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 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 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. 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. Extensive experiments substantiate the capability of our method on achieving proper weighting schemes in various data bias cases, like class imbalance, feature-independent and dependent label noises, 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. 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—Class imbalance, meta learning, noisy labels, partial-label learning, sample re-weighting, semi-supervised learning.

Manuscript received 31 January 2022; revised 14 December 2022; accepted 14 April 2023. Date of publication 1 May 2023; date of current version 5 September 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2022ZD0112900, in part by the National Natural Science Foundation of China (NSFC) under Grant 61721002, and in part by the Macao Science and Technology Development Fund under Grant 061/2020/A2. Recommended for acceptance by T. Xiang. (Corresponding author: Deyu Meng.)

Jun Shu and Xiang Yuan are with the School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi’an Jiaotong University, Xi’an, Shaanxi 710049, China (e-mail: xjtushujun@gmail.com; relojeffrey@gmail.com).

Deyu Meng is with the School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi’an Jiaotong University, Xi’an, Shaanxi 710049, China also with Pazhou Laboratory (Huangpu), Guangzhou, Guangdong 510555, China, and also with the Macau Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau 999078, China (e-mail: dyment@mail.xjtu.edu.cn).

Zongben Xu is with the School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi’an Jiaotong University, Xi’an, Shaanxi 710049, China, and also with Pazhou Laboratory (Huangpu), Guangzhou, Guangdong 510555, China (e-mail: zbxu@mail.xjtu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2023.3271451, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2023.3271451

I. 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].

A. Deficiencies of Sample Re-Weighting Approach

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 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 have 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.

B. Limitations and Meta-Essence Insight for MW-Net

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 dataset at hand. Experiments on datasets with class imbalance or noisy labels show that the automatically learned weighting schemes are consistent with the properly defined ones as traditional.

Nevertheless, MW-Net uses one unique weighting function shared by all classes of training dataset to deal with data bias, implying that different classes should possess consistent bias. For example, we plot the empirical probability density function (pdf) of training loss under the symmetric noise assumption (i.e., biases for each class are with almost equal possibility), as shown in Fig. 2(a). It can be seen that each class shares to an approximately homoscedastic training loss distribution. However, such homoscedastic bias assumption can not perfectly reflect the real complicated data bias scenarios. In fact, real-world biased datasets (like WebVision [14]) are often heterogeneous [38], i.e., biases are input-dependent, e.g., class-dependent or instance-dependent. Fig. 2(b) shows the training loss distribution under asymmetric (class-dependent) noise assumption. We can observe that losses of clean and noisy samples nearly overlap, and thus it is difficult to differentiate noisy samples from clean samples based on loss information. For such class-dependent bias case, MW-Net learns a monotonically increasing weighting function as shown in Fig. 3(c), which implies that MW-Net inclines to significantly lose efficacy. This naturally leads to significant performance degradation for MW-Net (see Table II). Considering real-world biased datasets always possess even more inter-class heterogeneous bias configurations than these simulated biased ones, it is thus fairly insufficient and improper to employ only single weighting function to deal with such complicated real-world 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 training classes/tasks.

The issue will be more prominent for practical large-scale datasets (e.g., WebVision [14]), especially for those containing a 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 class to obtain a better flexibility. This easy learning manner, however, is not only impractical due to its required

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. Illustration of the limitation and meta-essence understanding for MW-Net. The success of MW-Net is built upon homoscedastic bias assumption (e.g., in (a.1) and (a.2), each class has similar loss distributions of clean and noise samples). While MW-Net fails under the heterogeneous bias (e.g., in (b.1) and (b.2), each class has their specific loss distributions). The rationality can be revealed from the perspective of meta-learning (see Section I-B). The limitation of MW-Net demonstrates that only sample-level loss information can not sufficiently characterize the heterogeneous bias. This motivates us to introduce task-level information (i.e., scale of task) to reform MW-Net, making it able to distinguish individual bias properties of different tasks, and accumulate tasks with approximately homoscedastic data bias as a task family (see (a.3) and (b.3)). Please see more details in Sections I-C and III-C.

Fig. 3. (a)–(d) The weighting function extracted by MW-Net [9], and (e)–(h) three weighting functions extracted by CMW-Net (corresponding to three task families with small, moderate, large data scales), alongside the histogram of all sample weights calculated by them, for four types of simulated biased datasets. From left to right: Class imbalance (imbalance factor 10), Symmetric noise (noise rate 40%), Asymmetric noise (noise rate 40%), Feature-dependent noise (Type-I + 30% Asymmetric). The details of simulated biased datasets please refer to Section IV.
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, i.e., learning 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 features due to their mismatched input information to the meta-model.

C. CMW-Net and Our Contributions

Against the aforementioned issues, in this study, we substantially reform MW-Net to make it performable in practical scenarios with complicated data biases. Compared to that we use sample-level information (e.g., loss) to distinguish individual bias properties of different samples, the core idea is to extract certain higher task-level feature representation from all training classes/tasks to deliver their specific heterogeneous bias characteristics for discriminating training classes/tasks with similar data bias. And then we can accumulate tasks with approximately homoscedastic data bias (e.g., using a clustering algorithm according to the task feature) as a task family. Thus the training dataset can be divided into several task families, where intra-task-families own similar data bias, while inter-task-families own different data biases. To this aim, we simply take the scale level of each task (i.e., the number of samples for each class/task in our implementation) as the task feature, which can be validated to be effective and capable of assembling training classes with approximately homoscedastic loss distributions, as shown in Fig. 2. Then it is hopeful to deal with heterogeneous data bias by distinguishing individual bias properties of different classes/tasks, and adaptively ameliorating their imposed weighting function forms. Therefore, we can reform MW-Net by taking such task feature as the supplementary input information besides the sample loss into the weighting function, as shown in Fig. 1(b). We call this approach the Class-aware Meta-Weight-Net, or CMW-Net for brevity.

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

1) We propose a CMW-Net model, as shown in Fig. 1(b), to automatically learn a proper weighting strategy for real-world heterogeneous data bias in a meta-learning manner.

2) The proposed CMW-Net is model-agnostic, and is substantiated to be performable in different complicated data bias cases, and 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 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.

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 need to specifically extra tune hyperparameters of CMW-Net.

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. Section II discusses related work. Section III presents the proposed CMW-Net method as well as its learning algorithm and convergence analysis. Simulated and real-world experiments are demonstrated in Sections IV and V, respectively. Section VI evaluates the transferability of CMW-Net. Section VII introduces the evaluation of CMW-Net to several related applications. The conclusion is finally made.

II. 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 Section I, 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 datasets 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., triplet-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 issues, many methods have also been designed by making endeavors 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.

III. CLASS-AWARE META-WEIGHT-NET

A. Sample Re-Weighting Methodology

Consider a classification problem with biased training set \( D^r = \{x_i, y_i\}_{i=1}^N \), 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. \( f(x; w) \) denotes the classifier with \( w \) representing its model parameters. In current applications, \( f(x, w) \) 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 \( w^* \) can be extracted by minimizing the following training loss calculated on the training set:

\[
w^* = \arg\min_w \frac{1}{N} \sum_{i=1}^N \ell(f(x_i; w), y_i),
\]

where \( \ell(f(x; w), y) \) denotes the training loss on training sample \((x, y)\). In this study, we adopt the commonly adopted cross-entropy (CE) loss \( \ell(f(x; w), y) = -y^T \log(f(x; w)) \), where \( f(x; w) \) denotes the network output (especially, \( f(x; w) \in \Delta^c \) is a simplex when using Softmax function in the end layer of the network). For notation convenience, we denote \( L^r_i(w) = \ell(f(x_i; w), y_i) \) 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 \( w^* \) is calculated by minimizing the following weighted loss function:

\[
w^* = \arg\min_w \frac{1}{N} \sum_{i=1}^N v_i \ell(f(x_i; w), y_i).
\]

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.

B. 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 certain expert knowledge for the investigated problem. This naturally raises their difficulty in readily using them in real applications. Meanwhile, such weighting function pre-setting 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 \( \mathcal{V}(\ell; \theta) \), naturally succeeds from the previous sample re-weighting approaches, by setting its input as training loss and output as sample weight, with \( \theta \) as its network parameter. Just following standard MLP net, 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 \( D^{meta} = \{x^{(meta)}_i, y^{(meta)}_i\}_{i=1}^M \) (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 \( \theta^* \) of MW-Net can be obtained by minimizing the following bi-level optimization problem:

\[
\theta^* = \arg\min_{\theta} \frac{1}{M} \sum_{i=1}^M L^meta_i(w^*(\theta));
\]

\[
s.t. \ w^*(\theta) = \arg\min_w \frac{1}{N} \sum_{i=1}^N \mathcal{V}(L^r_i(w); \theta) L^r_i(w),
\]

where \( L^meta_i(w^*(\theta)) = \ell(f(x^{(meta)}_i; w^*(\theta)), y^{(meta)}_i) \). 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].

C. Class-Aware Meta-Weight-Net

In this section, we first show our motivation of constructing CMW-Net beyond MW-Net, and then we introduce the
fundamental consideration and principle of constructing the two branches contained in the CMW-Net architecture. Next we introduce how the two branches are practically formulated in our algorithm, and finally we summarize the overall formulation of CMW-Net and the bi-level optimization objective for calculating its final result.

According to the analysis in Section I-B, the main limitation of MW-Net is built on the premise that all training classes/tasks approximately follow a similar task distribution. Then MW-Net can use only one unique weighting scheme to handle homogeneous data biases over all training classes/tasks. However, real-world biased datasets are often heterogeneous with obvious inter-class variations of bias, especially for those with a large number of training classes. Since the premise is evidently hampered by heteroscedastic bias situations across different classes/tasks, MW-Net tends to largely lose its efficacy when encountering complicated biased datasets. This motivates us to reform MW-Net to make it possess adaptability of specifying proper weighting schemes to different classes/tasks based on their own internal bias characteristics.

To this aim, we propose a new weighting model as shown in Fig. 1(b), called Class-aware Meta-Weight-Net (CMW-Net for brevity). The architecture of CMW-Net is composed of two branches. The below branch integrates task-level feature knowledge into the original MW-Net as a beneficial compensation besides the original sample-level loss input. The function of this branch is to distinguish individual bias properties of different classes/tasks, and accumulate training classes/tasks with approximately homogeneous bias types (e.g., using a clustering algorithm according to the task feature) as a task family. Based on the identified task families, we can extract a possible faithful sample-weighting scheme shared by a task family, while suppressing the unexpected interference from other task families with heterogeneous ones. Such amelioration is expected to enable the output weight of a sample to correlate with its included training classes/tasks, so as to make it possibly adaptable to class bias variations.

In this study, we attempt to take the scale level (i.e., the number of samples) of each training class/task to represent its task feature. Albeit simple, such feature does be able to deliver helpful class/task pattern underlying its bias types. For instance, from Fig. 2, we can validate the effect for symmetric and asymmetric noise cases (please see more cases in supplementary material). 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 below branch of CMW-Net can be expressed as \( C(N_i; \Omega) \in \{0, 1\}^K \), by taking \( N_i \) as its input to represent the task feature, and including a hidden layer containing \( K \) nodes, attached with \( K \)-levels of scales \( \Omega = \{\mu_k\}_{k=1}^{K} \) sorted in 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., task family label), where 1-element is located at its \( k \)-th dimension corresponding to the nearest \( \mu_k \) to the input \( N_i \).

The other branch can be represented as \( V(L^{tr}_i(w); \Theta) \in [0, 1]^K \), 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 task families. The sharing hidden layer among these task families extracts the correlation among weighting principles of different task families, which helps reduce the risk of overfitting.

Then the CMW-Net weighting function is formulated as:

\[
V(L^{tr}_i(w), N_i; \Theta, \Omega) = V(L^{tr}_i(w); \Theta) \odot C(N_i; \Omega),
\]

where \( \odot \) denotes the dot product between two vectors. Through the modulation of the higher-level task feature information, CMW-Net is expected to learn a class-aware weighting function by accumulating training classes/tasks with homogeneous bias situations, and allow different training classes/tasks possessing different weighting schemes complying with their own internal bias characteristics.

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

\[
\{\Theta^*, \Omega^*\} = \arg\min_{\Theta, \Omega} \frac{1}{M} \sum_{i=1}^{M} L^{meta}_i(w^*(\Theta, \Omega)),
\]

\[
w^*(\Theta, \Omega) = \arg\min_w \sum_{i=1}^{N} V(L^{tr}_i(w), N_i; \Theta, \Omega)L^{tr}_i(w).
\]

Note that CMW-Net is degenerated to MW-Net if we take \( K = 1 \), i.e., all training classes own one weighting scheme.

\(^2\)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 task family located in the interval of \([0, 1]\).

Algorithm 1: The CMW-Net Meta-Training Algorithm.

\begin{enumerate}
\item Apply \( K \)-means on the sample numbers of all training classes to obtain \( \Omega = \{\mu_k\}_{k=1}^{K} \) sorted in ascending order.
\item Initialize classifier network parameter \( w^{(0)} \) and CMW-Net parameter \( \Theta^{(0)} \).
\item for \( t = 0 \) to \( T - 1 \) do
\item \{ \text{x, y} \} \leftarrow \text{SampleMiniBatch}(D^{tr}, n).
\item \{x^{meta}, y^{meta}\} \leftarrow \text{SampleMiniBatch}(D^{meta}, m).
\item Formulate the learning manner of classifier network \( \hat{w}^{(t+1)}(\Theta) \) by (7).
\item Update parameter \( \Theta^{(t+1)} \) of CMW-Net by (8).
\item Update parameter \( w^{(t+1)} \) of classifier by (9).
\end{enumerate}
**D. Learning Algorithm of CMW-Net**

1) **Meta-Training: Learning CMW-Net From Training Data:** We firstly discuss how to train the CMW-Net from the given training data. There are two groups of parameters, including $\Theta$ and $\Omega$, required to be optimized to attain the CMW-Net model. Therein, the optimization of the scale parameters $\Omega$ 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 pre-determine a rational specification of $\Omega^*$, 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 ascending order. Throughout all our experiments, we simply set $K = 3$. The small, moderate, and large-scale task families for different datasets can then be distinguished based on the ascending $\{\mu_k\}_{k=1}^K$. All our experiments show consistently and stably fine performance under such simple setting. This also implies that there remains a large room for further enhancement of our model by utilizing more elegant optimization techniques and designing more comprehensive task-level feature representations, which will be further investigated in our future research.

Then our aim is to solve the bi-level optimization of (5) and (6) to obtain optimal $\Theta^*$ and $w^*$. To make notation concise, we directly neglect $\Omega$ in (5) and (6) in the following. Note that exact solutions to (5) and (6) require solving the optimal $w^*$ whenever $\Theta$ 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 $w$ to online approximate the optimal classifier for a given $\Theta$, which guarantees the efficiency of the algorithm.

**Formulating Learning Manner of Classifier Network:** To optimize (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 $w(t)$ along the descent direction of (6) on the mini-batch training data as the following expression:

$$\hat{w}^{(t+1)}(\Theta) = w^{(t)} - \alpha \sum_{i=1}^n \mathcal{V}(L_{tr}^i(w^{(t)}), y_i, \Theta) \nabla_w L_{tr}^i(w) \big|_{w^{(t)}} \tag{7}$$

where $\alpha$ is the learning rate for the classifier network $f$.

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

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{m} \sum_{i=1}^m \nabla_\Theta L_{tr}^i(\hat{w}^{(t+1)}(\Theta)) \big|_{\Theta^{(t)}} \tag{8}$$

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

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

$$w^{(t+1)} = w^{(t)} - \alpha \sum_{i=1}^n \mathcal{V}(L_{tr}^i(w^{(t)}), y_i, \Theta^{(t+1)}) \nabla_w L_{tr}^i(w) \big|_{w^{(t)}} \tag{9}$$

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. 4 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 tend to be updated in a stable manner.

2) **Analysis on Intrinsic Learning Mechanism of CMW-Net:** We then present some insightful analysis for revealing some intrinsic learning mechanisms underlying CMW-Net. The updating step of (8) can be equivalently rewritten as (derivations are presented in supplementary material, available online):

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

where $G_{ij} = \frac{\partial L_{tr}^i(w)}{\partial w} \big|_{w^{(t)}} \frac{\partial L_{tr}^j(\hat{w}^{(t+1)}(\Theta))}{\partial w} \big|_{w^{(t)}}$. 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 ascending gradient of the weight function $\mathcal{V}(L_{tr}^i(w^{(t)}), y_i, \Theta)$. 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) **Meta-Test: Transferring CMW-Net to Unseen Tasks:** After the meta-training stage, the learned CMW-Net with parameter $\Theta^{(s)}$ can then be transferred to readily assign proper sample weights on unseen biased datasets. Specifically, for a query dataset $D^q = \{(x_i^q, y_i^q)\}_{i=1}^q$, 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 new scale parameters of CMW-Net. Then the learned CMW-Net can be directly used to impose sample weights to the classifier learning of the problem by solving the following objective of query task:

\[u^* = \arg\min_u \sum_{i=1}^{N^q} \mathcal{V}(L^q_u(u), N^q, \Theta, \Omega^q)L^q_u(u), \tag{11}\]

where \(L^q_u(u) = \ell(f(x_i; u), y_i^q)\), and \(N^q\) denotes the number of samples contained in the class to which \(x_i^q\) belongs. Then we can solve (11) with the learned \(\Theta^*\) to obtain classifier \(u^*\). The overall algorithm is summarized in Algorithm 2.

### E. Convergence of the CMW-Net Learning Algorithm

Next we attempt to establish a convergence result of our method for calculating (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 (5) and training loss (6) under some mild conditions in Theorems 1 and 2, respectively. The proofs are presented in the supplementary material, available online, (SM for brevity).

**Theorem 1:** Suppose the loss function \(\ell \) is Lipschitz smooth with constant \(L\), and CMW-Net \(\mathcal{V}(\cdot, \cdot; \Theta)\) is differential with \(\delta\)-bounded gradient and twice differential with its Hessian bounded by \(\mathcal{B}\), and the loss function \(\ell\) has \(\rho\)-bounded gradients with respect to training/meta data. Let the learning rate \(\alpha, \beta, 1 \leq t \leq T\) be monotonically decreasing sequences, and satisfy 
\[\alpha_t = \min\left\{\frac{1}{t^2}, \frac{\epsilon_1}{\sqrt{t}}\right\}, \beta_t = \min\left\{\frac{1}{t^2}, \frac{\epsilon_2}{\sqrt{t}}\right\},\]

for some \(\epsilon_1, \epsilon_2 > 0\), such that \(\frac{\epsilon_1}{c_1} \geq L, \frac{\epsilon_2}{c_2} \geq \delta\). Meanwhile, they satisfy 
\[\sum_{t=1}^{\infty} \alpha_t = \infty, \sum_{t=1}^{\infty} \alpha_2^{1/2} < \infty, \sum_{t=1}^{\infty} \beta_t = \infty, \sum_{t=1}^{\infty} \beta_t^2 < \infty.\]

Then CMW-Net can then achieve 
\[\mathbb{E}\left[\left\|\nabla L^\text{meta}(\hat{w}^t(\Theta^t))\right\|_2^2\right] \leq \epsilon \quad \text{in} \quad \mathcal{O}(1/e^2) \quad \text{steps. More specifically,}\]

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

where \(C\) is some constant independent of the convergence process.

**Theorem 2:** Under the conditions of Theorem 1, CMW-Net can achieve 
\[\mathbb{E}\left[\left\|\nabla L^\text{tr}(w^t(\Theta^t))\right\|_2^2\right] \leq \epsilon \quad \text{in} \quad \mathcal{O}(1/e^2) \quad \text{steps,}\]

where 
\[L^\text{tr}(w; \Theta) = \sum_{i=1}^{N} \mathcal{V}(L^t(x_i; w), N_i; \Theta)\]

More specifically,

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

where \(C\) is some constant independent of the convergence process.

### F. 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 regretfully 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 \(\ell\) 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.,

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

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

\[\ell_S(f(x; w), y) = -(v y + (1 - v)z)^T \log(f(x; w)). \tag{13}\]

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].

---

**Algorithm 2: The CMW-Net Meta-Test Algorithm.**

**Input:** Training dataset \(\mathcal{D}^t\), batch size \(n'\), max iterations \(T'\) and meta-learned CMW-Net with parameter \(\Theta^*\).

**Output:** Classifier parameter \(\hat{u}^*\).

1. Apply \(K\)-means on sample numbers of all training classes to obtain \(\Omega\) = \(\{\mu_k\}_{k=1}^K\) with the learned \(\hat{u}\).
2. Initialize classifier parameter \(u^{(0)}\).
3. **for** \(t = 0 \text{ to } T' - 1 \** do
4. **end for**
5. **Update classifier** \(u^{(t+1)}\) by solving (11).

---

![Main flowchart of the proposed CMW-Net meta-training algorithm (steps 6-8 in Algorithm 1).](image-url)
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 individual weights for different samples. Typically, SELFIE [78] iteratively selects clean samples by assigning weights \( v = 1 \) on them, and neglects doubtful noisy samples by setting their weights as \( v = 0 \) in (13). M-correction [7] ameliorates this hard weighting manner as soft, by fitting a two-component Beta 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 two-component Gaussian mixture model to assign a soft weight \( v \) for the corresponding sample.

However, all above methods require exploiting 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 (6) as (called CMW-Net-SL):

\[
\mathbf{w}^*(\Theta) = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} [\ell(\mathbf{L}_{i}^{tr}(\mathbf{w}), N_{i}; \Theta) L_{i}^{tr}(\mathbf{w}) + (1 - \mathbf{L}_{i}^{tr}(\mathbf{w}), N_{i}; \Theta)] L_{i}^{pre}(\mathbf{w})],
\]

(14)

where \( L_{i}^{pre}(\mathbf{w}) = \ell(f(x_i; \mathbf{w}), z_i) \). Taking a similar process as Section III-D2, we have

\[
\Theta(t+1) = \Theta(t) + \alpha \beta \sum_{j=1}^{n} \left[ \frac{1}{m} \sum_{i=1}^{m} (G_{ij} - G_{ij}^{t}) \right] \frac{\partial \mathbf{L}_{j}^{tr}(\mathbf{w}(t)), N_{j}; \Theta)}{\partial \Theta} |_{\Theta(t)}
\]

(15)

where \( G_{ij}^{t} = \frac{\partial L_{i}^{pre}(\mathbf{w})}{\partial \mathbf{w}} |_{\Theta(t)} \). Compared with CMW-Net, it is seen that CMW-Net-SL produces another term \( G_{ij}^{t} \) to control the learning of the meta-learner. Specifically, if \( \frac{1}{m} \sum_{i=1}^{m} (G_{ij} - G_{ij}^{t}) > 0 \), it means that the similarity between learning gradients 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.

In our experiments, we apply EMA [80] and temporal ensembling [81] techniques to produce pseudo-labels in our CMW-Net-SL algorithm, which has been verified to be effective in tasks like semi-supervised learning [81], and robust learning [7], [79]. 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.

IV. LEARNING WITH SYNTHETIC BIASED DATA

A. 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 [84], but are created by reducing the number of training samples per class according to an exponential function \( n_i = n_i \mu^i \), 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 pre-defined sample re-weighting techniques; 4) LDAM loss [82]: 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. 8) SADE [83]: leverage self-supervision to aggregate the learned multiple experts for achieving SOTA performance. More implementation details are specified in SM.

Results: Table I 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. 5 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 LDAM loss already has the capacity of mitigating the long-tailed issue by penalizing hard examples, our

\(^3\)The confusion matrix is calculated by applying the trained classifier to the corresponding testing set included with the CIFAR-10 dataset.
The comparison methods include: 1) ERM; 2) Focal loss [30]; 3) CB loss [4]; 4) LDAM loss [82]*; 1.2RW [26]; MW-Net [9]; 5) DivideMix; 6) L2RW [26] and 7) MW-Net [9]: represents the sample re-weighting methods by meta-learning. More experimental details are listed in SM.

Establishing Meta Dataset: Motivated by curriculum learning [31], [90], [91], we select the most confident training samples as meta data. 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 [92] 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 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 II 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(a) and thus CMW-Net learns three similar weighting schemes as that extracted by MW-Net, as shown in Fig. 3(f). 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 task families. Specifically, as shown in Fig. 3(g), weighting functions of small and moderate-scale task families tend to 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 a wider range of biased issues, it should be rational to say that CMW-Net is effective. 5) SADE uses multiple expertise-guided losses to produce competitive results, and the performance can be further boosted via introducing CMW-Net, demonstrating the effectiveness of our general weighting strategy.

To understand the weighing scheme learned by CMW-Net, we also depict the weighting functions learned by the CMW-Net in Fig. 3(e). It is seen that compared with MW-Net shown in Fig. 3(a), CMW-Net produces three weighting functions corresponding to small, moderate and large-scale task families. The overall tendency complies with conventional empirical setting for such class-wise weight functions, like CB loss [4] and MCW [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.

### Table I

| 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 | 56.19 | 48.86 | 44.29 | 29.50 |
| Focal loss [30] | 34.71 | 29.52 | 23.29 | 17.24 | 13.54 | 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 [82]* | - | 26.65 | - | - | 13.04 | - | 60.40 | - | - | - | - | - |
| 1.2RW [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 | 24.35 | 20.90 | 15.35 | 12.45 | 7.19 | 63.38 | 58.39 | 54.34 | 47.96 | 41.09 | 29.90 |
| MCW [61] with CE loss* | 29.34 | 23.59 | 19.49 | 13.34 | 11.15 | 7.21 | 60.69 | 56.05 | 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 | 13.63 | 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 |
| SAD [83] | 19.37 | 16.78 | 14.81 | 11.78 | 9.88 | 7.72 | 54.78 | 50.20 | 46.12 | 40.06 | 36.40 | 28.08 |
| CMW-Net with SAD | 19.11 | 16.04 | 13.54 | 10.25 | 9.39 | 5.39 | 54.59 | 49.50 | 46.01 | 39.42 | 34.78 | 27.50 |
TABLE II

| Datasets   | Noise                     | Symmetric Noise | Asymmetric Noise |
|------------|---------------------------|-----------------|-----------------|
|            | 0.2                       | 0.5             | 0.8             |
| CIFAR-10   | ERM                       | 86.89 ± 0.12    | 77.52 ± 0.41    | 73.63 ± 0.65    | 73.63 ± 0.65    | 33.82 ± 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 ± 0.36    | 74.96 ± 0.65    | 74.96 ± 0.65    | 54.64 ± 0.44    | 91.34 ± 0.28    | 89.67 ± 0.61    | 87.24 ± 0.96    | 61.07 ± 1.92    |
|            | GCE [6]                   | 89.99 ± 0.16    | 87.31 ± 0.53    | 82.15 ± 0.47    | 82.15 ± 0.47    | 57.36 ± 2.08    | 89.75 ± 1.53    | 87.75 ± 0.36    | 87.21 ± 3.64    | 57.46 ± 0.31    |
|            | M-correction [7]          | 93.80 ± 0.23    | 92.53 ± 0.11    | 90.30 ± 0.34    | 90.30 ± 0.34    | 68.60 ± 0.11    | 92.15 ± 0.16    | 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    | 94.23 ± 0.23    | 69.90 ± 0.23    | 93.96 ± 0.11    | 91.80 ± 0.78    | 80.14 ± 0.45    | 59.23 ± 0.38    |
|            | 1LRW [26]                 | 89.45 ± 0.62    | 87.18 ± 0.84    | 81.57 ± 0.66    | 81.57 ± 0.66    | 58.59 ± 1.84    | 90.46 ± 0.56    | 86.22 ± 0.71    | 85.17 ± 0.31    |
|            | MW-Net [9]                | 90.46 ± 0.52    | 86.53 ± 0.57    | 82.98 ± 0.34    | 82.98 ± 0.34    | 64.41 ± 0.92    | 92.69 ± 0.24    | 90.17 ± 0.11    | 86.55 ± 0.76    | 58.29 ± 1.33    |
|            | CMW-Net                   | 91.09 ± 0.54    | 86.91 ± 0.37    | 83.33 ± 0.55    | 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                | 92.60 ± 0.39    | 92.29 ± 0.14    | 94.51 ± 0.32    | 94.51 ± 0.32    | 64.20 ± 0.76    | 95.48 ± 0.29    | 94.51 ± 0.32    | 94.18 ± 0.21    | 93.97 ± 0.24    |

More emphasize those informative marginal samples since they barely contain replaced labels from other classes. While for large-scale task family, with many corrupted labels, CMW-Net tends to impose smaller weights on samples with relatively large losses to suppress the effect of these noisy labels. This shows that learned weighting schemes by CMW-Net can adapt 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 II. 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. 6 shows the confusion matrices obtained by CMW-Net and CMW-Net-SL in 80% label noise rate case. It is seen that CMW-Net-SL evidently improves the prediction accuracy, especially for classes with heavily 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. Different from prior works only reported results with a ratio of 40% for asymmetric noise, we consider more noise ratio settings to evidently show this phenomenon. It can be seen that DivideMix has a substantial degradation for higher noise ratio. 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(b), 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 a 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.

TABLE III

| Datasets   | Noise                     | Symmetric Noise | Asymmetric Noise |
|------------|---------------------------|-----------------|-----------------|
|            | 0.2                       | 0.5             | 0.8             |
| CIFAR-10   | DivideMix [8]             | 95.7 ± 0.34     | 92.9 ± 0.75     | 95.4 ± 0.75     |
|            | ELM [79]                  | 95.6 ± 0.34     | 93.9 ± 0.75     | 93.2 ± 0.75     |
|            | REFD [85]                 | 95.7 ± 0.45     | 94.1 ± 0.95     | 94.3 ± 0.95     |
|            | AugDrec [84]              | 95.6 ± 0.56     | 95.1 ± 0.91     | 94.5 ± 0.91     |
|            | C2D [87]                  | 96.2 ± 0.51     | 94.9 ± 0.73     | 94.7 ± 0.73     |
|            | Two-step [88]             | 95.6 ± 0.51     | 94.9 ± 0.73     | 94.7 ± 0.73     |
|            | CMW-Net                   | 95.7 ± 0.51     | 94.9 ± 0.73     | 94.7 ± 0.73     |
|            | CMW-Net-SL                | 95.6 ± 0.51     | 94.9 ± 0.73     | 94.7 ± 0.73     |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
MixMatch algorithms against noisy labels. Even though, benefiting from current SOTA methods like REED [85], C2D [87], AugDesc [86], Two-step [88], which additionally use two general tricks of self-supervised learning for performance improvement, i.e., adding a warm-up self-supervised pre-training step and imposing a data augmentation based consistency regularization as commonly used in semi-supervised algorithms, we also easily borrow these common tricks to boost our method (denoted by CMW-Net-SL+). It can be easily seen that the ameliorated CMW-Net-SL+ consistently outperforms the compared SOTA methods, and beats them on CIFAR-100 with 80% and 90% symmetric noise by an evident margin.

C. 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 [89], [95].

Datasets: We follow the PMD noise generation scheme proposed in [89]. 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 \( u_x = \arg\max_s \eta(x) \) of each training sample \( x \) to its second confident category \( s_x \), with conditional probability \( \tau_{u_x,s_x} = P(\tilde{y} = u_x | y = s_x, x) \). We use three types of \( \tau_{u_x,s_x} \) designed in [89] as follows:

- **Type-I**: \( \tau_{u_x,s_x} = -\frac{1}{2} [\eta(u_x|x) - \eta(s_x|x)]^2 + \frac{1}{2} \),
- **Type-II**: \( \tau_{u_x,s_x} = 1 - [\eta(u_x|x) - \eta(s_x|x)]^3 \),
- **Type-III**: \( \tau_{u_x,s_x} = 1 - \frac{1}{3} \left[ [\eta(u_x|x) - \eta(s_x|x)]^3 + [\eta(u_x|x) - \eta(s_x|x)]^3 + [\eta(u_x|x) - \eta(s_x|x)]^3 \right] \).

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

Baselines: Following the benchmark in [89], we compare the following baselines: 1) ERM; 2) LRT [63]; 3) GCE [6]; 4) MW-Net [9] and 5) PLC [89], 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 IV 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 V further shows the results on datasets corrupted with a combination of feature dependent and independent noises, where feature-independent noise is overlaid 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 [89], which are harder to distinguish and more likely to be mislabeled. From Fig. 3(b), it can be seen that the proposed method can still finely distinguish most 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. 3(d), 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 the the PLC method [89], 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 a 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.

V. LEARNING WITH REAL BIASED DATA

A. 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], [96], 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

### Table IV

| Datasets | Noise | ERM | LRT [63] | GCE [6] | MW-Net [9] | PLC [89] | CMW-Net | CMW-Net-SL |
|----------|-------|-----|---------|--------|-----------|----------|---------|-----------|
| CIFAR-10 | Type-I (35%) | 78.11 ± 0.74 | 80.98 ± 0.80 | 80.64 ± 0.39 | 82.20 ± 0.40 | 82.80 ± 0.27 | 82.27 ± 0.33 | 84.23 ± 0.17 |
| CIFAR-10 | 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 |
| CIFAR-100 | Type-I (35%) | 57.08 ± 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 |
| CIFAR-100 | 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 |
| CIFAR-100 | 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 |
| CIFAR-100 | 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 |
| CIFAR-100 | 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 |
| CIFAR-100 | Type-III (70%) | 40.01 ± 0.18 | 44.41 ± 0.19 | 40.53 ± 0.60 | 45.17 ± 0.77 | 44.58 ± 0.62 | 47.38 ± 0.65 | 48.78 ± 0.62 |
TABLE V

| Datasets | Noise | ERM [63] | LRT [63] | GCE [6] | MW-Net [9] | PLC [89] | 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 | 78.12 ± 0.47 | 82.00 ± 0.63 |
| CIFAR-10 | Type-I - Asymmetric | 75.21 ± 0.64 | 76.96 ± 0.35 | 76.91 ± 0.56 | 76.54 ± 0.54 | 78.31 ± 0.41 | 77.14 ± 0.38 | 80.69 ± 0.47 |
| CIFAR-100 | Type-II - Symmetric | 74.92 ± 0.63 | 75.94 ± 0.58 | 75.69 ± 0.21 | 75.35 ± 0.40 | 77.63 ± 0.30 | 77.08 ± 0.52 | 80.94 ± 0.14 |
| CIFAR-100 | Type-II - Asymmetric | 74.28 ± 0.39 | 77.03 ± 0.62 | 75.30 ± 0.81 | 75.95 ± 0.61 | 77.26 ± 0.82 | 79.06 ± 0.47 | 81.34 ± 0.55 |
| CIFAR-100 | Type-III - Symmetric | 73.00 ± 0.38 | 76.66 ± 0.57 | 77.00 ± 0.12 | 76.28 ± 0.82 | 78.06 ± 0.47 | 79.14 ± 0.30 | 80.68 ± 0.48 |
| CIFAR-100 | Type-III - Asymmetric | 73.31 ± 0.34 | 77.19 ± 0.74 | 75.70 ± 0.91 | 75.82 ± 0.77 | 77.54 ± 0.70 | 75.49 ± 0.88 | 80.68 ± 0.48 |

TABLE VI

| Method | Test Accuracy | Method | Test Accuracy |
|--------|---------------|--------|---------------|
| ERM    | 79.4 ± 0.13   | ActiveBias [93] | 80.5 ± 0.25   |
| Co-teaching [94] | 80.2 ± 0.13 | SELFIE [78] | 81.8 ± 0.09 |
| PLC   | 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 VII

| Method | WebVision | ILSVRC12 | WebVision | ILSVRC12 |
|--------|-----------|----------|-----------|----------|
| top1   | top5      | top1     | top5      |
| Forward* [70] | 61.12 | 82.68 | 75.36 | 82.36 |
| MentorNet* [37] | 63.00 | 81.40 | 75.80 | 79.92 |
| Co-teaching* [94] | 63.58 | 85.20 | 61.48 | 84.70 |
| Interative-C* [96] | 65.24 | 83.94 | 61.60 | 84.98 |
| MW-Net [9] | 69.34 | 87.44 | 65.80 | 87.92 |
| CMW-Net | 70.56 | 88.76 | 64.44 | 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 [87] | 79.42 | 93.37 | 78.37 | 93.04 |
| CMW-Net-SL-C2D | 80.44 | 93.56 | 77.36 | 93.48 |

B. 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.

Results: Tables VI and VII compare the test performance of all competing methods trained on the Animal-10 N and mini WebVision datasets, respectively. For the Animal-10 N dataset, we compare 4 methods that have reported performance on this dataset. Compared with sample selection methods ActiveBias [93] and Co-teaching [94], 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 [89]. Fig. 7(a) and (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. 8(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 VII 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 [87], we further boost the performance. In Fig. 7(c), we show some typical noisy examples corrected by the proposed method, showing its capability of recovering these easily confused samples.

WebVision [14] and the corresponding class samples of ImageNet [98]. ResNet-50 is adopted as the classifier network. More implementation details are specified in SM.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
VI. 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 Section IV-C since it finely simulates the real-world noise configuration. The extracted weighting function is depicted in Fig. 3(h). We deploy it on two large-scale real-world biased datasets, ImageNet-LT [5] and full WebVision [14].

Table IX 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, the performance on full WebVision is compared in Table X. It is interesting to see that by directly integrating the learned CMW-Net into the simple ERM algorithm with more training epochs, the performance can be further improved, outperforming most of these SOTA methods, only slightly inferior to the CurriculumNet method [16], whose results were obtained with ensemble of six models. Even with a relatively concise form, our method still outperforms the second-best Heteroscedastic method by an evident margin. This further validates the potential usefulness of CMW-Net to practical large-scale problems with
complicated data bias situations, with an intrinsic reduction of the labor and computation costs by readily specifying proper weighting scheme for a learning algorithm. More experimental details are presented in SM.

VII. 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.

A. Partial-Label Learning

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 $\mathcal{X} \in \mathbb{R}^d$ as the input space, $\mathcal{Y} := \{1, \ldots, C\}$ 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 $x$, 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 $Y$. The PLL risk estimator is then defined as:

$$R_{PLL}(f) = \mathbb{E}_{(x, y) \in \mathcal{D}}[\ell_{PLL}(f(x), Y)],$$

(16)

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

To estimate (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 [103] 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 $w_y$s consist of a one-hot vector, expected to reflect the confidence of the label $y \in Y$ being the true label.

2) CMW-Net Amelioration and Experiments: PRODEN [103] 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 Section IV-B) can be readily employed to further improve the classifier.

Following PRODEN [103], two sets of partial label datasets are generated from CIFAR-10 and CIFAR-100, respectively, under different flipping probabilities. As shown in Fig. 9, 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.

B. Semi-Supervised Learning

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 unlabeled dataset $D_U = \{(x_i)\}_{i=1}^u$, and $L \ll U$. Formally, SSL aims to solve the following optimization problem [105]:

$$\min_w \sum_{(x_i, y_i) \in D_L} \mathcal{L}_S(x_i, y_i; w) + \alpha \sum_{x_i \in D_U} \mathcal{L}_U(x_i; w),$$

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 [106] or a regularization term [107]. $w$ 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 [108], 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 $\mathcal{L}_S$). Recently, the SOTA SSL methods, like VAT [107], MixMatch [109], UDA [106], and FixMatch [47] makes good progress to enhance the pseudo labeling capability by using sample augmentation techniques, through encouraging consistency under different augmented data [106]. We take the recent SOTA Fixmatch [47] as a typical

| Methods       | WebVision top1 | WebVision top5 | ILSVRC12 top1 | ILSVRC12 top5 |
|---------------|---------------|---------------|--------------|--------------|
| ERM           | 69.7          | 67.0          | 62.9         | 83.6         |
| MentorNet [37]| 70.8          | 88.0          | 62.5         | 83.0         |
| MentorMix [100]| 74.3         | 90.5          | 67.5         | 87.2         |
| HAR [101]     | 75.0          | 90.6          | 67.1         | 86.7         |
| MIL [102]     | 76.5          | 90.9          | 68.7         | 86.4         |
| Heteroscedastic [38] | 76.6         | 92.1          | 68.6         | 87.1         |
| CurriculumNet [16] | 79.3         | 93.6          | -            | -            |
| ERM + CMW-Net-5L | 77.9         | 92.6          | 69.6         | 88.5         |
example. Denote $\alpha(x_i) \in \mathcal{A}(x_u)$ as augmentation operators imposed on labeled and unlabeled samples, respectively. Then the FixMatch model can be written as:

$$
\min_w \sum_{(x_i, y_l) \in D_L} \ell(f(\alpha(x_i); w), y_l) + \alpha \sum_{x_u \in D_U} 1(\max(z_u \geq \tau) \ell(f(\mathcal{A}(x_u); w), y_u), \quad (17)
$$

where $y_u$ is the pseudo label on $x_u$, calculated by $y_u = \arg \max_j z_{u,j}$; $z_u = f(\alpha(x_u); w)$ in iteration. $\tau$ is a scalar hyper-parameter denoting the threshold above which we retain a pseudo-label. Note that $1(\max(z_u \geq \tau)$ corresponds to a hard weighting scheme with manually specified hyperparameter $\tau$. 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 $\tau$ is often not easily and adaptably specified against different tasks. The method thus still has room for further performance enhancement.

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/tasks. Then the problem (17) can be ameliorated as:

$$
\min_w \sum_{(x_i, y_l) \in D_L} \ell(f(\alpha(x_i); w), y_l) + \alpha \sum_{(x_u) \in D_U} \mathcal{V}(L^r_u(w), N_i; \Theta) \ell(f(\mathcal{A}(x_u); w), y_u).
$$

The algorithm can also be readily designed by integrating the updating step for the meta-parameter $\Theta$ 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 $w$ in an automatic and more likely intelligent manner.

We conduct experiments on several standard SSL image classification benchmarks, including CIFAR-10, CIFAR-100 and ImageNet dataset [84]. Results are shown in Table XI. It is evident that our CMW-Net consistently helps improve the performance of FixMatch, showing its potential application prospects on this task. Especially, when FixMatch is trained with smaller labeled data resources, pseudo labels generated by FixMatch tend to be relatively unreliable, naturally resulting in performance degradation. CMW-Net is capable of adaptively reducing the negative effect of unreliable pseudo labels, and thus improves FixMatch significantly in this case. More experimental settings and results are presented in SM.

VIII. 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 and comprehensive statistical learning understanding for revealing its intrinsic generalization capability across different tasks [43]. Besides, we’ll try to build more wider range of connections with our method to previous techniques on exploring data insights, like importance weighting [110]. More sufficient and comprehensive task-level feature representation will also be further investigated in our future research. Further algorithm efficiency enhancement of our model will also be investigated in our future research.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Annu. Meeting Assoc. Comput. Linguistics: Hum. Lang. Technol., 2019, pp. 4171–4186.
[3] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” IEEE/ACM Trans. Audio Speech Lang. Process., vol. 22, no. 10, pp. 1533–1545, Oct. 2014.

[4] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, “Class-balanced loss based on effective number of samples,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 9260–9269.

[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 Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 2532–2541.

[6] Z. Zhang and M. R. Sabuncu, “Generalized cross entropy loss for training deep neural networks with noisy labels,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2018, pp. 8792–8802.

[7] E. Arazo, D. Ortego, P. Albert, N. O’Connor, and K. McGuinness, “Unsupervised label noise modeling and loss correction,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 312–321.

[8] J. Li, R. Socher, and S. C. Hoi, “DivideMix: Learning with noisy labels as semi-supervised learning,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–14.

[9] J. Shu et al., “Meta-weight-net: Learning an explicit mapping for sample weighting,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 1919–1930.

[10] W. Bi, L. Wang, J. T. Kwok, and Z. Tu, “Learning to predict from crowdsourced data,” in Proc. 30th Conf. Uncertainty Artif. Intell., 2014, pp. 82–91.

[11] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, “Do imagenet classifiers generalize to imagenet?,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 5389–5400.

[12] C. G. Northcutt, A. Athalye, and J. Mueller, “Do imagenet classifiers generalize to imagenet?,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 5389–5400.

[13] S. Yun, S. J. Oh, B. Heo, D. Han, J. Choe, and S. Chun, “Relabeling imageNet: From single to multi-labels, from global to localized labels,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 2340–2350.

[14] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, “Webvision database: Visual learning and understanding from web data,” 2017, arXiv:1708.02682.

[15] J. Shu, Z. Xu, and D. Meng, “Small sample learning in Big Data era,” 2018, arXiv:1808.04572.

[16] S. Guo et al., “CurriculumNet: Weakly supervised learning from large-scale web images,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 139–154.

[17] D. Arpit et al., “A closer look at memorization in deep networks,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 233–242.

[18] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” in Proc. Int. Conf. Mach. Learn. Representations, 2017, pp. 1–15.

[19] C. Zhang, B. Kang, B. Heo, S. Yan, and J. Feng, “Deep long-tailed learning: A survey,” 2021, arXiv:2110.04596.

[20] B. Frény and M. Verleysen, “Classification in the presence of label noise: A survey,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 23, no. 10, pp. 1533–1545, Oct. 2001.

[21] M. Kumar, B. Packer, and D. Koller, “Self-paced learning for labeled data,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 312–321.

[22] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[23] Z. Sun et al., “Weby supervised fine-grained recognition: Benchmark datasets and an approach,” in Proc. Int. Conf. Comput. Vis. Pattern Recognit., 2021, pp. 10585–10591.

[24] J. Shu, D. Meng, and Z. Xu, “Learning an explicit hyperparameter prediction policy conditioned on tasks,” 2021, arXiv:2107.02378.

[25] B. J. Zadrozny, “Learning and evaluating classifiers under sample selection bias,” in Proc. Int. Conf. Mach. Learn., 2004, pp. 1–9.

[26] B. J. Zadrozny, “Learning and evaluating classifiers under sample selection bias,” in Proc. Int. Conf. Mach. Learn., 2004, pp. 1–9.

[27] J. M. Johnson and T. M. Khoshgoftaar, “Survey on deep learning with noisy labels: Exploring techniques and remedies in medical image analysis,” Med. Image Anal., vol. 65, 2020, Art. no. 101759.

[28] J. M. Johnson and T. M. Khoshgoftaar, “Survey on deep learning with noisy labels: Exploring techniques and remedies in medical image analysis,” Med. Image Anal., vol. 65, 2020, Art. no. 101759.

[29] B. J. Zadrozny, “Learning and evaluating classifiers under sample selection bias,” in Proc. Int. Conf. Mach. Learn., 2004, pp. 1–9.

[30] J. M. Johnson and T. M. Khoshgoftaar, “Survey on deep learning with noisy labels: Exploring techniques and remedies in medical image analysis,” Med. Image Anal., vol. 65, 2020, Art. no. 101759.

[31] B. J. Zadrozny, “Learning and evaluating classifiers under sample selection bias,” in Proc. Int. Conf. Mach. Learn., 2004, pp. 1–9.
S. Laine and T. Aila, “Temporal ensembling for semi-supervised learning,” in Proc. Int. Conf. Learn. Representations, 2017, pp. 1–13.

H. Song, M. Kim, and J.-G. Lee, “SELFIE: Refurbishing unclean samples for robust deep learning,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 5907–5915.

S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda, “Early-learning regularization prevents memorization of noisy labels,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 20331–20342.

A. Tarvainen and M. Valpola, “Mentors are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 1195–1204.

S. Laine and T. Aila, “Temporal ensembling for semi-supervised learning,” in Proc. Int. Conf. Learn. Representations, 2017, pp. 1–13.

H. Song, M. Gaidon, N. Arbelaez, and T. Ma, “Learning imbalanced datasets with label-distribution-aware margin loss,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 1567–1578.
[108] D.-H. Lee et al., “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” in Proc. ICML Workshop: Challenges Representation Lear., 2013, pp. 1–6.

[109] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, “MixMatch: A holistic approach to semi-supervised learning,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 5049–5059.

[110] T. Liu and D. Tao, “Classification with noisy labels by importance reweighting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 3, pp. 447–461, Mar. 2016.

Jun Shu received the BE degree from Xi’an Jiaotong University, Xi’an, China, in 2016, where he is currently working toward the PhD degree, under the tuition of Prof. Deyu Meng and Prof. Zongben Xu. His current research interests include machine learning and computer vision, especially on meta learning, robust deep learning, and AutoML.

Xiang Yuan received the BE degree from Xi’an Jiaotong University, Xi’an, China, in 2020, where he is currently working toward the PhD degree, under the tuition of Prof. Deyu Meng and Prof. Zongben Xu. His current research interests include meta learning and robust deep learning.

Deyu Meng received the BSc, MSc, and PhD degrees from Xi’an Jiaotong University, Xi’an, China, in 2001, 2004, and 2008, respectively. He was a visiting scholar with Carnegie Mellon University, Pittsburgh, PA, USA, from 2012 to 2014. He is currently a professor with the School of Mathematics and Statistics, Xi’an Jiaotong University, and an adjunct professor with the Faculty of Information Technology, Macau University of Science and Technology, Taipa, Macau, China. His research interests include model-based deep learning, variational networks, and meta learning.

Zongben Xu received the PhD degree in mathematics from Xi’an Jiaotong University, Xi’an, China, in 1987. He currently serves as the Academician of the Chinese Academy of Sciences, the chief scientist with the National Basic Research Program of China (973 Project), and the director with the Institute for Information and System Sciences, Xi’an Jiaotong University. His current research interests include nonlinear functional analysis and intelligent information processing. He was a recipient of the National Natural Science Award of China, in 2007, and the winner of the CSIAM Su Buchin Applied Mathematics Prize, in 2008.