experiments. We repeat the experiments with 3 random trials and report the mean value and standard deviation.

B.4 Additional Open-set Label Noise Experiment

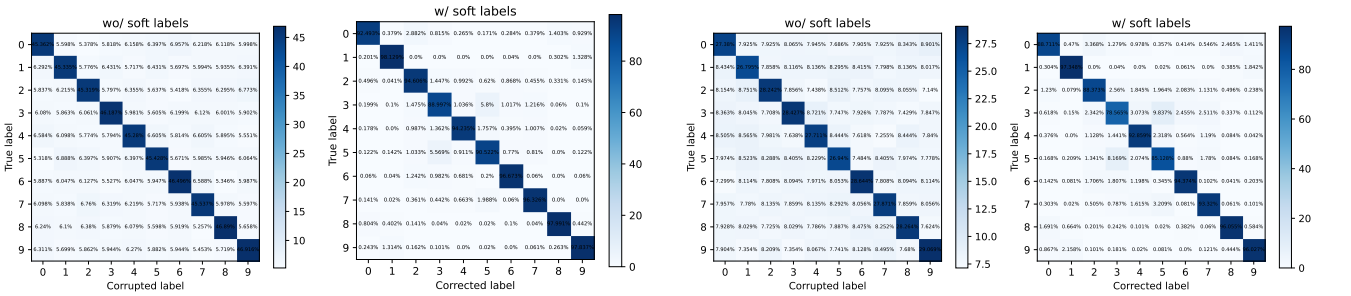
Open-set noise experiments use training samples that do not belong to any of the original class in the dataset considered in the classification task. Following [104], we yield CIFAR-10 with open-set noise by randomly replacing 40% of its training images with images from CIFAR-100. We used wide ResNet-28-2 [105] as the base classifier network with softmax cross-entropy loss by SGD with a momentum 0.9, a weight decay 5×10^{-4}. We set the initial learning rate of classification network as 0.1 and the learning rate is divided by 10 after 80 and 100 epoch (for a total 120 epochs). The batch size is 128 for all experiments. We adopt Adam optimizer to learn CMW-Net, with a learning rate 10^{-3}, and a weight decay 10^{-4}. We repeat the experiments with 3 random trials and report the mean value and standard deviation. We adopt the meta-data generation strategy as introduced in Sec. 4.2 of the main text, by randomly selecting 10 images per class at every epoch from the training set as the meta-data set. We train the network 20 epochs with cross-entropy loss for a warm-up to get the meta dataset stably.

The compared methods include: 1) ERM: use standard cross-entropy loss to train DNNs; 2) Forward [70]: correct the prediction by the label transition matrix; 3) GCE [6]: behave as a robust loss to handle the noisy labels; 4) M-correction [7] and 5) DivideMix [8]: use different label correction methods; 6) L2RW [26] and 7) MW-Net [9]: represent the typical sample reweighting methods by meta-learning.

The classification accuracy on CIFAR-10 noisy datasets with 40% open-set noise is reported in Table 11. As can be seen, our method evidently outperforms all other competing methods, verifying that our model is capable of learning more



(a) Symmetry Noise (20%) (b) Symmetry Noise (40%)



(c) Symmetry Noise (60%) (d) Symmetry Noise (80%)

Fig. 12. Confusion matrices obtained by CMW-Net without (left) or with (right) soft label amelioration on CIFAR-10 with symmetry noise with varying noise rates ranging from 20% to 80%.

TABLE 11
Test accuracy (%) of all comparison methods under open-set noise on CIFAR-10.

Methods	ERM	Forward [70]	GCE [6]	M-correction [7]	DivideMix [8]	L2RW [26]	MW-Net [9]	CMW-Net	CMW-Net-SL
Accuracy	84.17±0.80	84.63±0.80	85.96 ± 0.72	89.71 ± 0.53	90.16±0.40	83.60±0.24	84.78 ± 0.51	84.81 ± 0.51	**92.12 ± 0.18**

accurate representation directly from datasets with open-set noisy labels. Such capability supports that our method can be applied to learning from web-search data possibly containing such type of open-set noisy labels, e.g., WebVision [14].

B.5 Ablation Study

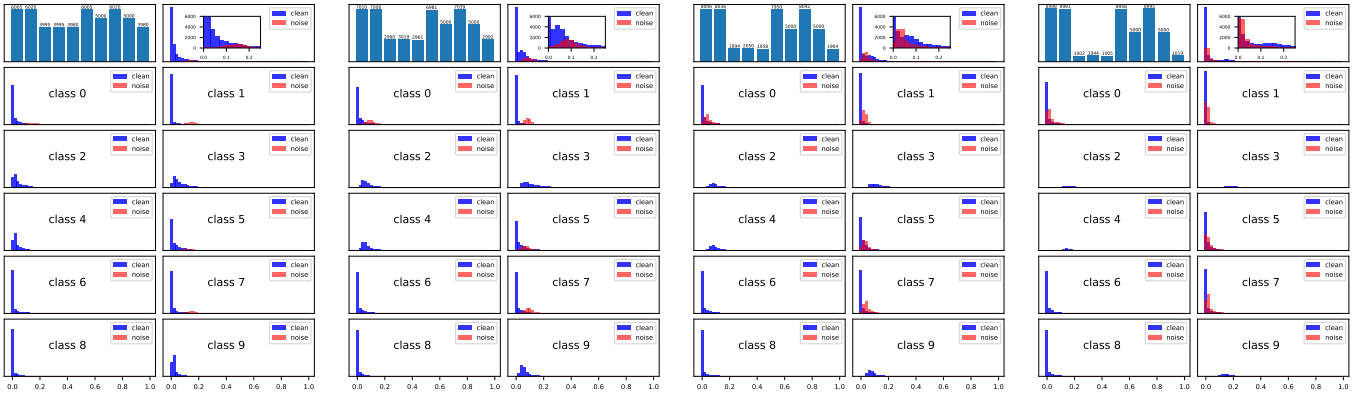
We perform ablation study to verify the effectiveness of two important components involved in our method: 1) the number of meta-classes; 2) whether to use an extra clean meta-data or using the automatic meta-data-generation strategy as proposed in Sec. 4.2 of the main text. As shown in Fig. 15(a), by setting the number of meta-class as three, our method can consistently adapt to inter-class heterogenous data bias. Actually, by setting $K = 3$, where the training classes/tasks are separated as small, moderate, large-scales, correspondingly, all our experiments can achieve a stably fine performance. Furthermore, by observing Fig. 15(b), we can see that the utilized meta-data-generation strategy is practicable for dealing with real-world noisy datasets, in which an extra ideal clean meta data is always hard to be collected.

APPENDIX C
MORE EXPERIMENTAL RESULTS AND EXPERIMENTAL SETTINGS IN SECTION 5

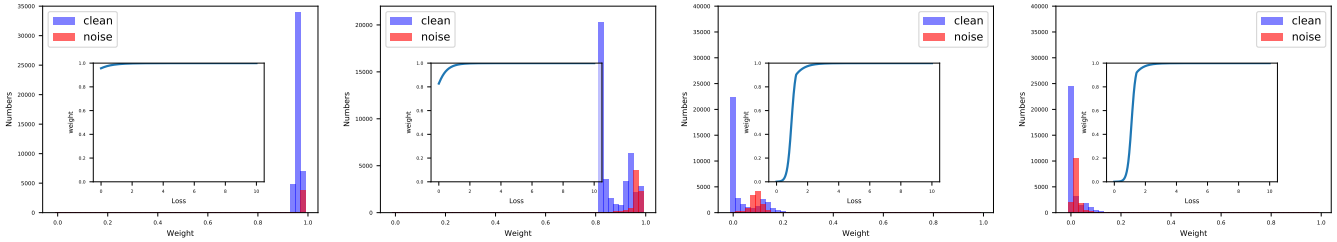
C.1 Learning with Real-world Noisy Datasets

Animal-10N. ANIMAL-10N [78] contains 55,000 human-labeled online images for 10 animals with confusable appearance. The estimated label noise rate is 8%. Following previous works [78], [83], 50,000 images are exploited as the training set and the left for testing. Following SELFIE [78], we use VGG-19 [106] with batch normalization as the classifier network. The SGD optimizer is employed to train the network with a momentum 0.9, a weight decay 1×10^{-3} for 100 epochs. We use an initial learning rate of 0.1, which is divided by 5 at 50% and 75% of the total number of epochs. The batch size is 128. We repeat the experiments with 3 random trials and report the mean value and standard deviation.

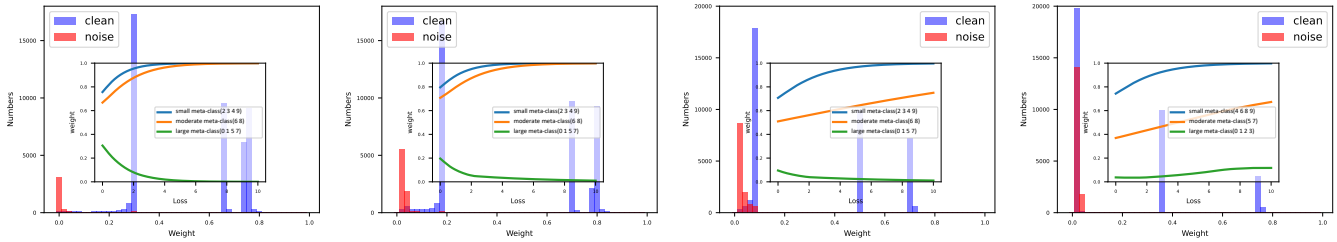
Mini-WebVision. As the full dataset of WebVision is very large, we follow [37] to use a mini version, which contains the first 50 classes of the Google subset of the data for a total of about 61,000 images. Following the standard protocol [37], we test the trained model on the WebVision validation set and the ImageNet validation set. Following C2D [88], we used ResNet-50 architecture as the classifier network for training. For self-supervised pre-training, we directly use the pretrained self-supervised models released at https://github.com/ContrastToDivide/C2D, which is based on the SimCLR implementation. We trained the network with softmax cross-entropy loss by SGD with a momentum 0.9, a weight decay 5×10^{-4}. We set the initial learning rate as 0.01 and the learning rate of classification network is divided by 10 after 50 epoch (for a total 90 epochs). The learning rate of CMW-Net is fixed as 10^{-4}, and the weight decay of CMW-Net is fixed as 10^{-5}.



(a) Empirical pdf of cross-entropy loss for each class on CIFAR-10 dataset with varying noise rates under asymmetric noise.

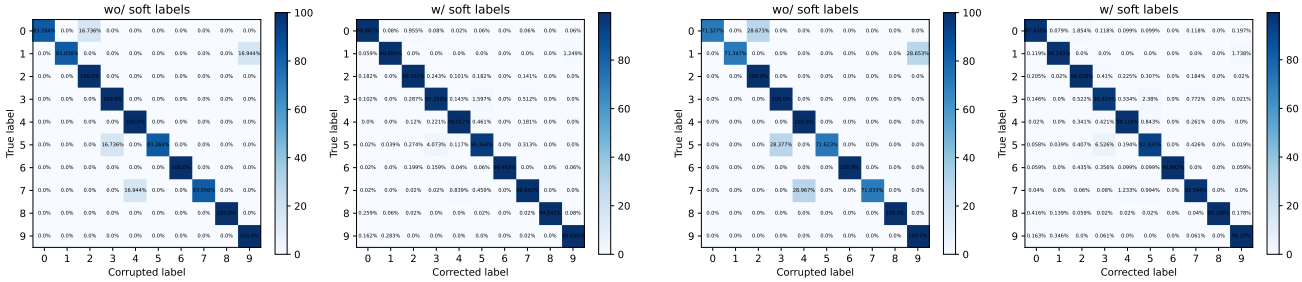


(b) Weighting functions and histograms of all sample weights over all training examples learned by CMW-Net under asymmetric noise.



(c) Weighting functions and weight distributions over the training examples learned by our CMW-Net under asymmetric noise.

Fig. 13. (a) Empirical pdf of the cross-entropy loss calculated on all samples of each class on CIFAR-10 with varying noise rates (from left to right, the noise rates are 20%, 40%, 60%, 80%) under asymmetric noise; (b)(c) The weighting functions and histograms of all sample weights over all training examples learned by MW-Net and CMW-Net.



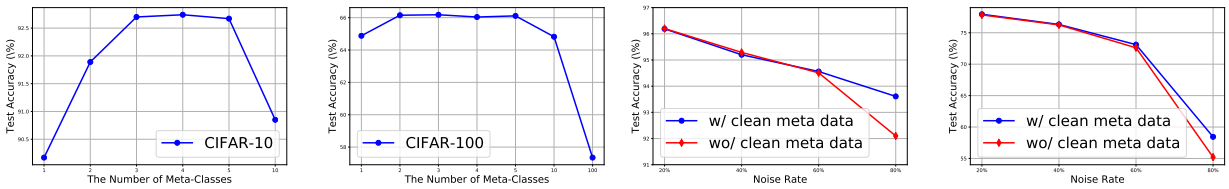
(a) Asymmetry Noise (20%)

(b) Asymmetry Noise (40%)

(c) Asymmetry Noise (60%)

(d) Asymmetry Noise (80%)

Fig. 14. Confusion matrices obtained by CMW-Net without (left) or with (right) soft label amelioration on CIFAR-10 with asymmetry noise with varying noise rates ranging from 20% to 80%.



(a) On different numbers of meta-classes (b) On whether to use extra clean meta data

Fig. 15. Some ablation studies for parameter setting issues of our proposed method.

TABLE 12
Validation accuracy of InceptionResNet-v2 with transferable CMW-Net on full WebVision and ImageNet validation sets.

Methods	WebVision		ILSVRC12		Methods	WebVision		ILSVRC12	
	top1	top5	top1	top5		top1	top5	top1	top5
ERM	69.7	87.0	62.9	83.6	MentorMix [93]	74.3	90.5	67.5	87.2
MentorNet [37]	70.8	88.0	62.5	83.0	HAR [92]	75.0	90.6	67.1	86.7
Curriculumnet [16]	72.1	89.2	64.8	84.9	ERM + CMW-Net	**75.4**	**91.5**	**67.8**	**87.6**

The batch size is 64. We adopt the meta-data-generation strategy as introduced in Sec. 4.2 of the main text, to randomly select 10 images per class at every epoch from the training set as the meta-data set for the above two real-world biased datasets.

More typical noisy examples corrected by the proposed method on Animal-10N and Mini-WebVision are shown in Figs. 16 and 17, respectively. This further demonstrates our method's capability of recovering these easily confusable samples.

C.2 Webly Supervised Fine-Grained Recognition

WebFG-496. This dataset consists of three sub-datasets: Web-aircraft, Web-bird, and Web-car. WebFG-496 reuses the category labels of three famous manually labeled fine-grained datasets, FGVC-Aircraft, CUB200-2011, and Stanford Cars, which contain 100 types of airplanes, 200 species of birds, and 196 categories of cars, respectively, by collecting images from the Internet. It contains 53,339 training images with total 496 classes. The testing data take the testing sets in the original FGVC-Aircraft, CUB200-2011, and Stanford Cars. We used Bilinear-CNN [107] as the classifier network. The network is pre-trained on ImageNet, and then fine-tuned on three sub-datasets of WebFG496. Following [107], we adopt a two-stage training strategy. We firstly freeze the convolutional layer parameters and only optimize the last fully connected layers with the learning rate and batch size being 10^{-3} and 64 for total 200 epoch. Then we optimize the parameters of all layers in the fine-tuned model with learning rate and batch size being set as 10^{-4} and 32, respectively, for total 200 epoch. The learning rate of CMW-Net is fixed as 10^{-3}, and the weight decay of CMW-Net is fixed as 10^{-4}. We adopt the meta-data-generation strategy, as introduced in Sec. 4.2 of the main test, to randomly select 10 images per class at every epoch from the training set as the meta-data set.

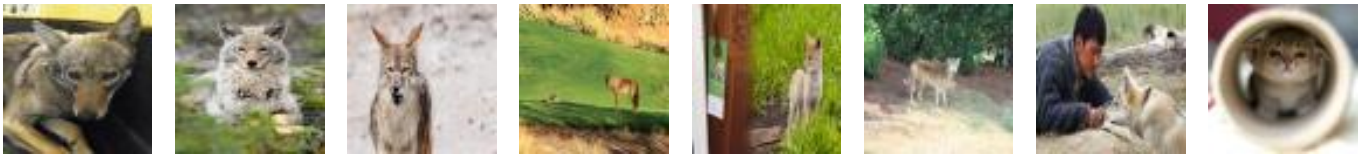
APPENDIX D
MORE EXPERIMENTAL RESULTS AND EXPERIMENTAL SETTINGS IN SECTION 6

ImageNet-LT. The dataset is constructed as a long-tailed version of the original ImageNet-2012 [89] by sampling a subset following the Pareto distribution with the power value 6. It totally has 115.8K images from 1000 categories with maximally 1280 images per class and minimally 5 images per class. Following OLTR [5], besides the overall top-1 classification accuracy over all classes, we also calculate the accuracy of three disjoint subsets: many-shot classes (each with over training 100 samples), medium-shot classes (each with 20-100 training samples) and few-shot classes (each under 20 training samples). We adopt the two-stage training protocol following [5]. We use a Resnet-10 model initialized from scratch (i.e., random initialization) as the classifier model. We train the model with softmax cross-entropy loss by SGD with a momentum 0.9, a weight decay 5×10^{-4}, an initial learning rate 0.1 and a batch size of 128 for 30 epochs, and divide learning rate by 10 at 10 epoch. The transferred CMW-Net is used at the first stage to produce proper sample weights for robust training. And it follows training protocol in [5] at the second stage.

WebVision. WebVision [14] contains 2.4 million images crawled from Google and Flickr using 1,000 labels shared with the ImageNet dataset. Its training set is both heteroskedastic label noise and class imbalanced (more detailed statistics can be found in [14]), and it is considered as a popular benchmark for robust learning in the presence of heavy label noises. We trained Inception-ResNet v2 [108] with softmax cross-entropy loss by SGD with a momentum 0.9, a weight decay 10^{-4}, an initial learning rate 0.1 and a batch size of 256. The learning rate is divided by 10 after 30 and 40 epoch (for a total 50 epochs). The transferred CMW-Net is used at every iteration to produce proper sample weights for robust training. The test performance on full WebVision and ILSVRC12 dataset are presented in Table 12.

(a) Samples selected from Animal-10N [78]. The original training label is cat.

(b) Samples selected from Animal-10N [78]. The original training label is lynx.



(c) Samples selected from Animal-10N [78]. The original training label is wolf.

(d) Samples selected from Animal-10N [78]. The original training label is coyote.

(e) Samples selected from Animal-10N [78]. The original training label is chimpanzee.

(f) Samples selected from Animal-10N [78]. The original training label is cachimpanzeet.

(g) Samples selected from Animal-10N [78]. The original training label is hamster.

(h) Samples selected from Animal-10N [78]. The original training label is guinea pig.

Fig. 16. Examples of randomly selected samples with noisy labels corrected by our method on Animal-10N dataset [78]. The original training labels and generated pseudo-labels by our model are shown in red and blue, respectively.

(a) Samples selected from mini-WebVision [14]. The original training labels are electric ray, crampfish, numbfish, torpedo.

(b) Samples selected from mini-WebVision [14]. The original training labels are house finch, linnet, Carpodacus mexicanus.

(c) Samples selected from mini-WebVision [14]. The original training labels are robin, American robin, Turdus migratorius.

(d) Samples selected from mini-WebVision [14]. The original training labels are leatherback turtle, leatherback, leathery turtle, Dermochelys coriacea.

(e) Samples selected from mini-WebVision [14]. The original training labels are common iguana, iguana, Iguana iguana.

(f) Samples selected from mini-WebVision [14]. The original training labels are American chameleon, anole, Anolis carolinensis.

(g) Samples selected from mini-WebVision [14]. The original training labels are Komodo dragon, Komodo lizard, dragon lizard, giant lizard, Varanus komodoensis.

(h) Samples selected from mini-WebVision [14]. The original training label is tailed frog.

Fig. 17. Examples of randomly selected samples with noisy labels corrected by our method on mini-WebVision [14]. The original training labels and generated pseudo-labels by model are shown in red and blue, respectively.

APPENDIX E
MORE EXPERIMENTAL RESULTS AND EXPERIMENTAL SETTINGS IN SECTION 7

E.1 Partial-Label Learning

We adopted two training stages to solve the problem. At the first stage, we train the network using the recent SOTA method, PRODEN [94], for 100 epochs and then we can get the training data with single noisy labels by one-hot encoding of the model predictions. Now the partial-label learning problem becomes a conventional learning problem with all samples attached with single noisy labels. It is naturally to use the proposed method to further deal with such a problem. Thus at the second stage, we trained the network with obtained single noisy labeled data using the proposed CMW-Net method. We use a SGD optimizer with a momentum 0.9, a weight decay 5×10^{-4}, an initial learning rate 0.1, a batch size 128. The learning rate is divided by 10 after 80 and 100 epoch (for a total 120 epochs). We adopt Adam optimizer to learn CMW-Net. The learning rate of CMW-Net is fixed as 10^{-3}, and the weight decay of CMW-Net is fixed as 10^{-4}. We repeat the experiments with 3 random trials and report the mean value and standard deviation. We adopt the meta-data-generation strategy as introduced in Sec. 4.2 of the main text, by randomly selecting 10 images per class at every epoch from the training set as the meta-data set.

Following PRODEN [94], we manually corrupt these datasets into partially labeled versions by a flipping probability q, where $q = P(\tilde{y} = 1|y = 0)$ gives the probability that a false positive label $\tilde{y}$ is flipped from a negative label y. We adopt a binomial flipping strategy: $c - 1$ independent experiments are conducted on all training examples, each determining whether a negative label is flipped with probability q. Then for the examples that none of the negative labels are flipped, we additionally flip a random negative label to the candidate label set for ensuring all the training examples are partially labeled. We use five widely used benchmark datasets, including MNIST [109], Fashion-MNIST [110], Kuzushiji-MNIST [111], CIFAR-10 and CIFAR-100 [82]. For MNIST, Fashion-MNIST, and Kuzushiji-MNIST datasets, we use 5-layer perceptron (MLP), and for CIFAR-10 and CIFAR-100 dataset, we use ResNet-32 [1] as the classifier network.

Table 13 reports the mean test accuracies with standard deviation on five benchmark datasets. It can be seen that our method can consistently outperform the baseline PRODEN method under both less-partial circumstances $q = 0.1$ and stronger-partial circumstances $q = 0.7$. Observing that PRODEN method behaves under strong-partial circumstances similarly as that under less-partial circumstances, which implies it tends to easily overfit to pseudo-labels estimated by model prediction. Considering that the obtained results are calculated on the basis of PRODEN method as single noisy labels dataset, our method can alleviate such pseudo-label issue and bring further performance improvement for such a partial label learning problem. Through introducing our method as a post-processing learning, we can obtain a more robust model based on the over-confident information. Particularly, our method can improve PRODEN method about 4-8 points on CIFAR-10 and 8-13 points on CIFAR-100 in classification accuracy. Applying our method to more partial-label learning method to obtain more robust result is thus potentially expected, and we leave this research for our future study.

E.2 Semi-Supervised Learning

Following Fixmatch [47], we consider the settings by giving 4/25/400 labeled images for each class on CIFAR-10 and 4/25/100 labeled images for each class on CIFAR-100. We used WRN-28-2/WRN-28-8 for CIFAR-10/CIFAR-100 as the classifier network with an initial learning rate 0.03, a batch size 64 for label data and 448 for unlabel data. For ImageNet experiment, we use 10 % of the training data for each class as labeled and treat the rest as unlabeled examples. We used ResNet-50 as the classifier and the batch size for labeled (unlabeled) images is 64 (320) with initial learning rate 0.03. We adopt RandAugment [112] as the strong augmentation for this experiment. We adopt Adam optimizer to learn CMW-Net. The learning rate of CMW-Net is fixed as 10^{-3}, and the weight decay of CMW-Net is fixed as 10^{-4}.

Table 14 shows the classification error rates on CIFAR-10/100 and ImageNet. From the table, one can observe that our method improves Fixmatch method and achieves the best performance under all label conditions on all datasets. Specifically, our method achieves around 2 points improvement on ImageNet as compared with Fixmatch, showing that our method is capable of finely handling such large-scale sample weight learning issue.

E.3 Selective Classification

E.3.1 Problem Formulation

We consider the selective classification problem in DNNs (supervised learning with a rejection option), which allows the learned classifier to abstain whenever they are not sufficiently confident in their prediction, so as to finely detect and control statistical uncertainties of training cases [115]. Specifically, let $P(X, Y)$ be the underlying joint distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}, \mathcal{Y}$ denote the sample and label spaces, respectively, and $f : \mathcal{X} \to \mathcal{Y}$ be the prediction function (DNNs here). The expected risk is:

$$R(f) = \mathbb{E}_{P(X,Y)}[\ell(f(x), y)],$$

where $\ell : Y \times Y \to \mathbb{R}^+$ is the loss function. Given a dataset $D^{tr} = \{(x_i, y_i)\}_{i=1}^N$ where all (x_i, y_i)s are i.i.d. drawn from $X \times Y$, the empirical risk is then specified as

$$\hat{R}_{D^{tr}}(f) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i).$$

TABLE 13
Performance comparison of classification accuracy (%) on partially labeled benchmark datasets.

Dataset	Methods	Classifier	$q = 0.1$	$q = 0.3$	$q = 0.5$	$q = 0.7$
MNIST	PRODEN	MLP	98.59 ± 0.01	98.07 ± 0.03	98.42 ± 0.03	98.09 ± 0.05
	PRODEN+ Ours	MLP	**98.99 ± 0.01**	**98.83 ± 0.02**	**98.57 ± 0.04**	**98.33 ± 0.02**
Fashion-MNIST	PRODEN	MLP	89.51 ± 0.07	88.79 ± 0.06	88.32 ± 0.07	87.21 ± 0.13
	PRODEN+ Ours	MLP	**90.47 ± 0.02**	**90.07 ± 0.05**	**89.38 ± 0.12**	**87.84 ± 0.13**
Kuzushiji-MNIST	PRODEN	MLP	91.07 ± 0.07	90.24 ± 0.12	88.31 ± 0.14	85.55 ± 0.58
	PRODEN+ Ours	MLP	**93.07 ± 0.04**	**91.65 ± 0.03**	**88.86 ± 0.07**	**86.11 ± 0.17**
CIFAR-10	PRODEN	ResNet-32	82.09 ± 0.05	81.70 ± 0.58	80.72 ± 1.08	76.24 ± 1.35
	PRODEN+ Ours	ResNet-32	**89.77 ± 0.36**	**88.01 ± 0.27**	**86.04 ± 0.32**	**80.57 ± 1.33**
–	–	–	$q = 0.03$	$q = 0.05$	$q = 0.07$	$q = 0.10$
CIFAR-100	PRODEN	ResNet-32	48.06 ± 0.95	47.07 ± 1.32	46.49 ± 1.73	46.30 ± 1.98
	PRODEN+ Ours	ResNet-32	**61.22 ± 0.03**	**60.25 ± 0.17**	**59.17 ± 0.17**	**54.64 ± 0.15**

TABLE 14
Performance comparison of our method with SOTA methods trained on CIFAR-10, CIFAR-100 and ImageNet datasets in terms of test error over 3 trials. Results for all baselines are directly copied from [47].

	CIFAR-10			CIFAR-100			ImageNet
Method	40 labels	250 labels	4000 labels	400 labels	2500 labels	10000 labels	10% labels
Π-Model [113]	-	54.26 ± 3.97	14.01 ± 0.38	-	57.25 ± 0.48	37.88 ± 0.11	-
Pseudo-Labeling [99]	-	49.78 ± 0.43	16.09 ± 0.28	-	57.38 ± 0.46	36.21 ± 0.19	-
Mean Teacher [114]	-	32.32 ± 2.30	9.19 ± 0.19	-	53.91 ± 0.57	35.83 ± 0.24	-
MixMatch [100]	47.54 ± 11.50	11.05 ± 0.86	6.42 ±0.10	67.61 ± 1.32	39.94 ± 0.37	28.31 ± 0.33	-
UDA [97]	29.05 ± 5.93	8.82 ± 1.08	4.88 ± 0.18	59.28 ± 0.88	33.13 ± 0.22	24.50 ± 0.25	-
FixMatch [47]	13.81 ± 3.37	5.07 ± 0.65	4.26 ± 0.05	48.85 ± 1.75	28.29 ± 0.11	22.60 ± 0.12	32.9 (top1), 13.3 (top5)
FixMatch + CMW-Net	**9.6 ± 0.62**	**4.73 ± 0.15**	**4.25 ± 0.03**	**47.7 ± 1.14**	**27.43 ± 0.12**	**22.55 ± 0.09**	**30.8** (top1), **11.3** (top5)

The selective classifier is then defined as a pair of functions (f, g), where $g : \mathcal{X} \to \mathbb{R}$ is a selection function that reveals the underlying uncertainty of inputs. Specifically, given input x, (f, g) outputs:

$$(f, g)(x) = \begin{cases} f(x), & \text{if } g(x) \geq \tau \\ \text{Abstain} & \text{otherwise} \end{cases},$$

i.e., the model abstains from making a prediction when selection function $g(x)$ falls bellow a predetermined threshold τ. We call $g(x)$ the uncertainty score of x, and different methods tend to use different g. The coverage is defined as the probability mass of the non-rejected region in $\mathcal{X}$, expressed as:

$$\phi(g) = \mathbb{E}_{P(X)}[g(x)],$$

and its empirical coverage is

$$\hat{\phi}_{D^{tr}}(g) = \frac{1}{m} \sum_{i=1}^{N} g(x_i).$$

The selective risk of (f, g) is defined as

$$R(f, g) = \frac{\mathbb{E}_{P(X,Y)}[\ell(f(x), y)g(x)]}{\phi(g)},$$

and empirical version is

$$\hat{R}_{D^{tr}}(f, g) = \frac{\frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i)g(x_i)}{\hat{\phi}_{D^{tr}}(g)}.$$

The SelectiveNet [115] tries to optimize the objective

$$\mathcal{L} = \alpha \mathcal{L}_{D^{tr}}(f, g) + (1 - \alpha)\hat{R}_{D^{tr}}(h),$$

where

$$\mathcal{L}_{D^{tr}}(f, g) = \hat{R}_{D^{tr}}(f, g) + \lambda \max(0, c - \hat{\phi}_{D^{tr}}(g))^2,$$

c is the given coverage, and α, λ control the relative importance of each term. As stated in [115], the auxiliary cross-entropy loss $\hat{R}_{D^{tr}}(h)$ exposes the main body block to all training samples throughout the training process to avoid SelectiveNet overfitting to the wrong subset of the training set.

TABLE 15
Selective classification error (%) on CIFAR-10, CIFAR-100 datasets for various coverage rates (%) for SelectiveNet (left in each panel) and SelectiveNet+CMW-Net (right in each panel). The better result in each case is highlighted in bold.

Dataset	Coverage	200		100		50		20		10		0	
		SelectiveNet	Ours	SelectiveNet	Ours	SelectiveNet	Ours	SelectiveNet	Ours	SelectiveNet	Ours	SelectiveNet	Ours
CIFAR-10	100	55.50 ± 0.44	**55.44 ± 0.08**	34.94 ± 0.94	**33.31 ± 0.23**	25.23 ± 0.48	**22.49 ± 0.23**	18.16 ± 0.02	**15.15 ± 0.43**	14.62 ± 0.57	**12.23 ± 0.12**	6.79 ± 0.03	**6.02 ± 0.07**
	95	52.63 ± 1.48	**52.10 ± 0.08**	32.60 ± 0.98	**30.95 ± 0.20**	22.72 ± 0.49	**21.50 ± 0.34**	15.57 ± 0.04	**13.21 ± 0.41**	12.07 ± 0.53	**9.94 ± 0.06**	4.16 ± 0.09	**4.00 ± 0.11**
	90	50.83 ± 1.34	**50.63 ± 0.07**	30.62 ± 0.11	**29.49 ± 0.10**	20.65 ± 0.49	**19.65 ± 1.00**	13.37 ± 0.13	**11.41 ± 0.40**	10.04 ± 0.41	**8.01 ± 0.16**	2.43 ± 0.08	**2.29 ± 0.15**
	85	48.95 ± 0.99	**47.91 ± 0.12**	28.85 ± 0.33	**28.21 ± 0.13**	18.84 ± 0.43	**17.89 ± 0.84**	11.61 ± 0.03	**9.62 ± 0.35**	8.32 ± 0.21	**6.34 ± 0.15**	1.43 ± 0.08	**1.17 ± 0.02**
	80	46.99 ± 0.76	**45.11 ± 0.11**	27.27 ± 0.46	**26.32 ± 0.30**	17.33 ± 0.32	**16.31 ± 0.65**	10.07 ± 0.01	**8.03 ± 0.30**	6.91 ± 0.12	**4.80 ± 0.15**	0.86 ± 0.06	**0.80 ± 0.01**
	75	45.09 ± 0.62	**42.19 ± 0.01**	25.85 ± 0.65	**24.33 ± 0.26**	16.02 ± 0.30	**14.64 ± 0.35**	8.88 ± 0.12	**6.52 ± 0.30**	5.69 ± 0.11	**3.59 ± 0.09**	0.48 ± 0.02	**0.55 ± 0.03**
	70	43.10 ± 0.47	**39.21 ± 0.30**	24.36 ± 0.76	**22.36 ± 0.19**	14.79 ± 0.29	**13.13 ± 0.23**	7.81 ± 0.22	**5.29 ± 0.27**	4.75 ± 0.16	**2.57 ± 0.11**	0.32 ± 0.01	**0.35 ± 0.04**
CIFAR-100	100	68.74 ± 0.42	**65.62 ± 0.25**	65.85 ± 0.15	**60.77 ± 0.09**	61.21 ± 0.19	**55.70 ± 0.22**	55.04 ± 0.39	**46.68 ± 0.14**	49.12 ± 0.38	**40.46 ± 0.31**	27.72 ± 0.35	**25.75 ± 0.23**
	95	67.41 ± 0.43	**64.13 ± 0.33**	64.41 ± 0.16	**59.08 ± 0.06**	59.55 ± 0.22	**53.81 ± 0.20**	53.07 ± 0.39	**44.46 ± 0.13**	46.99 ± 0.34	**37.97 ± 0.33**	24.99 ± 0.38	**23.03 ± 0.24**
	90	66.09 ± 0.48	**62.48 ± 0.28**	63.01 ± 0.17	**57.36 ± 0.04**	57.81 ± 0.20	**51.84 ± 0.20**	51.12 ± 0.39	**42.24 ± 0.18**	44.89 ± 0.33	**35.59 ± 0.27**	22.59 ± 0.31	**20.43 ± 0.18**
	85	64.65 ± 0.54	**60.79 ± 0.22**	61.51 ± 0.16	**55.53 ±0.04**	56.21 ± 0.27	**49.76 ± 0.21**	49.24 ± 0.32	**40.05 ± 0.13**	42.81 ± 0.20	**33.24 ± 0.27**	20.31 ± 0.33	**17.98 ± 0.21**
	80	63.09 ± 0.54	**59.00 ± 0.26**	59,85 ± 0.11	**53.57 ± 0.09**	54.25 ± 0.20	**47.62 ± 0.15**	47.15 ± 0.34	**37.54 ± 0.26**	40.67 ± 0.23	**30.95 ± 0.21**	18.17 ± 0.28	**15.45 ± 0.01**
	75	61.50 ± 0.57	**57.10 ± 0.13**	58.11 ± 0.06	**51.42 ± 0.18**	52.19 ± 0.20	**45.24 ± 0.19**	45.03 ± 0.38	**35.14 ± 0.28**	38.65 ± 0.36	**28.41 ± 0.11**	16.32 ± 0.42	**12.97 ± 0.03**
	70	59.72 ± 0.66	**54.93 ± 0.11**	56.21 ± 0.06	**49.10 ± 0.16**	50.04 ± 0.32	**42.56 ± 0.29**	42.77 ± 0.34	**32.55 ± 0.33**	36.42 ± 0.40	**25.89 ± 0.20**	14.63 ± 0.59	**10.69 ± 0.05**

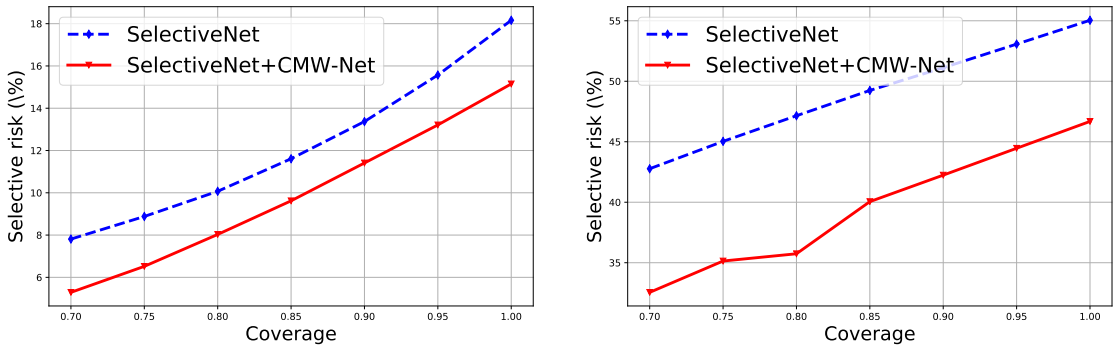


Fig. 18. Risk coverage curves of SelectiveNet w/o CMW-Net strategy on (left) CIFAR-10 and (right) CIFAR100 under imbalance factor 20.

It can be seen that the rationality of the auxiliary cross-entropy loss $\hat{R}_{D^{tr}}(h)$ still inclines to be negatively affected by the data biased issues, like commonly existed class imbalance and noisy label cases in practical datasets. It is thus natural to employ the proposed CMW-Net on the term for making the learned SelectiveNet with better robustness to training samples.

E.3.2 CMW-Net Amelioration and Experiments

We can then readily ameliorate the selective classification by embedding CMW-Net weighting schemes into its optimization problem. Specifically, provided a meta dataset as $D^{meta} = \{x_i^{meta}, y_i^{meta}\}_{i=1}^{M}$, the objective of the problem is then reformulated as the following bi-level problem:

$$\Theta^* = \arg\min_{\Theta} \ \mathcal{L}_{D^{meta}}(f_\Theta^*, g_\Theta^*)$$

$$\{f_\Theta^*, g_\Theta^*\} = \arg\min_{f,g} \ \alpha\mathcal{L}_{D^{tr}}(f,g) + (1-\alpha)\sum_{i=1}^{N}\mathcal{V}(\ell_i, N_i; \Theta)\ell(h(x_i), y_i).$$

Through properly assigning sample weights to the loss terms for all training samples, it is expected to better eliminate the negative influence brought by complicated data biases.

We use long-tailed versions of CIFAR-10 and CIFAR-100 datasets under different imbalance factors for performance evaluation. The generation strategy is similar to that introduced in Sec. 4.1 of the main text. The baseline method is the recent SOTA method for this task: SelectiveNet [115]. We use the VGG-16 network [106] with batch normalization [116] and dropout [117] as the classifier network in experiments. The network is optimized using SGD with initial learning rate of 0.1, momentum of 0.9, weight decay of 5×10^{-4}, batch size of 128, and total training epoch of 300. The learning rate is decayed by 0.5 in every 25 epochs. As the meta-data-generation strategy as introduced in Sec. 4.2 of the main text, we randomly select 10 images per class at every epoch from the training set as the meta-data set.

The obtained experimental results are summarized in Table 15 and Fig. 18. Specifically, the figure compares the risk-coverage curves of SelectiveNet equipped with and without CMW-Net for weighting its sample loss. It is easy to see the performance gain brought to the method by CMW-Net. From the figure, one can more comprehensively observe that selective classification errors of SelectiveNet consistently grow as we increase the degree of class imbalance. Comparatively, under the assistance of CMW-Net, the errors can be consistently reduced in all cases. These results validate the usefulness of CMW-Net for this specific learning task under biased data.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *ACL*, 2019.
[3] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.
[4] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *CVPR*, 2019.
[5] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," in *CVPR*, 2019.
[6] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *NeurIPS*, 2018.
[7] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," in *ICML*, 2019.
[8] J. Li, R. Socher, and S. C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," in *ICLR*, 2019.
[9] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," in *NeurIPS*, 2019.
[10] W. Bi, L. Wang, J. T. Kwok, and Z. Tu, "Learning to predict from crowdsourced data." in *UAI*, 2014.
[11] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" in *ICML*, 2019.
[12] C. G. Northcutt, A. Athalye, and J. Mueller, "Pervasive label errors in test sets destabilize machine learning benchmarks," *arXiv preprint arXiv:2103.14749*, 2021.
[13] S. Yun, S. J. Oh, B. Heo, D. Han, J. Choe, and S. Chun, "Re-labeling imagenet: from single to multi-labels, from global to localized labels," in *CVPR*, 2021.
[14] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "Webvision database: Visual learning and understanding from web data," *arXiv preprint arXiv:1708.02862*, 2017.
[15] J. Shu, Z. Xu, and D. Meng, "Small sample learning in big data era," *arXiv preprint arXiv:1808.04572*, 2018.
[16] S. Guo, W. Huang, H. Zhang, C. Zhuang, D. Dong, M. R. Scott, and D. Huang, "Curriculumnet: Weakly supervised learning from large-scale web images," in *ECCV*, 2018.
[17] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *ICML*, 2017.
[18] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *ICLR*, 2017.
[19] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng, "Deep long-tailed learning: A survey," *arXiv preprint arXiv:2110.04596*, 2021.
[20] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE TNNLS*, 2013.
[21] G. Algan and I. Ulusoy, "Image classification with deep learning in the presence of noisy labels: A survey," *Knowledge-Based Systems*, vol. 215, p. 106771, 2021.
[22] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour, "Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis," *Medical Image Analysis*, vol. 65, 2020.
[23] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *arXiv preprint arXiv:2007.08199*, 2020.
[24] B. Han, Q. Yao, T. Liu, G. Niu, I. W. Tsang, J. T. Kwok, and M. Sugiyama, "A survey of label-noise representation learning: Past, present and future," *arXiv preprint arXiv:2011.04406*, 2020.
[25] H. Kahn and A. W. Marshall, "Methods of reducing sample size in monte carlo computations," *Journal of the Operations Research Society of America*, vol. 1, no. 5, pp. 263–278, 1953.
[26] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *ICML*, 2018.
[27] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.
[28] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, 2000.
[29] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011.
[30] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *TPAMI*, 2018.
[31] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *NeurIPS*, 2010.
[32] D. l. T. Fernando and J. B. Mkchael, "A framework for robust subspace learning," *IJCV*, 2003.
[33] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann, "Easy samples first: Self-paced reranking for zero-example multimedia search," in *ACM MM*, 2014.
[34] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann, "Self-paced learning with diversity," in *NeurIPS*, 2014.
[35] Y. Wang, A. Kucukelbir, and D. M. Blei, "Robust probabilistic modeling with bayesian data reweighting," in *ICML*, 2017.
[36] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, 2001.
[37] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *ICML*, 2018.
[38] P. Chen, J. Ye, G. Chen, J. Zhao, and P.-A. Heng, "Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise," in *AAAI*, 2021.
[39] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *TPAMI*, 2021.
[40] J. Baxter, "A model of inductive bias learning," *Journal of artificial intelligence research*, vol. 12, pp. 149–198, 2000.
[41] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.
[42] G. Denevi, M. Pontil, and C. Ciliberto, "The advantage of conditional meta-learning for biased regularization and fine tuning," in *NeurIPS*, 2020.
[43] J. Shu, D. Meng, and Z. Xu, "Learning an explicit hyperparameter prediction policy conditioned on tasks," *arXiv preprint arXiv:2107.02378*, 2021.
[44] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *CVPR*, 2015.
[45] X.-S. W. Y. Z. F. S. J. W. J. Z. H. T. S. Zeren Sun, Yazhou Yao, "Webly supervised fine-grained recognition: Benchmark datasets and an approach," in *CVPR*, 2021.
[46] R. Jin and Z. Ghahramani, "Learning with multiple labels," in *NeurIPS*, 2002.
[47] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," in *NeurIPS*, 2020.
[48] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," in *NeurIPS*, 2017.
[49] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
[50] Q. Dong, S. Gong, and X. Zhu, "Class rectification hard mining for imbalanced deep learning," in *ICCV*, 2017.
[51] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *ICML*, 2004.
[52] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, 2019.

[53] J. Shu, Q. Zhao, Z. Xu, and D. Meng, "Meta transition adaptation for robust deep learning with noisy labels," *arXiv preprint arXiv:2006.05697*, 2020.
[54] M. Dehghani, A. Mehrjou, S. Gouws, J. Kamps, and B. Schölkopf, "Fidelity-weighted learning," in *ICLR*, 2018.
[55] Y. Fan, F. Tian, T. Qin, X.-Y. Li, and T.-Y. Liu, "Learning to teach," in *ICLR*, 2018.
[56] Y.-X. Wang, D. Ramanan, and M. Hebert, "Learning to model the tail," in *NeurIPS*, 2017.
[57] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie, "Large scale fine-grained categorization and domain-specific transfer learning," in *CVPR*, 2018.
[58] R. Wang, K. Hu, Y. Zhu, J. Shu, Q. Zhao, and D. Meng, "Meta feature modulator for long-tailed recognition," *arXiv preprint arXiv:2008.03428*, 2020.
[59] C. Huang, Y. Li, C. Change Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *CVPR*, 2016.
[60] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, "Range loss for deep face recognition with long-tailed training data," in *ICCV*, 2017.
[61] M. A. Jamal, M. Brown, M.-H. Yang, L. Wang, and B. Gong, "Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective," in *CVPR*, 2020.
[62] L. Huang, C. Zhang, and H. Zhang, "Self-adaptive training: beyond empirical risk minimization," in *NeurIPS*, 2020.
[63] S. Zheng, P. Wu, A. Goswami, M. Goswami, D. Metaxas, and C. Chen, "Error-bounded correction of noisy labels," in *ICML*, 2020.
[64] Y. Wu, J. Shu, Q. Xie, Q. Zhao, and D. Meng, "Learning to purify noisy labels via meta soft label corrector," in *AAAI*, 2021.
[65] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *AAAI*, 2017.
[66] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *ICCV*, 2019.
[67] E. Amid, M. K. Warmuth, R. Anil, and T. Koren, "Robust bi-tempered logistic loss based on bregman divergences," in *NeurIPS*, 2019.
[68] X. Ma, H. Huang, Y. Wang, S. Romano, S. Erfani, and J. Bailey, "Normalized loss functions for deep learning with noisy labels," in *ICML*, 2020.
[69] J. Shu, Q. Zhao, K. Chen, Z. Xu, and D. Meng, "Learning adaptive loss for robust learning with noisy labels," *arXiv preprint arXiv:2002.06482*, 2020.
[70] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *CVPR*, 2017.
[71] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," in *NeurIPS*, 2018.
[72] M. Lukasik, S. Bhojanapalli, A. Menon, and S. Kumar, "Does label smoothing mitigate label noise?" in *ICML*, 2020.
[73] C. Bishop, "Pattern recognition and machine learning," *Pattern Recognition and Machine Learning*, 2006.
[74] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.
[75] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019.
[76] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv preprint arXiv:1803.02999*, 2018.
[77] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *ICLR workshop*, 2015.
[78] H. Song, M. Kim, and J.-G. Lee, "Selfie: Refurbishing unclean samples for robust deep learning," in *ICML*, 2019, pp. 5907–5915.
[79] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda, "Early-learning regularization prevents memorization of noisy labels," in *NeurIPS*, 2020.
[80] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *ICLR*, 2017.
[81] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," in *NeurIPS*, 2019.
[82] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.
[83] Y. Zhang, S. Zheng, P. Wu, M. Goswami, and C. Chen, "Learning with feature-dependent label noise: A progressive approach," in *ICLR*, 2021.
[84] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR*, 2018.
[85] H.-S. Chang, E. Learned-Miller, and A. McCallum, "Active bias: Training more accurate neural networks by emphasizing high variance samples," in *NeurIPS*, 2017.
[86] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: robust training deep neural networks with extremely noisy labels," in *NeurIPS*, 2018.
[87] P. Chen, B. B. Liao, G. Chen, and S. Zhang, "Understanding and utilizing deep neural networks trained with noisy labels," in *ICLR*, 2019.
[88] E. Zheltonozhskii, C. Baskin, A. Mendelson, A. M. Bronstein, and O. Litany, "Contrast to divide: self-supervised pre-training for learning with noisy labels," in *CVPR*, 2021.
[89] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
[90] E. Malach and S. Shalev-Shwartz, "Decoupling" when to update" from" how to update"," in *NeurIPS*, 2017.
[91] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *CVPR*, 2016.
[92] K. Cao, Y. Chen, J. Lu, N. Arechiga, A. Gaidon, and T. Ma, "Heteroskedastic and imbalanced deep learning with adaptive regularization," in *ICLR*, 2021.
[93] L. Jiang, D. Huang, M. Liu, and W. Yang, "Beyond synthetic noise: Deep learning on controlled noisy labels," in *ICML*, 2020.
[94] J. Lv, M. Xu, L. Feng, G. Niu, X. Geng, and M. Sugiyama, "Progressive identification of true labels for partial-label learning," in *ICML*, 2020.
[95] Z. Cai, A. Ravichandran, S. Maji, C. Fowlkes, Z. Tu, and S. Soatto, "Exponential moving average normalization for self-supervised and semi-supervised learning," in *CVPR*, 2021.
[96] X. Yang, Z. Song, I. King, and Z. Xu, "A survey on deep semi-supervised learning," *arXiv preprint arXiv:2103.00550*, 2021.
[97] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," in *NeurIPS*, 2020.
[98] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *TPAMI*, 2018.
[99] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *ICML Workshop : Challenges in Representation Learning*, 2013.
[100] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *NeurIPS*, 2019.
[101] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE TPAMI*, 2015.
[102] S. R. He, Xiangyu Zhang and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016.
[103] F. H. Loshchilov, "Sgdr: Stochastic gradient descent with warm restarts," in *ICLR*, 2017.
[104] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia, "Iterative learning with open-set noisy labels," in *CVPR*, 2018.
[105] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMCV*, 2016.
[106] A. Z. Simonyan, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.
[107] S. M. Lin, Aruni RoyChowdhury, "Bilinear cnn models for fine-grained visual recognition," in *ICCV*, 2015.
[108] V. V. Szegedy, Sergey Ioffe and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI*, 2017.

[109] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[110] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[111] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *arXiv preprint arXiv:1812.01718*, 2018.
[112] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *CVPR Workshops*, 2020.
[113] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," *NeurIPS*, 2015.
[114] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NeurIPS*, 2017.
[115] Y. Geifman and R. El-Yaniv, "Selectivenet: A deep neural network with an integrated reject option," in *ICML*, 2019.
[116] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
[117] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.