A Probabilistic Formulation for Meta-Weight-Net

Qian Zhao, Jun Shu, Xiang Yuan, Ziming Liu, and Deyu Meng, *Member, IEEE*

***Abstract*— In the last decade, deep neural networks (DNNs) have become dominant tools for various of supervised learning tasks, especially classification. However, it is demonstrated that they can easily overfit to training set biases, such as label noise and class imbalance. Example reweighting algorithms are simple and effective solutions against this issue, but most of them require manually specifying the weighting functions as well as additional hyperparameters. Recently, a meta-learning-based method Meta-Weight-Net (MW-Net) has been proposed to automatically learn the weighting function parameterized by an MLP via additional unbiased metadata, which significantly improves the robustness of prior arts. The method, however, is proposed in a deterministic manner, and short of intrinsic statistical support. In this work, we propose a probabilistic formulation for MW-Net, probabilistic MW-Net (PMW-Net) in short, which treats the weighting function in a probabilistic way, and can include the original MW-Net as a special case. By this probabilistic formulation, additional randomness is introduced while the flexibility of the weighting function can be further controlled during learning. Our experimental results on both synthetic and real datasets show that the proposed method improves the performance of the original MW-Net. Besides, the proposed PMW-Net can also be further extended to fully Bayesian models, to improve their robustness.**

***Index Terms*— Example reweighting, meta-learning, probabilistic model, robustness.**

I. Introduction

IN RECENT years, deep neural networks (DNNs) have been widely used in machine learning applications, and become dominant tools for various tasks, especially classification, due to their powerful capacity for modeling complex input patterns. However, it has been shown that DNNs can easily overfit to biased training set [1], [2], such as containing label noise [3] or with class imbalance [4], which leads to their poor generalization performance on test set. In practice, such data biases are always encountered. On the one hand, datasets collected from a crowdsourcing systems [5] or search engines [6], [7], for example, often contain a large number of noisy labels due to limited experiences of human workers or ambiguities of search engines. Even for the well-known and widely used ImageNet, the label noise issue is unavoidable [8]. On the other hand, class imbalance issue occurs in many real applications by the mechanism of data generation, such as learning-based object detection [9]–[11] and semantic segmentation [12]. These robust learning issues have also been theoretically investigated in previous work [13]–[17], and become an important while challenging research topic [1], [18].

Example reweighting is one of the most commonly used strategies to alleviate the aforementioned robust learning issue, due to its simplicity and effectiveness. The main idea is to design a weighting function that maps from the loss value of an example to its weight, and then alternate between calculating weights by loss values with current classifier and minimizing the weighted loss function for updating the classifier. There are mainly two kinds of methodologies, which are contradictory, for example reweighting. The first is to design the mapping as a monotonically decreasing function, which assigns larger weights to examples with smaller loss values and vice versa. This strategy is reasonable for datasets with noisy labels, since the clean examples are generally easier to learn with smaller loss values, and thus should be emphasized. Typical methods based on this consideration include self-paced learning (SPL) [19]–[21], iterative reweighting [22], [23].[1] In contrast, the other methodology assumes the weighting function monotonically increasing, which sets larger weights to examples with larger loss values and vice versa. This strategy enforces the learning process to emphasize hard examples, which are more likely to locate on the classification boundary, and is suitable for datasets with class imbalance since the numbers of boundary examples are more balanced for different classes. Representative methods belonging to this group include AdaBoost [24], [25], hard negative mining [10], and focal loss [11].

Although they can improve the robustness of learning algorithms on biased training datasets, these example reweighting

Manuscript received October 24, 2020; revised April 13, 2021 and May 25, 2021; accepted August 10, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0106300; in part by the National Natural Science Foundation of China (NSFC) project under Contract 62076196, Contract 11690011, Contract 61721002, and Contract U1811461; and in part by Macao Science and Technology Development Fund under Grant 061/2020/A2. *(Corresponding author: Deyu Meng.)*

Qian Zhao, Jun Shu, Xiang Yuan, and Ziming Liu are with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Key Laboratory for Intelligent Networks and Network Security, Ministry of Education, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: timmy.zhaoqian@xjtu.edu.cn; xjtushujun@gmail.com; relojeffrey@gmail.com; lzmlzm@stu.xjtu.edu.cn).

Deyu Meng is with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China, also with the Key Laboratory for Intelligent Networks and Network Security, Ministry of Education, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Faculty of Information Technology, Macau University of Science and Technology, Taipa, Macau, China (e-mail: dymeng@mail.xjtu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3105104.

Digital Object Identifier 10.1109/TNNLS.2021.3105104

[1]Note that, as reviewed in the next section, there are many other methodologies for dealing with such noisy label issue, while we focus on example reweighting due to its simplicity.

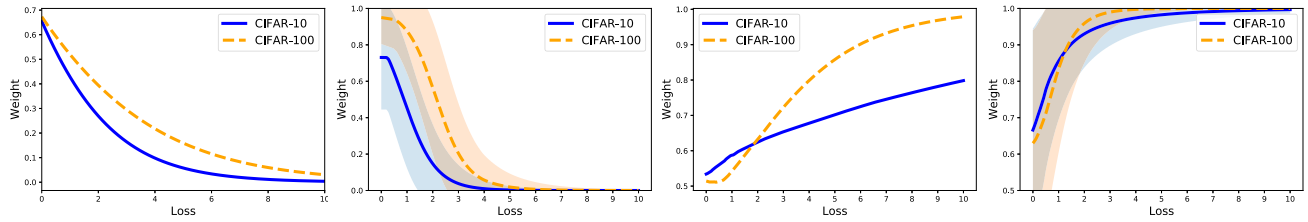


Fig. 1. (a) and (c) plot the weighting functions learned by MW-Net under label noise and class imbalance, respectively. (b) and (d) plot the weighting functions learned by PMW-Net under label noise and class imbalance, respectively, with the curve and shadow being the mean and variance, respectively. The label noise case is with 40% symmetric noise, and the class imbalance case is with imbalance factor 10.

methods suffer from the major limitation that the forms, as well as the corresponding hyperparameters, of the weighing function should be prespecified or tuned by cross-validation. In real applications, however, biases existing in observed data are generally complicated, possibly including both label noise and class imbalance [26], and thus subjectively setting the weighting function tends to decrease the performance of learning algorithms.

To alleviate this issue, focusing on the classification task, we proposed Meta-Weight-Net (MW-Net) framework in our earlier work [2]. Instead of a subjectively prespecified increasing or decreasing loss-to-weight function, this method formulates the function form as an MLP, which can well approximate a wide range of function expressions, and thus can generally include the previous contradictory ones as its special cases. For example, as can be seen from Fig. 1(a), when implemented on datasets with label noise, the learned weighting function is monotonically nonincreasing, which can produce larger weights to the clean examples with relatively smaller loss values, and thus emphasize them, similar to SPL. In contrast, when applied to datasets with class imbalance, as shown in Fig. 1(c), the learned weighting function is monotonically nondecreasing, which emphasize the hard examples with relatively larger loss values by assigning smaller weights to them, similar to AdaBoost. Albeit simple to implement, MW-Net can significantly improve the robustness of DNN classifiers against complex training set biases, and achieve superior performance compared with previous state-of-the-arts both specifically set for label noise and class imbalance.

The original MW-Net, however, is constructed solely in a deterministic manner, and thus is short of a probabilistic interpretation as well as the merits brought by probabilistic modeling, such as the uncertainty estimation. Besides, due to the deterministic nature, MW-Net cannot be directly applied to Bayesian machine learning problems. To this end, we deeply dig into the working mechanism of MW-Net and discover its probabilistic understanding in this article. Specifically, instead of the deterministic weighting function adopted in the original MW-Net, we reformulate the weighting function as a distribution whose parameters are determined by MLPs in an amortized fashion. The parameters of the MLPs are then learned by optimizing an approximated evidence lower bound (ELBO) with reparameterization gradient. By this probabilistic formulation, four advantages as compared with the deterministic MW-Net can be expected, all of which are substantiated by our experiments on a series of datasets with different complex data biases.

1) Formulating the output of weighting function as a distribution can directly provide the uncertainties about example weights, as illustrated in Fig. 1(c) and (d), which enhances the interpretability of the loss-to-weight weighting mechanism.
2) Due to the reparameterization gradient used for updating the parameters of the weighting function, extra randomness can be introduced. As is well known in machine learning community, randomness is often helpful for the overall performance of a learning algorithm [27]–[29].
3) By the probabilistic formulation, an extra KL-divergence term is naturally induced in the optimization objective, which acts as an additional regularization. This regularization facilitates a more stable training of the weighting function, and is also expected to contribute to further performance improvement.
4) By virtue of the probabilistic nature of the weighting function, the proposed framework can be easily extended to fully Bayesian modeling, to deal with the robustness issue in Bayesian machine learning methods.

The article is organized as follows. In Section II, we review some related studies on the concerned problem. In Section III, we presents our probabilistic formulation of MW-Net, i.e., probabilistic MW-Net (PMW-Net), and corresponding learning algorithm, with some implementation considerations. The effectiveness of our proposed method is empirically verified in Section IV on several synthetic and real-world datasets with complex data biases. We then discuss in Section V the extension of PMW-Net to fully Bayesian modeling, with example application of Bayesian logistic regression, and finally conclude the whole article in Section VI.

II. Related Work

A. Traditional Example Weighting Methods

The idea of example weighting has a long history in machine learning, and various methods have been proposed. A typical strategy is to compute the example weights as a pre processing step using certain prior knowledge about the task or data, such as dataset resampling [30]–[32] and instance reweighting [33]. A more reasonable data-driven strategy is to design a weighting function mapping the loss values of the

training examples to their weights, and automatically assign weights to examples during the training process [34], [35]. There are mainly two classes of weighting functions considered in existing methods. The first class includes weighting functions that are monotonically decreasing with respect to loss values, such as SPL [19], [20], [36], [37], and iterative reweighting [22], [23], which are expected to improve the robustness of learning algorithms against label noise. The other class consists of weighting functions that are monotonically increasing with loss values, including AdaBoost [24], [25] and its variants [38], hard negative mining [10], and focal loss [11]. These methods, however, require manually specifying the forms of weighting functions beforehand, as well as the included hyperparameters, which is inconvenience for real applications.

B. Meta-Learning-Based Example Reweighting

Motivated by recent advances in meta-learning [39]–[41], some methods were proposed to automatically learn an adaptive weighting scheme with the aid of additional meta data. For example, FWL [42] designed the weighting function as a Bayesian function approximator, learning-to-teach method [43], [44] adopted the DNN with attention mechanism, and MentorNet [18] used a bidirectional LSTM. Instead of only taking the loss values as inputs, these methods considered more information of the training data. However, with much more complex forms of weighting functions, as well as input features, these methods lose the simplicity of the traditional reweighting methods, and thus are relatively difficult to implement and reproduce by general users.

Inspired by MAML [39], L2RW [1] method was proposed to directly learn the example weights without complicated parameterization. In specific, it learns the weights by approximately solving a bi-level optimization problem, which aims at minimizing the prediction loss evaluated on meta data. Very recently, Shu *et al.* [2] generalized this idea by explicitly define the loss-to-weight mapping as a simple MLP, which on one hand provides a more stable weighting behavior during training, and on the other hand enables the learned weighting function to be generalized from a certain task to related other ones. A dominant superiority of this method is that it includes traditional example reweighting as its special cases, finely succeeding the good generalization and interpretability of previous arts.

C. Example Reweighting for Probabilistic Modeling

The example reweighting strategy has also been explored in probabilistic modeling to improve the robustness of statistical models against outliers. Examples include manually assigning binary weights to data points [45], and covariate shift adaptation/importance sampling [46]–[49]. Different from these methods, the RPM method proposed by Wang *et al.* [50] treats the example weights as latent variables, and directly includes them in the probabilistic generative model. By this formulation, the weights can be automatically inferred from data without custom design. However, RPM is mainly designed for dealing with the outliers (or label noise), and thus not be applicable in class imbalance situations. Besides, there are still no investigations of revealing the probabilistic insight of meta-learning approach for example reweighting, which yet should be specifically meaningful for this line of research to deeply understand its statistical insight and further improve its stability.

D. Robust Learning Against Label Noise

As discussed before, example reweighting methods can be used to improve the robustness of machine learning, especially deep learning, under label noise. Here, we briefly review other typical robust learning methods against label noise, in addition to example reweighting ones, which can mainly be categorized into two groups.

The first group addressed this robust issue by designing robust loss. For example, Zhang and Sabuncu [23] proposed the generalized cross entropy (GCE) loss, which can be seen as a combination of mean absolute error (MAE) and cross entropy (CE) loss, Wang *et al.* [51] proposed to add the reversed cross entropy (RCE) term to CE loss to improve the robustness of deep learning, and Amid *et al.* [52] introduced two temperature parameters to the traditional softmax layer and CE loss, making the loss bounded and heavy-tail. Many robust loss methods are closely related to example reweighting ones. For example, GCE can be optimized by example reweighting [23], and SPL have been shown to optimize some robust losses [21].

The second group tried to correct noisy labels during the training process. For example, Patrini *et al.* [53] proposed to estimating the noise transition matrix, to recover the underlying ground-truth label distribution to guide the training. Hendrycks *et al.* [54] further made use of a small amount of clean data to help the estimation of noise transition matrix. These methods, however, need to estimate the whole noise transition matrix, which might be problematic when the number of class is relatively large. Another methodology is to directly correct the noisy labels by exploiting the prediction of network. For example, Reed [55] adopted the bootstrapping loss to incorporate a perceptual consistency term, which assigns a new label by the convex combination of current prediction and the original noisy one. Tanaka *et al.* [56] proposed a joint optimization framework, which uses two progressive steps to update the whole data labels and classifier weights alternatively. These label correction methods highly rely on the generated soft labels, which depend on the quality of classifiers trained on the noisy data, and thus may produce degraded results if such classifiers have poor performance.

III. Probabilistic MW-Net

A. Review of Deterministic MW-Net

Since our PMW-Net is based on the deterministic MW-Net, we first briefly review the optimization model and the learning algorithm of it. Denote the training set as $\mathcal{D}^{\text{tr}} = \{x_i^{\text{tr}}, y_i^{\text{tr}}\}_{i=1}^{N}$ and meta set as $\mathcal{D}^{\text{meta}} = \{x_j^{\text{meta}}, y_j^{\text{meta}}\}_{j=1}^{M}$. Our goal is to learn a weighting function for training examples, under which we can properly learn the decision function on the training set, such that the prediction error on the meta set is as small as

possible. By doing this, the generalization on unseen test data is expected to be finely ameliorated, since the meta set is chosen unbiased. Mathematically, this can be formulated as the following bi-level optimization problem

$$\begin{aligned}&\min_{\theta} \sum_{j=1}^{M} \ell\big(y_j^{\text{meta}}, f_{\boldsymbol{w}^*(\theta)}\big(x_j^{\text{meta}}\big)\big)\\ &\text{s.t. } \boldsymbol{w}^*(\theta) = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} v_\theta\big(\ell\big(y_i^{\text{tr}}, f_{\boldsymbol{w}}\big(x_i^{\text{tr}}\big)\big)\big)\ell\big(y_i^{\text{tr}}, f_{\boldsymbol{w}}\big(x_i^{\text{tr}}\big)\big)\end{aligned} \tag{1}$$

where $f_{\boldsymbol{w}}(\cdot)$ is the decision function parameterized by $\boldsymbol{w}$, $v_\theta(\cdot)$ is the MW-Net parameterized by θ and $\ell(\cdot,\cdot)$ is the loss function. For notation convenience, we denote that $\mathscr{L}_i^{\text{tr}}(\boldsymbol{w}) = \ell(y_i^{\text{tr}}, f_{\boldsymbol{w}}(x_i^{\text{tr}}))$ and $\mathscr{L}_j^{\text{meta}}(\boldsymbol{w}) = \ell(y_j^{\text{meta}}, f_{\boldsymbol{w}}(x_j^{\text{meta}}))$. Note that the solution to the optimization in the second line of (1) can be seen as a function with respect to θ. Similar bi-level optimization formulations have recently attracted much attention and shown to be effective for many machine learning tasks, especially on learning-to-teach [44] and hyperparameter optimization [57], [58].

Problem (1) is generally difficult to solve, and thus in [2], an online strategy inspired by MAML [39], which has been shown to be effective and efficient in optimization-based few-shot learning [39], [40], [59], [60], is proposed to alternatively update $\boldsymbol{w}$ and θ. First, we can treat θ as to-be-updated parameters and formulate the parameters of the updated classifier, i.e., $\boldsymbol{w}$, as a function with respect to θ, using a stochastic gradient descent (SGD) step

$$\hat{\boldsymbol{w}}^{(t)}(\theta) = \boldsymbol{w}^{(t)} - \alpha\frac{1}{n}\sum_{i\in\mathcal{B}_t^{\text{tr}}} v_\theta\big(\mathscr{L}_i^{\text{tr}}(\boldsymbol{w})\big)\nabla_{\boldsymbol{w}}\mathscr{L}_i^{\text{tr}}(\boldsymbol{w})\Big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}} \tag{2}$$

where $\mathcal{B}_t^{\text{tr}}$ ($|\mathcal{B}_t^{\text{tr}}| = n$) is the index set of the current batch of training data, and α is the step size. Then the MW-Net parameters θ can be updated, also in an SGD manner

$$\theta^{(t+1)} = \theta^{(t)} - \beta\frac{1}{m}\sum_{j\in\mathcal{B}_t^{\text{meta}}} \nabla_\theta\mathscr{L}_j^{\text{meta}}\big(\hat{\boldsymbol{w}}^{(t)}(\theta)\big)\Big|_{\theta=\theta^{(t)}} \tag{3}$$

where $\mathcal{B}_t^{\text{meta}}$ ($|\mathcal{B}_t^{\text{meta}}| = m$) is the index set of the current batch of meta data, and β is the step size, to make $\hat{\boldsymbol{w}}^{(t)}(\theta)$ perform as best as possible on meta data, i.e., minimizing the upper-level optimization. After that, fixing the parameters of MW-Net as $\theta^{(t+1)}$, the parameters of the decision function are finally updated with example weighting function $v_{\theta^{(t+1)}}(\cdot)$

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \alpha\frac{1}{n}\sum_{i\in\mathcal{B}_t^{\text{tr}}} v_{\theta^{(t+1)}}\big(\mathscr{L}_i^{\text{tr}}\big(\boldsymbol{w}^{(t)}\big)\big)\nabla_{\boldsymbol{w}}\mathscr{L}_i^{\text{tr}}(\boldsymbol{w})\Big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}} \tag{4}$$

which also provides the start point of the gradient step (2) for the next iteration.

By iteratively implementing the above process, the weighting function can be automatically learned with the guide of a small amount of meta data, and the examples can be properly weighted simultaneously with the gradually updated weighting function. Consequently, the classifiers can be robustly trained against data biases. The extensively experiments in [2] have verified the effectiveness of MW-Net.

B. Formulation of Probabilistic MW-Net

1) Constructing Overall Objective: We first derive the overall optimization objective of PMW-Net, which can be seen as a probabilistic generalization of (1). To do this, we first need to build the generative model for the example weighting mechanism on training data. According to [50], such a probabilistic generative model can be simply constructed as

$$\begin{aligned}p\big(\mathcal{D}^{\text{tr}}, \boldsymbol{w}, \boldsymbol{v}\big) &= p\big(\boldsymbol{y}^{\text{tr}}|\boldsymbol{x}^{\text{tr}}, \boldsymbol{w}, \boldsymbol{v}\big)p(\boldsymbol{w})p(\boldsymbol{v})p\big(\boldsymbol{x}^{\text{tr}}\big)\\ &= \frac{1}{Z}\prod_{i=1}^{N} l\big(y_i^{\text{tr}}, f_{\boldsymbol{w}}\big(x_i^{\text{tr}}\big)\big)^{v_i} p(\boldsymbol{w})p(\boldsymbol{v})p\big(\boldsymbol{x}^{\text{tr}}\big)\end{aligned} \tag{5}$$

where $\boldsymbol{y}^{\text{tr}} = (y_1^{\text{tr}}, \ldots, y_N^{\text{tr}})$, $\boldsymbol{x}^{\text{tr}} = (x_1^{\text{tr}}, \ldots, x_N^{\text{tr}})$, $l(\cdot,\cdot)$ is the likelihood function, Z is the normalizing factor, $\boldsymbol{w}$ and $\boldsymbol{v} = (v_1, \ldots, v_N)^T$ are the classifier parameters and example weights on the training set respectively, and $p(\boldsymbol{w})$ and $p(\boldsymbol{v})$ are the priors. By the generative model, we can compute the conditional posterior of $\boldsymbol{w}$ given $\boldsymbol{v}$ as

$$p(\boldsymbol{w}|\mathcal{D}^{\text{tr}}, \boldsymbol{v}) = \frac{p\big(\mathcal{D}^{\text{tr}}, \boldsymbol{w}, \boldsymbol{v}\big)}{p\big(\mathcal{D}^{\text{tr}}, \boldsymbol{v}\big)} = \frac{p\big(\boldsymbol{y}^{\text{tr}}, \boldsymbol{w}, \boldsymbol{v}|\boldsymbol{x}^{\text{tr}}\big)}{\int p\big(\boldsymbol{y}^{\text{tr}}, \boldsymbol{w}, \boldsymbol{v}|\boldsymbol{x}^{\text{tr}}\big)\mathrm{d}\boldsymbol{w}}. \tag{6}$$

With the above conditional posterior, we can formulate the probabilistic objective regarding the goal of MW-Net. As discussed before, the optimization goal of MW-Net is to minimize the total loss evaluated on the meta set. This, in a probabilistic way, can be interpreted as maximizing the marginal log-likelihood

$$\log p\big(\boldsymbol{y}^{\text{meta}}|\mathcal{D}^{\text{tr}}, \boldsymbol{x}^{\text{meta}}\big) \tag{7}$$

where $\boldsymbol{y}^{\text{meta}} = (y_1^{\text{meta}}, \ldots, y_N^{\text{meta}})$ and $\boldsymbol{x}^{\text{meta}} = (x_1^{\text{meta}}, \ldots, x_N^{\text{meta}})$, by integrating out the model parameters, where

$$\begin{aligned}&p\big(\boldsymbol{y}^{\text{meta}}|\mathcal{D}^{\text{tr}}, \boldsymbol{x}^{\text{meta}}\big)\\ &\quad= \int_{\boldsymbol{v}}\int_{\boldsymbol{w}} p\big(\boldsymbol{y}^{\text{meta}}, \boldsymbol{w}, \boldsymbol{v}|\mathcal{D}^{\text{tr}}, \boldsymbol{x}^{\text{meta}}\big)\mathrm{d}\boldsymbol{w}\mathrm{d}\boldsymbol{v}\\ &\quad= \int_{\boldsymbol{v}}\int_{\boldsymbol{w}} \prod_{j=1}^{M} p\big(y_j^{\text{meta}}|x_j^{\text{meta}}, \boldsymbol{w}\big)p\big(\boldsymbol{w}|\mathcal{D}^{\text{tr}}, \boldsymbol{v}\big)p(\boldsymbol{v})\mathrm{d}\boldsymbol{w}\mathrm{d}\boldsymbol{v}.\end{aligned} \tag{8}$$

Then we consider the variational formulation for the objective (7), to obtain its ELBO and make the optimization tractable. Denoting $q(\boldsymbol{v}|\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{meta}})$ as the approximation to the true posterior $p(\boldsymbol{v}|\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{meta}})$, and also omitting $\mathcal{D}^{\text{tr}}$ and $\mathcal{D}^{\text{meta}}$ in $q(\boldsymbol{v}|\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{meta}})$ for convenience, we can drive the ELBO as follows:

$$\begin{aligned}&\log p\big(\boldsymbol{y}^{\text{meta}}|\mathcal{D}^{\text{tr}}, \boldsymbol{x}^{\text{meta}}\big)\\ &= \mathbb{E}_{q(\boldsymbol{v})}\left[\log\frac{p\big(\boldsymbol{y}^{\text{meta}}, \boldsymbol{v}|\mathcal{D}^{\text{tr}}, \boldsymbol{x}^{\text{meta}}\big)}{q(\boldsymbol{v})}\right]\\ &\quad + D_{\text{KL}}\big(q(\boldsymbol{v})||p\big(\boldsymbol{v}|\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{meta}}\big)\big)\\ &\geq \mathbb{E}_{q(\boldsymbol{v})}\left[\log\frac{\big[\int p\big(\boldsymbol{y}^{\text{meta}}|\boldsymbol{x}^{\text{meta}}, \boldsymbol{w}\big)p\big(\boldsymbol{w}|\mathcal{D}^{\text{tr}}, \boldsymbol{v}\big)\mathrm{d}\boldsymbol{w}\big]p(\boldsymbol{v})}{q(\boldsymbol{v})}\right]\\ &= \mathbb{E}_{q(\boldsymbol{v})}\big[\log\mathbb{E}_{p(\boldsymbol{w}|\mathcal{D}^{\text{tr}}, \boldsymbol{v})}\big[p\big(\boldsymbol{y}^{\text{meta}}|\boldsymbol{x}^{\text{meta}}, \boldsymbol{w}\big)\big]\big] - D_{\text{KL}}(q(\boldsymbol{v})||p(\boldsymbol{v}))\\ &\triangleq \mathcal{L}(q(\boldsymbol{v}))\end{aligned} \tag{9}$$

where $D_{\mathrm{KL}}(\cdot||\cdot)$ is the KL-divergence between two distributions. The above ELBO can be seen as a probabilistic version of the MW-Net objective (1), if we parameterize $q(\upsilon)$ using a neural network. Specifically, the neural network takes the loss value (or log-likelihood) of an example as its input and output the parameters of $q(\upsilon)$ (detailed in Section III-B3). In such way, the weight υ can be seen as a stochastic function of loss value. We denoted such parameterization of $q(\upsilon)$ as $q_\theta(\upsilon)$.

In practice, we can further simplify this ELBO using a point estimation instead of the full posterior in the expectation $\mathbb{E}_{p(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v})}[p(\boldsymbol{y}^{\mathrm{meta}}|\boldsymbol{x}^{\mathrm{meta}},\boldsymbol{w})]$. Specifically, we treat $p(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v})$ as a Dirac delta distribution

$$p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v}\big)\approx\delta(\boldsymbol{w}-\boldsymbol{w}^*(\boldsymbol{v})) \tag{10}$$

where

$$\begin{aligned}\boldsymbol{w}^*(\boldsymbol{v})&=\arg\max_{\boldsymbol{w}}\log p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v}\big)\\&=\arg\max_{\boldsymbol{w}}\sum_{i=1}^{N}\upsilon_i\log l\big(y_i^{\mathrm{tr}},f_{\boldsymbol{w}}\big(\boldsymbol{x}_i^{\mathrm{tr}}\big)\big)+\log p(\boldsymbol{w})\end{aligned} \tag{11}$$

is the maximum *a posteriori* (MAP) estimation for $\boldsymbol{w}$, according to (6). This approximation is reasonable in deep learning, since fully computing the posterior $p(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v})$ requires a huge number of parameters, which is generally intractable and often unnecessary in practice for a deep neural network. We can now approximate the concerned expectation as

$$\mathbb{E}_{p(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v})}\big[p\big(\boldsymbol{y}^{\mathrm{meta}}|\boldsymbol{x}^{\mathrm{meta}},\boldsymbol{w}\big)\big]\approx p\big(\boldsymbol{y}^{\mathrm{meta}}|\boldsymbol{x}^{\mathrm{meta}},\boldsymbol{w}^*(\boldsymbol{v})\big) \tag{12}$$

and correspondingly the ELBO defined in (9) as

$$\begin{aligned}\mathcal{L}(q_\theta(\boldsymbol{v}))\approx{}&\mathbb{E}_{q_\theta(\boldsymbol{v})}\big[\log p\big(\boldsymbol{y}^{\mathrm{meta}}|\boldsymbol{x}^{\mathrm{meta}},\boldsymbol{w}^*(\boldsymbol{v})\big)\big]\\&-D_{\mathrm{KL}}(q_\theta(\boldsymbol{v})||p(\boldsymbol{v}))\triangleq\mathcal{L}^{\mathrm{meta}}\big(q_\theta(\boldsymbol{v}),\boldsymbol{w}^*(\boldsymbol{v})\big).\end{aligned} \tag{13}$$

Then the optimization model of our PMW-Net is finally formulated as

$$\begin{aligned}&\max_{\theta}\ \mathcal{L}^{\mathrm{meta}}\big(q_\theta(\boldsymbol{v}),\boldsymbol{w}^*(\boldsymbol{v})\big)\\&\text{s.t. }\boldsymbol{w}^*(\boldsymbol{v})=\arg\max_{\boldsymbol{w}}\log p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v}\big).\end{aligned} \tag{14}$$

It is noted that if we assume uninformative priors for $\boldsymbol{w}$ and $\boldsymbol{v}$, such that $p(\boldsymbol{w})\propto 1$ and $p(\boldsymbol{v})\propto 1$, and also apply point approximation to $q_\theta(\boldsymbol{v})$, then (14) degenerates to (1). In our formulation, we keep the uninformative prior for $\boldsymbol{w}$, since there are many other ways for regularizing network parameters other than explicitly introducing priors in deep learning practice. However, we introduce prior to $\boldsymbol{v}$, and aim at learning the full posterior $q_\theta(\boldsymbol{v})$ parameterized by the PMW-Net in an amortized fashion [61] as discussed later. This prior, to some degree, plays the role of regularizing the learned weighting function.

2) Optimization of Classifier: Once $q_\theta(\boldsymbol{v})$ is learned by (14), we can update the classifier parameters $\boldsymbol{w}$ with several ways. The simplest one is to sample one $\boldsymbol{v}$, denoted as $\bar{\boldsymbol{v}}$, from $q_\theta(\boldsymbol{v})$, and solve the following MAP problem:

$$\boldsymbol{w}^*=\arg\max_{\boldsymbol{w}}\log p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\bar{\boldsymbol{v}}\big). \tag{15}$$

However, this tends to lead to instability for learning $\boldsymbol{w}$, and degenerate the overall performance. A more reasonable way is to apply MAP estimation conditioning also on $\mathcal{D}^{\mathrm{meta}}$

$$\begin{aligned}\boldsymbol{w}^*&=\arg\max_{\boldsymbol{w}}\log p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\mathcal{D}^{\mathrm{meta}}\big)\\&=\arg\max_{\boldsymbol{w}}\log\int p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v}\big)p\big(\boldsymbol{v}|\mathcal{D}^{\mathrm{tr}},\mathcal{D}^{\mathrm{meta}}\big)\mathrm{d}\boldsymbol{v}\\&\approx\arg\max_{\boldsymbol{w}}\log\mathbb{E}_{q_\theta(\boldsymbol{v})}\big[p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v}\big)\big]\end{aligned} \tag{16}$$

by integrating $\boldsymbol{v}$ out. Since $\log\mathbb{E}_{q_\theta(\boldsymbol{v})}\big[p(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v})\big]$ is generally intractable, we can further lower bound it as

$$\log\mathbb{E}_{q_\theta(\boldsymbol{v})}\big[p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v}\big)\big]\geq\mathbb{E}_{q_\theta(\boldsymbol{v})}\big[\log p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v}\big)\big]. \tag{17}$$

Correspondingly, the final optimization for $\boldsymbol{w}$ becomes

$$\begin{aligned}\boldsymbol{w}^*&=\arg\max_{\boldsymbol{w}}\ \mathbb{E}_{q_\theta(\boldsymbol{v})}\big[\log p\big(\boldsymbol{w}|\mathcal{D}^{\mathrm{tr}},\boldsymbol{v}\big)\big]\\&=\arg\max_{\boldsymbol{w}}\sum_{i=1}^{N}\mathbb{E}_{q_\theta(\boldsymbol{v})}[\upsilon_i]\log l\big(y_i^{\mathrm{tr}},f_{\boldsymbol{w}}\big(\boldsymbol{x}_i^{\mathrm{tr}}\big)\big)+\log p(\boldsymbol{w})\\&=\arg\max_{\boldsymbol{w}}\sum_{i=1}^{N}\mathbb{E}_{q_\theta(\boldsymbol{v})}[\upsilon_i]\log l\big(y_i^{\mathrm{tr}},f_{\boldsymbol{w}}\big(\boldsymbol{x}_i^{\mathrm{tr}}\big)\big)\end{aligned} \tag{18}$$

where the last equality holds by the uninformative prior assumption $p(\boldsymbol{w})\propto 1$. When $q_\theta(\boldsymbol{v})$ is known, $\mathbb{E}_{q_\theta(\boldsymbol{v})}[\upsilon_i]$ is easy to compute.

3) Specification of Probabilistic Weighting Function: We finally need to discuss the prior and posterior of $\boldsymbol{v}$ in the PMW-Net formulation. As suggested in [50], we assume the bank of Beta priors for $\boldsymbol{v}$

$$p(\boldsymbol{v})=\prod_{i=1}^{N}p(\upsilon_i)=\prod_{i=1}^{N}\mathrm{Beta}(\upsilon_i;a_0,b_0) \tag{19}$$

where a_0 and b_0 are the hyperparameters, which can be adaptively tuned during the learning process as discussed in Section III-D2. With this setting, it is reasonable and convenient to also consider a bank of Beta distributions for the approximate posterior $q_\theta(\boldsymbol{v})$

$$\begin{aligned}q_\theta(\boldsymbol{v})&=\prod_{i=1}^{N}q_\theta(\upsilon_i)\\&=\prod_{i=1}^{N}\mathrm{Beta}\big(\upsilon_i;a_\theta\big(L_i^{\mathrm{tr}}(\boldsymbol{w})\big),b_\theta\big(L_i^{\mathrm{tr}}(\boldsymbol{w})\big)\big)\end{aligned} \tag{20}$$

where $L_i^{\mathrm{tr}}(\boldsymbol{w})=\log l(y_i^{\mathrm{tr}},f_{\boldsymbol{w}}(x_i^{\mathrm{tr}}))$ for convenience, and $a_\theta(\cdot)$ and $b_\theta(\cdot)$ denote the to-be-learned PMW-Net. It is seen that the parameters of the posterior are given in an amortized fashion with the input $L_i^{\mathrm{tr}}(\boldsymbol{w})$s, which shares the similar idea with the deterministic MW-Net while also extends it.

C. Learning Process

Similar as MW-Net, the learning process of PMW-Net can be decomposed into two alternating steps, i.e., updating θ and $\boldsymbol{w}$. We discuss the two steps, as well as the setting of hyperparameters, in the following. The overall learning process is summarized in Algorithm 1 (with implementing considerations discussed in Section III-D). For notation convenience, we denote that $L_j^{\mathrm{meta}}(\boldsymbol{w})=\log l(y_j^{\mathrm{meta}},f_{\boldsymbol{w}}(x_j^{\mathrm{meta}}))$.

1) Updating θ: Updating θ with respect to objective (14) requires solving the lower level optimization, which would be time-consuming and generally intractable in practice. Therefore, instead of completely computing $\boldsymbol{w}^*(\boldsymbol{v})$, we consider a stochastic gradient descent (SGD) step for the parameters $\boldsymbol{w}$

$$\hat{\boldsymbol{w}}^{(t)}(\boldsymbol{v}) = \boldsymbol{w}^{(t)} + \alpha \frac{1}{n} \sum_{i \in \mathcal{B}_t^{\text{tr}}} v_i \nabla_{\boldsymbol{w}} L_i^{\text{tr}}(\boldsymbol{w}) \Big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}} \tag{21}$$

where

$$v_i \sim \text{Beta}\big(v_i; a_\theta\big(L_i^{\text{tr}}(\boldsymbol{w})\big), b_\theta\big(L_i^{\text{tr}}(\boldsymbol{w})\big)\big). \tag{22}$$

Notice that

$$\begin{aligned}&\mathcal{L}^{\text{meta}}(q_\theta(\boldsymbol{v}), \boldsymbol{w}(\boldsymbol{v})) \\ &\quad = \sum_{j=1}^{M} \mathbb{E}_{q_\theta(\boldsymbol{v})}\big[L_j^{\text{meta}}(\boldsymbol{w}(\boldsymbol{v}))\big] - \sum_{i=1}^{N} D_{\text{KL}}(q_\theta(v_i)||p(v_i))\end{aligned} \tag{23}$$

by the prior and posterior assumptions for $\boldsymbol{v}$. Then the PMW-Net parameters θ can be updated in a one-step SGD manner

$$\begin{aligned}\theta^{(t+1)} = \theta^{(t)} &+ \beta \frac{1}{m} \sum_{j \in \mathcal{B}_t^{\text{meta}}} \nabla_\theta \mathbb{E}_{q_\theta(\boldsymbol{v})}\big[L_j^{\text{meta}}\big(\hat{\boldsymbol{w}}^{(t)}(\boldsymbol{v})\big)\big]\Big|_{\theta=\theta^{(t)}} \\ &- \beta \frac{1}{m} \sum_{i \in \mathcal{B}_t^{\text{tr}}} \nabla_\theta D_{\text{KL}}(q_\theta(v_i)||p(v_i))\Big|_{\theta=\theta^{(t)}}.\end{aligned} \tag{24}$$

It is a little tricky to compute the gradient in the above equation, since intractable expectations are involved. Nevertheless, the generalized reparameterization trick [62] can be used to estimate the corresponding gradient, which has already been integrated into the `rsample()` method in PyTorch [63]. In practice, one example sampling is usually enough for ensuring good performance.

2) Updating $\boldsymbol{w}$: Once θ has been updated, $\boldsymbol{w}$ can also be updated with one-step SGD, with respect to the objective function in (18)

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \alpha \frac{1}{n} \sum_{i \in \mathcal{B}_t^{\text{tr}}} \mathbb{E}_{q_{\theta^{(t+1)}}(v_i)}[v_i] \nabla_{\boldsymbol{w}} L_i^{\text{tr}}(\boldsymbol{w}) \Big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}. \tag{25}$$

D. Implementing Considerations

1) Controlling the KL Regularization: As discussed previously, the KL divergence term can be seen as a regularization to θ. However, since the number of training examples is generally much larger than that of meta examples, i.e., $N \gg M$, the KL regularization might be too strong and dominate the objective $\mathcal{L}^{\text{meta}}(q_\theta(\boldsymbol{v}), \boldsymbol{w}(\boldsymbol{v}))$, which tends to restrict the posterior produced by PMW-Net being too close to prior. To alleviate this issue, we introduce a balancing parameter λ to control the intensity of the KL term, and the objective correspondingly becomes

$$\begin{aligned}&\mathcal{L}_\lambda^{\text{meta}}(q_\theta(\boldsymbol{v}), \boldsymbol{w}(\boldsymbol{v})) \\ &\quad = \sum_{j=1}^{M} \mathbb{E}_{q_\theta(\boldsymbol{v})}\big[L_j^{\text{meta}}(\boldsymbol{w}(\boldsymbol{v}))\big] - \lambda \sum_{i=1}^{N} D_{\text{KL}}(q_\theta(v_i)||p(v_i)).\end{aligned} \tag{26}$$

Algorithm 1 The PMW-Net Learning Algorithm

Input: Training dataset $\mathcal{D}^{\text{tr}}$, meta-dataset $\mathcal{D}^{\text{meta}}$, batch size n, m, number of iterations T.

Output: Classifier and PMW-Net parameters $\boldsymbol{w}^{(T)}$, $\theta^{(T)}$.

1: Initialize $\boldsymbol{w}^{(0)}$ and $\theta^{(0)}$.

2: **for** $t = 0$ **to** $T - 1$ **do**

3: $\{x_i^{\text{tr}}, y_i^{\text{tr}}\}_{i \in \mathcal{B}_t^{\text{tr}}} \leftarrow \text{SampleMiniBatch}(\mathcal{D}^{\text{tr}}, n)$.

4: $\{x_j^{\text{meta}}, y_j^{\text{meta}}\}_{j \in \mathcal{B}_t^{\text{meta}}} \leftarrow \text{SampleMiniBatch}(\mathcal{D}^{\text{meta}}, m)$.

5: Formulate the update function of classifier parameters $\hat{\boldsymbol{w}}^{(t)}(\boldsymbol{v})$ by Eq. (21):

$$\hat{\boldsymbol{w}}^{(t)}(\boldsymbol{v}) = \boldsymbol{w}^{(t)} + \alpha \frac{1}{n} \sum_{i \in \mathcal{B}_t^{\text{tr}}} v_i \nabla_{\boldsymbol{w}} L_i^{\text{tr}}(\boldsymbol{w}) \Big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}.$$

6: Update PMW-Net parameters $\theta^{(t+1)}$ by Eq. (27) using reparameterization gradient:

$$\begin{aligned}\theta^{(t+1)} = \theta^{(t)} &+ \beta \frac{1}{m} \sum_{j \in \mathcal{B}_t^{\text{meta}}} \nabla_\theta \mathbb{E}_{q_\theta(\boldsymbol{v})}[L_j^{\text{meta}}(\hat{\boldsymbol{w}}^{(t)}(\boldsymbol{v}))]\Big|_{\theta=\theta^{(t)}} \\ &- \beta \frac{\lambda}{m} \sum_{i \in \mathcal{B}_t^{\text{tr}}} \nabla_\theta D_{\text{KL}}(q_\theta(v_i)||q_{\theta^{(t)}}(v_i))\Big|_{\theta=\theta^{(t)}}.\end{aligned}$$

7: Update classifier parameters $\boldsymbol{w}^{(t+1)}$ by Eq. (25):

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \alpha \frac{1}{n} \sum_{i \in \mathcal{B}_t^{\text{tr}}} \mathbb{E}_{q_{\theta^{(t+1)}}(v_i)}[v_i] \nabla_{\boldsymbol{w}} L_i^{\text{tr}}(\boldsymbol{w}) \Big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}.$$

8: **end for**

It is noted that when $\lambda < 1$, the above objective is no longer an approximation to the strict ELBO (9). Nevertheless, it is still reasonable if we treat the first term as "loss" and the second "regularization." In all our experiments, we simply set $\lambda = 10^{-3}$ and find it works well.

2) Adapting Weights Prior: Compared with the deterministic MW-Net, there are two hyperparameters extra introduced in PMW-Net, i.e., a_0 and b_0 in the prior $p(\boldsymbol{v})$ of example weights. To avoid manually tuning these hyperparameters, we adopt the following strategy: $p(\boldsymbol{v})$ is first initialized by setting $a_0 = b_0 = 1$ in an uninformative way, and then we replace the prior $p(\boldsymbol{v})$ with $q_{\theta^{(t)}}(\boldsymbol{v})$ when computing $\theta^{(t+1)}$ during the iteration. Then, combining the λ parameter discussed in Section III-D1, the updating (24) becomes

$$\begin{aligned}\theta^{(t+1)} = \theta^{(t)} &+ \beta \frac{1}{m} \sum_{j \in \mathcal{B}_t^{\text{meta}}} \nabla_\theta \mathbb{E}_{q_\theta(\boldsymbol{v})}\big[L_j^{\text{meta}}\big(\hat{\boldsymbol{w}}^{(t)}(\boldsymbol{v})\big)\big]\Big|_{\theta=\theta^{(t)}} \\ &- \beta \frac{\lambda}{m} \sum_{i \in \mathcal{B}_t^{\text{tr}}} \nabla_\theta D_{\text{KL}}(q_\theta(v_i)||q_{\theta^{(t)}}(v_i))\Big|_{\theta=\theta^{(t)}}.\end{aligned} \tag{27}$$

This strategy is reasonable, since it enforces the weights of examples not changing to much between two adjacent iterations, which is expected to make the training more stable.

3) Choice of Optimizer: In Section III-C, we have presented the updating of θ in an SGD fashion. Obviously, this updating step can also be implemented with any optimizer other than SGD. Therefore, we use the Adam [64] optimizer for θ in all our experiments. For the network parameters $\boldsymbol{w}$, we still adopt

TABLE I
AVERAGE TEST ACCURACY (±STD) (%) OF ALL COMPETING METHODS UNDER CLOSED-SET NOISE WITH DIFFERENT NOISE RATIO η

Dataset	Method	No Noise	Symmetric Noise			Asymmetric Noise	
			$\eta = 0.2$	$\eta = 0.4$	$\eta = 0.6$	$\eta = 0.2$	$\eta = 0.4$
CIFAR-10	Baseline	92.89±0.32	76.83±2.30	70.77±2.31	63.21±4.22	76.83±2.30	70.77±2.31
	Forward [53]	92.03±0.11	86.49±0.15	80.51±0.28	75.55±2.25	87.38±0.48	78.98±0.35
	SPL [19]	91.40±0.39	87.53±0.48	81.49±0.34	75.87±0.25	85.99±1.77	82.71±0.99
	GCE [23]	90.03±0.30	88.51±0.37	85.48±0.16	81.09±0.23	88.55±0.22	83.31±0.14
	Fine-tuning	**93.23±0.89**	81.47±0.35	75.34±1.21	67.43±1.32	82.47±3.64	74.07±1.56
	GLC [54]	92.43±0.27	89.06±0.30	84.78±0.43	80.52±0.87	89.68±0.33	**88.92±0.24**
	L2RW [1]	89.25±0.37	87.42±0.39	84.96±0.28	80.04±0.67	87.86±0.36	85.66±0.51
	MW-Net [2]	92.04±0.15	89.19±0.57	86.10±0.18	81.31±0.37	90.33±0.61	87.54±0.23
	PMW-Net	92.45±0.12	**89.84±0.25**	**86.79±0.13**	**82.20±0.29**	**90.47±0.14**	87.69±0.25
CIFAR-100	Baseline	70.50±0.12	50.86±0.27	43.01±1.16	34.43±0.94	50.86±0.27	43.01±1.16
	Forward [53]	67.81±0.61	58.75±0.38	52.53±0.15	39.44±1.03	63.28±0.23	57.90±0.57
	SPL [19]	68.26±0.25	58.41±0.38	50.16±0.30	40.23±0.47	61.05±0.61	52.09± 0.26
	GCE [23]	67.39±0.12	61.97±0.43	54.33±0.35	41.73±0.36	62.07±0.41	55.25±0.09
	Fine-tuning	**70.74±0.25**	54.45±0.43	48.12±1.35	38.52±1.02	56.98±1.50	46.37±0.25
	GLC [54]	69.75±0.17	60.30±0.63	55.25±0.69	44.86±0.57	63.07±0.53	**62.22±0.62**
	L2RW [1]	65.42±0.23	56.28±0.45	52.64±0.56	42.15±0.34	57.47±1.16	50.98±1.55
	MW-Net [2]	68.53±0.33	62.22±0.28	57.34±0.47	47.03±0.76	64.22±0.28	58.64±0.47
	PMW-Net	70.28±0.18	**65.54±0.33**	**60.62±1.04**	**52.11±1.52**	**64.95±0.21**	58.72±0.16

the original SGD optimizer (details are provided in experiment section).

IV. EXPERIMENTS

A. Label Noise Experiments

1) Experimental Settings: We consider both closed-set noise, which indicates that the dataset is corrupted by labeling selected examples to wrong classes within the dataset, and open-set noise [65], which means irrelevant examples are labeled as certain classes within the considered dataset. For closed-set noise, we employ two benchmark datasets, i.e., CIFAR-10 and CIFAR-100 [66], and manually corrupt them with label noise. In specific, we consider two types of label noise: 1) Symmetric noise, for which noisy labels are generated by independently changing the label of each example to a random class with probability η [3], and 2) Asymmetric noise, for which the label of each example is flipped to similar classes with total probability η. For open-set noise, following the similar settings in [65], we construct the corrupted datasets through randomly substituting the examples of CIFAR-10 by examples from: 1) ImageNet32 and 2) ImageNet32 and CIFAR-100. The ratio of open-set noise varies from 20% to 60%.

In each experiment, 1000 examples with clean labels are randomly selected from the validation set to construct the meta set. For the DNN classifier, we adopt the ResNet-32 [67]. The competing methods include: Baseline, referring to ResNet-32 directly trained on the biased training set without pre or post-processing; Fine-tuning, meaning that fine-tune the Baseline classifier on the meta data with clean labels; representative robust learning methods Forward [53], SPL [19] and GCE [23]; typical meta-learning methods L2RW [1] and MW-Net [2]; label correction method GLC [54]. Among all the implemented ones, meta-learning methods, together with Fine-tuning and GLC, use the information of the additional meta set, while the rest not.

All the methods are trained using SGD with a momentum 0.9, a weight decay 5×10^{-4} and an initial learning rate 0.1. The learning rate is divided by 10 after the 40th and 50th epoch (for a total of 60 epochs), respectively, in closed-set noise experiments, and after the 60th and 120th epoch (for a total of 200 epochs), respectively, in open-set noise experiments.

2) Results of Closed-Set Noise Experiments: The average performance of each method, in terms of test accuracy, over 5 runs is summarized in Table I. As can be observed from the table, methods that make use of meta-data generally perform better than others, especially when the noise ratio is high. Among these methods, our PMW-Net achieves the best performance across almost all datasets and noise ratios, except the second for 40% asymmetric noise, showing its evident robustness against label noise. It is also seen that, our PMW-Net outperforms the original MW-Net in all cases, especially on CIFAR-100 dataset, in which the noisy fusion structure of data could be more complex due to more classes being introduced, making it more challenging to distinguish the clean examples from noisy ones. This better performance can be attributed to the more properly weighting of PMW-Net, as show in Fig. 2. In this figure, we compare the weight distributions learned by MW-Net and PMW-Net, and observe that our PMW-Net can better distinguish clean examples from the noisy ones by assigning smaller weights to them, and vice versa. The promising reweighting ability of PMW-Net can be possibly explained by its stochastic weighting manner, which gives the weight of an example more chances to escape from a local suboptimal solution and be properly tuned (more analyses of such mechanism can be found in Section IV-A4). In contrast, due to the deterministic weighting manner of MW-Net, the weight is more easily to be stuck to an improper value during learning. This naturally leads to more robust performance of PMW-Net against label noise.

To further explore the properties of PMW-Net, we depict the probabilistic weighting function learned by it in Fig. 1(b), as compared with the deterministic one learned by MW-Net

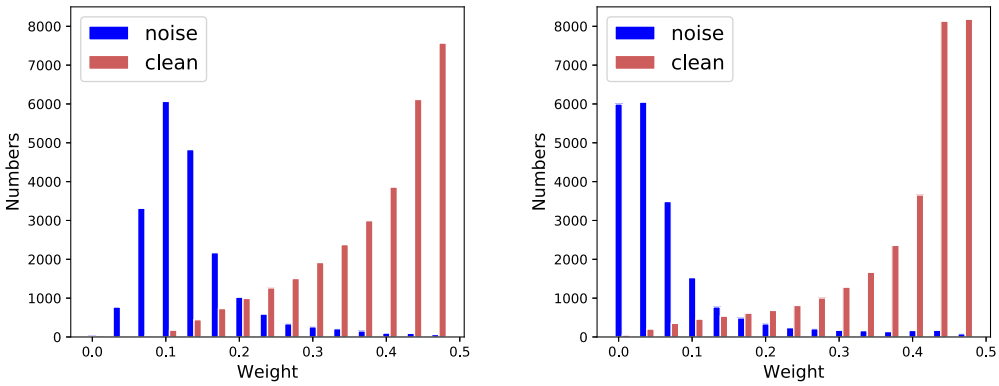


Fig. 2. Example weight distributions by MW-Net (left) and PMW-Net (right) under 40% symmetric noise on CIFAR-100 dataset.

TABLE II
TEST ACCURACY (%) OF ALL COMPETING METHODS UNDER OPEN-SET NOISE ON CIFAR-10

Open-set Noise Type	Noise Ratio	Baseline	Forward [53]	SPL [19]	GCE [23]	Fine-tuning	GLC [54]	L2RW [1]	MW-Net [2]	PMW-Net
ImageNet32	20%	88.93	89.54	88.95	89.27	89.11	88.64	87.64	88.89	**90.42**
	40%	85.28	85.77	85.93	86.17	87.41	86.46	86.38	85.25	**87.61**
	60%	81.11	81.77	81.80	81.29	80.72	81.47	81.21	82.12	**83.50**
ImageNet32 + CIFAR-100	20%	88.49	87.73	89.01	89.56	89.39	88.47	87.49	89.53	**90.07**
	40%	85.45	85.52	85.07	86.38	86.52	86.65	86.82	87.10	**87.98**
	60%	80.73	81.65	81.57	81.82	81.38	81.12	81.66	82.12	**83.42**

in Fig. 1(a). It is easy to see that, both of MW-Net and PMW-Net can properly weight the example according to its loss value, i.e., assigning relatively smaller weight to the example with larger loss and vice versa. However, our PMW-Net additionally provides the variance information, which measures the uncertainty of the weight. It is interesting that, the variance or uncertainty tends to vanish as the loss becomes large, indicating that we are more confident to weight examples with large loss values. This is consistent to human experiences of the label noise, that examples with large loss values tend to be with noisy label, and the larger the loss, the more possible it is noisy.

3) Results of Open-Set Noise Experiments: The performance of each method in terms of test accuracy is summarized in Table II. It can be seen that our PMW-Net outperforms other competing methods in all cases, showing its robustness against the open-set label noise. Another interesting observation is that, GLC, which is very competitive under the close-set noise, performs not very well, and even worse than some methods that do not make use of clean meta data. This could be attributed to the fact that the label correction model behind GLC intrinsically relies on the close-set noise assumption, which deviates from the open-set noise situation. In contrast, example weighting methods, including MW-Net and PMW-Net, provide a universally available solution, though might not be optimal in specific cases, against all types of label noise.

To further compare MW-Net and PMW-Net, we depict in Fig. 3 the tendency curves for the differences of weights calculated between adjacent epochs. Ten noisy examples are randomly chosen to draw the mean curve, surrounded by the region illustrating the standard deviations calculated on them. It is easily seen that, although the differences of weights given by both methods converge close to zero, the curve of PMW-Net is with less variations. Such a tendency of PMW-Net is reasonable, since we generally do not expect the algorithm switching the status of an example between "noisy" and "clean" too often. This advantage of PMW-Net should be attributed to the KL regularization term in the optimization objective.

We also plot in Fig. 4 the tendency curves of the mini-batch training accuracy, as well as the test accuracy on clean test data, in the open-set label noise experiments. From the figure, we can find that the BaseModel can easily overfit to the noisy labels, whose training accuracy keeps increasing during learning process, while test accuracy tends to decrease after the first learning rate decays. In contrast, both MW-Net and our PMW-Net achieve lower training accuracy but decrease less than BaseModel in test accuracy, showing their robustness against label noise. Besides, the test accuracy of PMW-Net decreases less significant than MW-Net, showing that it is less affected by the noise examples.

4) Discussion on the Weighting Uncertainty: One of the additional features of PMW-Net compared with MW-Net is that it provides uncertainties to example weights. To see the benefits brought by this feature, we select four typical "noisy" examples, which are labeled as "dog," from the corrupted training set under 60% open-set label noise, and see their mean weights and weight variances given by PMW-Net, shown in Table III. Here, the noisy examples indicate the images out of the original CIFAR-10, specifically from ImageNet, according to the construction process detailed in the main text.

As illustrated in Table III, the first two examples are clearly noisy ones, and thus are unsurprisingly assigned very small weights by PMW-Net. Besides, the weight variances are also very small, showing that PMW-Net is confident to recognize them as noisy examples.

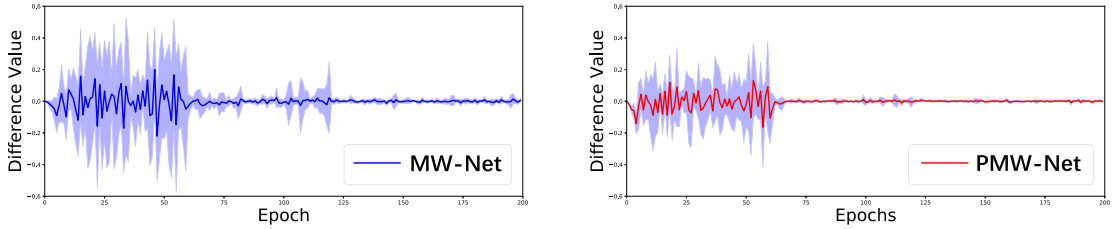


Fig. 3. Weight variation curves by (a) MW-Net and (b) PMW-Net on CIFAR-10 under open-set noise induced by ImageNet32 with noise ratio 60%.

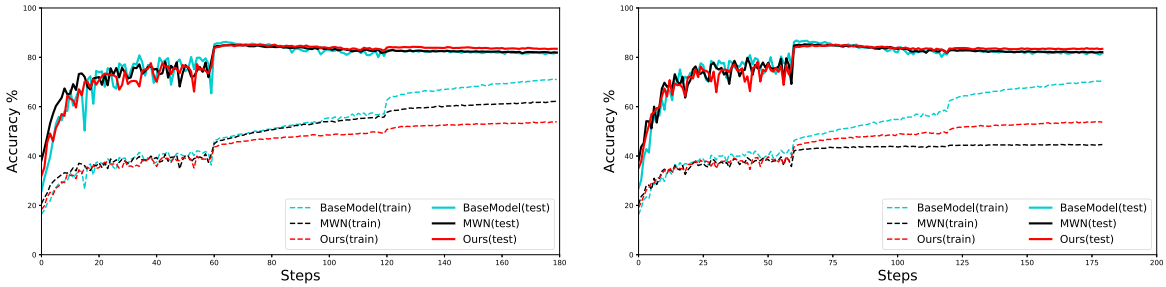


Fig. 4. Tendency curves of training and test accuracies on CIFAR-10 under open-set label noise with noise ratio 60%. (a) Corrupted by ImageNet32. (b) Corrupted by ImageNet32 and CIFAR-100.

TABLE III

ILLUSTRATION OF THE MEAN WEIGHTS AND THEIR VARIANCES FOR THE SELECTED "NOISY" EXAMPLES

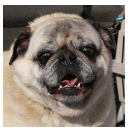
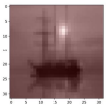
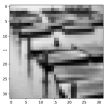
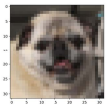
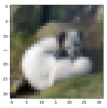
Original Image from ImageNet				
Noisy Example (resized)				
Weight Mean	9.644e-10	3.164e-10	0.4012	0.4287
Weight Variance	5.577e-10	1.832e-10	0.1085	0.1076

TABLE IV

TEST ACCURACY (%) OF ALL COMPETING METHODS ON LONG-TAILED CIFAR-10 AND CIFAR-100

Datasets	Long-Tailed CIFAR-10						Long-Tailed CIFAR-100					
Imbalance Factor	200	100	50	20	10	1	200	100	50	20	10	1
Baseline	65.68	70.36	74.81	82.23	86.39	92.89	34.84	38.32	43.85	51.14	55.71	70.50
Focal Loss [11]	65.29	70.38	76.71	82.76	86.66	93.03	35.62	38.41	44.32	51.95	55.78	70.52
Class-Balanced [68]	68.89	74.57	79.27	84.36	87.49	92.89	36.23	39.60	45.32	52.59	57.99	70.50
Fine-tuning	66.08	71.33	77.42	83.37	86.42	**93.23**	**38.22**	41.83	46.40	52.11	57.44	**70.72**
L2RW [1]	66.51	74.16	78.93	82.12	85.19	89.25	33.38	40.23	44.44	51.64	53.73	64.11
MW-Net [2]	68.91	75.21	80.06	84.94	87.84	92.66	37.91	42.09	46.74	54.37	58.46	70.37
PMW-Net	**68.94**	**75.53**	**80.53**	**85.08**	**88.17**	92.78	37.92	**42.21**	**46.76**	**54.52**	**58.71**	70.43

For the third example, although is considered as a noisy one by the construction, its true label in ImageNet is also "dog," which is the same to its target in the noisy data, and is reasonably assigned to relatively a large weight by PMW-Net.

The most interesting observation is about the fourth example. Specifically, it is originally labeled as "fox" in ImageNet, and as "dog" in the constructed noisy training set. PMW-Net assigned a relatively large weight, as well as large variance to it. This coincides with human intuition that the fox in the image looks very similar to a dog, and thus we are likely to recognize it as a clean example but less confident about it. Besides, the training of the weight-net can also be benefitted from this uncertainty. In specific, for the deterministic MW-Net, if a noisy example was assigned to

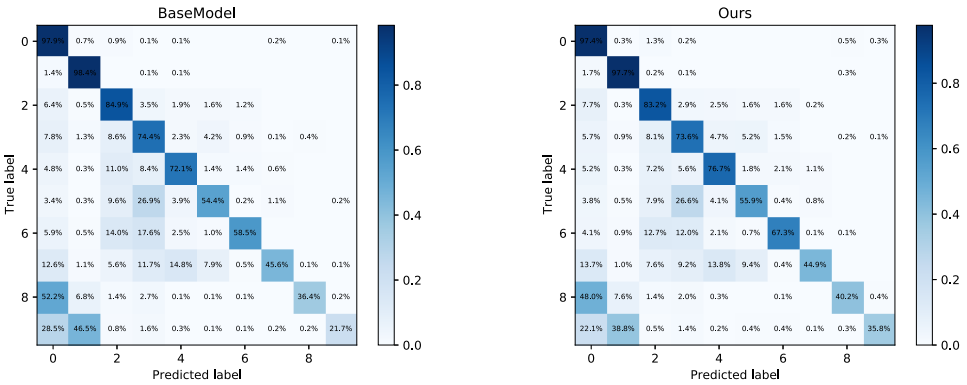


Fig. 5. Prediction confusion matrix by baseline ce loss (basemodel) and PMW-Net (ours) on long-tailed CIFAR-10 with imbalance factor 200.

large weight, it would negatively affect the final performance. In comparison, by the stochastic fashion of weight assigning, if the weight variance is large, there is still a chance for assigning small weights to noisy examples even if their mean weights are large, when updating the weight-net, and thus the negative effects could be alleviated to some degree.

B. Class Imbalance Experiments

For the class imbalance experiments, we adopt the Long-Tailed CIFAR dataset [68], that reduces the number of training data per class of the original CIFAR dataset according to an exponential function $\bar{n}_i = n_i \mu^i$, where i is the class index, n_i is the original number of training examples in the ith class, and $\mu \in (0, 1)$. Then the imbalance factor of a dataset is defined as the number of training examples in the largest class divided by that of the smallest. We randomly select 10 examples per class in the validation set as the meta set. The compared methods include: Baseline, referring to ResNet-32 directly trained on the imbalanced training set; Fine-tuning, which fine-tunes the Baseline classifier on the balanced meta data; Focal Loss [11] and Class-Balanced [68], which represent the state-of-the-arts of the predefined example reweighting techniques for dealing with the class imbalance issue; L2RW [1] and MW-Net [2], which automatically learn the example weights with an additional meta set. Among all the implemented methods, Fine-tuning, L2RW, MW-Net, and PMW-Net make use of the information of the additional meta data.

All the models are trained using SGD with a momentum 0.9, a weight decay 5×10^{-4} and an initial learning rate 0.1. The learning rate for ResNet-32 is divided by 10 after the 80th and 90th epoch (for a total 100 epochs), respectively. The learning rate for MW-Net and the proposed PMW-Net is fixed as 10^{-5}. The performance of each method in terms of test accuracy is summarized in Table IV.

It can be seen from Table IV that, our method outperforms other competing methods on datasets with class imbalance in most cases, showing its robustness against this kind of data bias. It should be mentioned that, when the imbalance factor is 200 on Long-tailed CIFAR-100, the smallest class has only two samples, and thus it is not surprising that fine-tuning the classifier on the balanced meta set performs better than reweighting based methods, including L2RW, MW-Net, and PMW-Net. Besides, when the imbalance factor is 1, i.e., all classes are with the same number of examples, our method can still have promising performance compared with other methods. As in the previous label noise experiments, we also plot the curves of the weighting functions learned by MW-Net and PMW-Net in Fig. 1(b) and (d). The similar conclusion can be drawn as before, i.e., we are more confident to weight the examples with larger loss values, which, in this case, tends to be more important for characterizing the decision boundaries of classification.

For intuitively visualizing the effectiveness of the proposed PMW-Net, we also draw in Fig. 5 the prediction confusion matrix produced by it on Long-Tailed CIFAR-10 with imbalance factor 200, in comparison with the baseline method. As can be seen from the figure, in this extremely difficult situation, our method can significantly improve the prediction accuracy for most of the classes with relatively fewer examples. For example, for the class with fewest examples, the prediction accuracy can be boosted from 21.7% to 35.8% by PMW-Net, which consequently leads to a better overall performance.

C. Real-World Data Experiments

To further verify the availability of the proposed method on real-world problems, we conduct experiments on the Clothing 1M dataset [26], which contains 1 million images of clothing collected from online shopping websites. The images are categorized to 14 classes, including T-shirt, Shirt, Knitwear, etc. The labels contain many errors, since they are generated using the surrounding texts of the images provided by sellers. Besides, the dataset is also imbalanced, i.e., the size of the smallest class "Sweater" (19743 examples) is about only one fifth of that of the largest one "Dress" (92177 examples). According to [26], the collectors have manually refined the labels of a small portion of images, which are considered as clean validation data. We thus randomly chose 7k examples form the validation set as the meta set. Following the previous work [53], [56], we adopt ResNet-50 pre trained on ImageNet [8] as the classifier. For pre processing, we resize the images to 256×256, crop the middle 224×224, and perform normalization. We use SGD with a momentum 0.9,

TABLE V
CLASSIFICATION ACCURACY (%) OF DIFFERENT METHODS ON CLOTHING 1M TEST SET

Method	CE	Forward [53]	GCE [23]	Joint Opt. [56]	DMI [69]	LCCN [70]	MLNT [71]	MW-Net [2]	PMW-Net
Accuracy	68.94	69.84	69.75	72.23	72.46	73.07	73.47	73.72	**73.80**

TABLE VI
CLASSIFICATION ACCURACY (±STD) (%) OF PMW-NET WITH DIFFERENT SIZES OF META ON CIFAR-10 WITH 20% SYMMETRIC NOISE

Size of Meta Set	500	1000	1500
Accuracy	89.32±0.11	89.39±0.20	89.43±0.19

TABLE VII
TEST ACCURACY (%) UNDER DIFFERENT NOISE SETTINGS ON CIFAR-100 WITH PRE LEARNED WEIGHTING FUNCTION

	Symmetric Noise			Asymmetric Noise	
	$\eta = 0.2$	$\eta = 0.4$	$\eta = 0.6$	$\eta = 0.2$	$\eta = 0.4$
Accuracy	65.42	61.35	51.99	64.31	54.36

a weight decay 10^{-3}, an initial learning rate 0.01, and batch size 32. The learning rate for ResNet-50 is divided by 10 after the 5th epoch (for a total 10 epochs), and the learning rate for MW-Net and the proposed PMW-Net is fixed as 10^{-3}.

The results are summarized in Table V. It can be seen that the proposed method achieves the best accuracy among all competing methods, especially that it improves the performance of MW-Net, which further verifies the effectiveness of our probabilistic formulation.

D. Further Analyses and Discussions

1) Effect of the Size of Meta Set: Since our method depend on the meta set, it is necessary to analyze the effect of its size, to provide a practical guideline. Specifically, we conduct experiments with 20% symmetric noise on CIFAR-10, by varying the size of meta set from 500 to 1500 for the proposed PMW-Net. The results in terms of accuracy averaged over 5 runs are summarized in Table VI[2]. It can be seen that, the performance of PMW-Net is quite stable with respect to the size of meta set, though slight improvement can be achieved by adding more meta data. We may thus say that, a moderate size of meta set is sufficient for a promising performance in practice.

2) Generalization Ability of Learned Weighting Function: In conventional meta-learning, the learned methodologies are expected to be generalized to other similar tasks. Therefore, here we investigate the generalization ability of the learned weighting function. In specific, we only run the complete PMW-Net algorithm to CIFAR-10 with 20% symmetric noise, and then freeze the learned weight-net and use it to weight examples on noisy CIFAR-100 to guide training classifier. We consider all noise settings used in Table I, and summarize the test results in Table VII. It is easy to see that the performance is still competitive as compared with that of Table I, where the full PMW-Net is applied. These results indicate that the learned weighting function can, to some degree, well adapt to different types of noise on similar tasks, inheriting the desirable property of conventional meta-learning methods.

[2]Note that the result with 1000 meta data is slightly worse than that reported in Table I. This is due to that the size of training set is smaller here, since we need to keep it being fixed when varying the size of meta set.

3) Running Time Comparison: Since meta-learning based methods are generally known to be complicated, it is necessary to investigate the time complexity of the proposed method. To this aim, we record the average running time per epoch of each competing method on CIFAR-10 with 40% symmetric noise, as summarized in Table VIII. We can see from the table that, meta-learning based methods, including L2RW, MW-Net and the proposed PMW-Net, are unsurprisingly more time-consuming. Nevertheless, although additional operations are involved due to reparameterization, the running time of PMW-Net is not much more than that of MW-Net. Considering the better performance, the extra cost should be affordable.

4) Limitations: The major limitation of the proposed method, as well as other meta data-dependent methods, such as GLC, L2RW, and MW-Net, is that it requires a set of unbiased data, in addition to the training set. This labeling cost of clean meta data tends to limit the applicability of the proposed method in real scenarios, and thus should be avoided. A possible solution is to construct the meta set directly from training data and use certain unsupervised criteria as the meta loss. For example, we could adopt the consistency loss, previously designed for data augmentation [72], which is under our current investigation.

V. EXTENSION TO FULLY BAYESIAN MODELS

In Section III, although the PMW-Net has been formulated in Bayesian fashion, we have restricted the learning machine as a deterministic one, i.e., using point estimation to $\boldsymbol{w}$. Nevertheless, it is straightforward to also treat $\boldsymbol{w}$ in a probabilistic way to achieve a fully Bayesian formulation. In this section, we discuss such an extension and illustrate a typical example application.

A. Fully Bayesian Formulation for PMW-Net

The key to extending PMW-Net to fully Bayesian models is approximating the posterior $p(\boldsymbol{w}|\mathcal{D}^{\text{tr}}, \boldsymbol{v})$ with a full distribution instead of a point estimation in (10)

$$p\big(\boldsymbol{w}|\mathcal{D}^{\text{tr}}, \boldsymbol{v}\big) \approx q_\phi(\boldsymbol{w}|\boldsymbol{v}) \tag{28}$$

where ϕ is the parameters to define the approximate posterior $q_\phi(\boldsymbol{w}|\boldsymbol{v})$. By this approximation, and following the derivations in Section III-B, we can present the optimization model of

TABLE VIII
AVERAGE RUNNING TIME (S) PER EPOCH OF ALL COMPETING METHODS ON CIFAR-10 WITH 40% SYMMETRIC NOISE

Method	Baseline	Forward	SPL	GCE	Fine-tuning	GLC	L2RW	MW-Net	PMW-Net
Time	38.83	38.21	46.35	38.50	39.57	39.39	134.50	140.59	144.46

PMW-Net within fully Bayesian framework, which can be seen as an extension of (14), as

$$\begin{aligned} &\max_{q_\theta(\boldsymbol{v})} \mathcal{L}^{\text{meta}}\big(q_\theta(\boldsymbol{v}), q_{\phi^*}(\boldsymbol{w}|\boldsymbol{v})\big) \\ &\text{s.t. } q_{\phi^*}(\boldsymbol{w}|\boldsymbol{v}) = \arg\max_{q_\phi(\boldsymbol{w}|\boldsymbol{v})} \mathcal{L}^{\text{tr}}\big(q_\phi(\boldsymbol{w}|\boldsymbol{v})\big) \end{aligned} \tag{29}$$

where

$$\begin{aligned} &\mathcal{L}^{\text{meta}}\big(q_\theta(\boldsymbol{v}), q_\phi(\boldsymbol{w}|\boldsymbol{v})\big) \\ &\quad = \mathbb{E}_{q_\theta(\boldsymbol{v})}\left[\log \mathbb{E}_{q_\phi(\boldsymbol{w}|\boldsymbol{v})}\left[\prod_{j=1}^{M} p\big(y_j^{\text{meta}}|x_j^{\text{meta}}, \boldsymbol{w}\big)\right]\right] \\ &\qquad - D_{\text{KL}}(q_\theta(\boldsymbol{v})||p(\boldsymbol{v})) \end{aligned} \tag{30}$$

and

$$\begin{aligned} &\mathcal{L}^{\text{tr}}\big(q_\phi(\boldsymbol{w}|\boldsymbol{v})\big) \\ &\quad = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\boldsymbol{w}|\boldsymbol{v})}\big[v_i \log l\big(y_i^{\text{tr}}, f_{\boldsymbol{w}}\big(x_i^{\text{tr}}\big)\big)\big] - \log Z \\ &\qquad - D_{\text{KL}}(q_\phi(\boldsymbol{w}|\boldsymbol{v})||p(\boldsymbol{w})) \end{aligned} \tag{31}$$

where Z is the normalizing factor irrelevant to the to-be-learned parameters, similar as that in (5).

The learning process is similar as that discussed in Section III-C, except that we update ϕ to parameterize the distribution of $\boldsymbol{w}$. Also note that the expectations with respect to $q_\phi(\boldsymbol{w}|\boldsymbol{v})$ might not be able to analytically calculated, and therefore Monte Carlo estimation can generally be used instead.

B. Example Application to Bayesian Logistic Regression

We apply this fully Bayesian formulation of PMW-Net to a typical model for classification, i.e., Bayesian logistic regression, to demonstrate its effectiveness.

The generative model of Bayesian logistic regression can be formulated as

$$\begin{aligned} y_i &\sim \text{Bernoulli}\big(\sigma\big(\boldsymbol{w}^T x_i\big)\big) \\ \boldsymbol{w} &\sim \mathcal{N}(\boldsymbol{0}, \gamma \boldsymbol{I}) \end{aligned} \tag{32}$$

where $\sigma(z) = 1/(1+e^{-z})$ is the sigmoid function, $\mathcal{N}(\cdot,\cdot)$ denotes the multivariate Gaussian distribution, and γ is the hyperparameter. It is noted that one could further impose prior to γ to make the model more flexible, while we leave it as a fixed hyperparameter, which is set to 100 in this work, for simplicity.

We then use a Gaussian distribution to approximate the posterior of $\boldsymbol{w}$

$$q_\phi(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{33}$$

where the to-be-estimated parameters are $\phi = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$.

1) Toy Data Example: We first show a toy example, which is similar as that used in [73]. The clean data points of two classes are generated from two 2-D Gaussians

$$\begin{aligned} p(x|y=+1) &= \mathcal{N}\big((1,-1)^T, \boldsymbol{I}\big) \\ p(x|y=-1) &= \mathcal{N}\big((3,3)^T, 16\boldsymbol{I}\big). \end{aligned}$$

The outliers are also generated from a 2-D Gaussian

$$p(x|y=+1) = \mathcal{N}\big((7,0)^T, 0.01\boldsymbol{I}\big)$$

which are labeled as $+1$ but located in the region of -1 class, as can be seen in Fig. 6. We generate 2000 data, with 1000 for each class, as clean training data, and add certain number of outliers, varying from 0 to 800, to construct the final training set. Then we further generate 80 data (40 for each class) as meta set. The compared methods include: KL, referring to learning the approximate posterior with respect to the KL-divergence minimization in the traditional variational inference (VI); RPM, denotes the example reweighting method proposed in [50]; β-divergence, a robust Bayesian learning method using β-divergence [73]. For a fair comparison, we tune the parameter of β-divergence via optimizing the performance on meta set.

The results are shown in Fig. 6. It is easy to observed that, all the methods obtain the similar decision boundaries when only clean data are used for training. As a moderate number of outliers introduced, the decision boundaries by traditional KL based VI and RPM begin to diverge from that learned in the no outlier case, while β-divergence and PMW-Net can still obtain reasonable decision boundaries. However, when more outliers are added, the performance of β-divergence also degenerates, and only PMW-Net consistently perform robustly.

2) UCI Data Experiments: We then construct experiments on three UCI dataset, Heart Disease, Breast Cancer, and Waveform. For Heart Disease and Breast Cancer, we randomly split the original datasets to training, meta and test sets with ratio 4 : 1 : 5, while for Waveform, since the training/test sets has already been specified, we only split the original training set to training and meta sets with ratio 4 : 1. Then we randomly select data points from the training set, and flip their labels to simulate noisy data. The noise ratio varies from 0 to 0.2. The compared methods are the same as that in the previous toy example.

The results averaged over five randomly runs are summarized in Table IX. As can be seen from the table, the models learned by traditional KL-based VI give degenerated performance as label noise introduced, while other robust Bayesian learning methods can more or less alleviate this issue. Among all the methods, the proposed PMW-Net method consistently achieves the best performance in all cases, showing its superior robustness. Beside, when noise ratio is zero, i.e., no label noise is introduced to the training set, PMW-Net also achieve

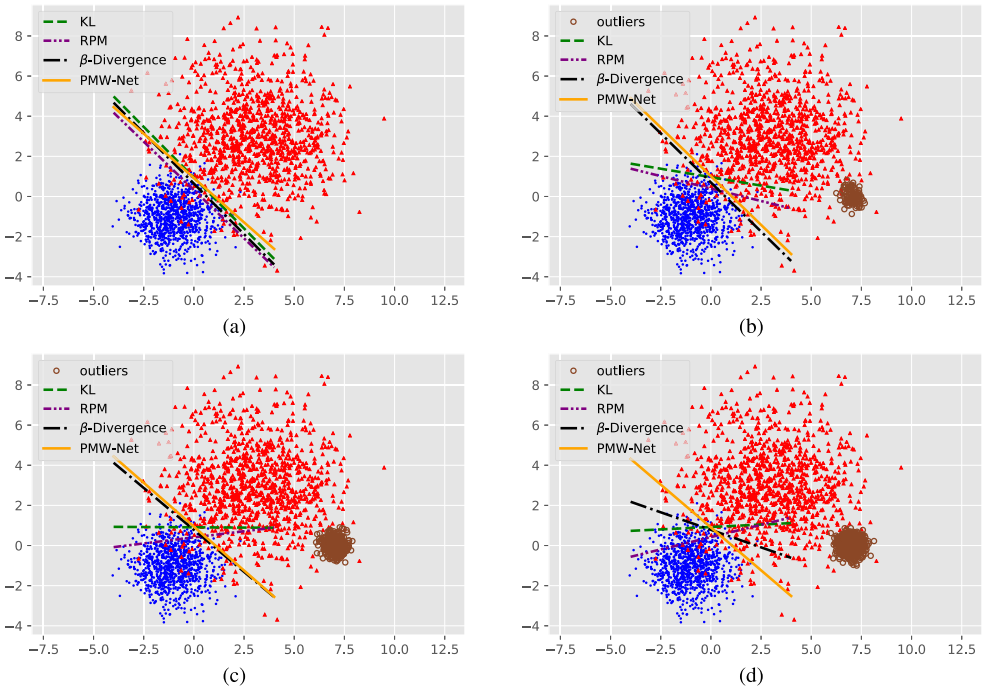


Fig. 6. Decision boundaries of Bayesian logistic regression learned by different methods on toy data with varying number of outliers. (a) No outlier. (b) 100 outliers. (c) 400 outliers. (d) 800 outliers.

TABLE IX
AVERAGE TEST ACCURACY (±STD) (%) OF ALL COMPETING METHODS FOR BAYESIAN LOGISTIC REGRESSION ON THREE UCI DATASETS

Datasets	Heart Disease			Breast Cancer			Waveform		
Noise Ratio	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2
KL	80.00±2.38	79.47±2.51	78.29±2.24	96.96±0.66	94.80±1.10	94.21±1.71	86.24±0.24	82.58±1.31	78.64±1.42
RPM	79.61±2.60	80.53±1.07	76.71±2.26	96.78±0.52	96.37±0.71	96.26±0.97	86.08±0.60	85.38±1.18	82.73±2.13
β-divergence	82.50±1.59	83.82±1.93	81.32±1.42	97.02±0.57	96.90±0.60	**96.96±0.57**	83.21±0.19	82.83±0.73	83.29±0.34
PMW-Net	**83.29±2.55**	**84.74±1.47**	**81.45±1.28**	**97.25±0.68**	**97.02±0.81**	**96.96±0.54**	**86.33±0.35**	**85.63±0.53**	**84.22±1.31**

higher accuracy than the traditional KL-based VI, which might be attributed its robustness against class imbalance issue, as detailed in Section IV-B.

VI. CONCLUSION

We have proposed a probabilistic formulation for MW-Net, which weights the example with a probability distribution instead of a deterministic function, for robust deep learning against training set bias. The proposed PMW-Net can not only provide the uncertainty measure for the weight of each example, but also improve the performance of the original MW-Net by the extra randomness and regularization introduced by probabilistic modeling. The effectiveness of our PMW-Net has been empirically evaluated on a series of synthetic and real-world datasets with training set bias, including label noise and class imbalance. We have also extended the PMW-Net framework to fully Bayesian models for improving the robustness of general Bayesian learning problems. In the future, we will try to introduce more useful knowledge to the probabilistic weight function, such as curriculum learning [74], by designing proper priors. We will also devote to applying the proposed methodology to more real problems, such as object recognition, where the training data biases are easily observed and regression task is generally required in addition to classification. Besides, avoiding extra labeling cost for obtaining clean meta set is extremely expected in real applications.

REFERENCES

[1] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4334–4343.
[2] J. Shu *et al.*, "Meta-Weight-Net: Learning an explicit mapping for sample weighting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1917–1928.
[3] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–15.
[4] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[5] 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.
[6] J. Liang, L. Jiang, D. Meng, and A. G. Hauptmann, "Learning to detect concepts from webly-labeled video data," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 1746–1752.
[7] B. Zhuang, L. Liu, Y. Li, C. Shen, and I. Reid, "Attend in groups: A weakly-supervised deep learning framework for learning from web data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2915–2924.

[8] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[9] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2001, p. 1.

[10] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-SVMs for object detection and beyond," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 89–96.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.

[12] S. Jadon, "A survey of loss functions for semantic segmentation," in *Proc. IEEE Conf. Comput. Intell. Bioinf. Comput. Biol. (CIBCB)*, Oct. 2020, pp. 1–7.

[13] B. Neyshabur, S. Bhojanapalli, D. Mcallester, and N. Srebro, "Exploring generalization in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5947–5956.

[14] D. Arpit *et al.*, "A closer look at memorization in deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 233–242.

[15] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and generalization in neural networks: An empirical study," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–21.

[16] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst., Man, C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[17] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Netw.*, vol. 106, pp. 249–259, Oct. 2018.

[18] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2304–2313.

[19] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1189–1197.

[20] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann, "Easy samples first: Self-paced reranking for zero-example multimedia search," in *Proc. 22nd ACM Int. Conf. Multimedia*, Nov. 2014, pp. 547–556.

[21] D. Meng, Q. Zhao, and L. Jiang, "A theoretical understanding of self-paced learning," *Inf. Sci.*, vol. 414, pp. 319–328, Nov. 2017.

[22] F. De la Torre and M. J. Black, "A framework for robust subspace learning," *Int. J. Comput. Vis.*, vol. 54, no. 1, pp. 117–142, Aug. 2003.

[23] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 8778–8788.

[24] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[25] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

[26] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2691–2699.

[27] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, 1995, pp. 278–282.

[28] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[30] 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, no. 1, pp. 321–357, 2002.

[31] Q. Dong, S. Gong, and X. Zhu, "Class rectification hard mining for imbalanced deep learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1869–1878.

[32] Q. Dong, S. Gong, and X. Zhu, "Imbalanced deep learning by minority class incremental rectification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 6, pp. 1367–1381, Jun. 2019.

[33] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, p. 114.

[34] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. 17th Int. Joint Conf. Artif. Intell.*, vol. 2, 2001, pp. 973–978.

[35] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3573–3587, Aug. 2018.

[36] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann, "Self-paced learning with diversity," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2078–2086.

[37] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann, "Self-paced curriculum learning," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2694–2700.

[38] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, pp. 1–54, Dec. 2019.

[39] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[40] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–11.

[41] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4077–4087.

[42] M. Dehghani, A. Mehrjou, S. Gouws, J. Kamps, and B. Schölkopf, "Fidelity-weighted learning," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–20.

[43] Y. Fan, F. Tian, T. Qin, X. Li, and T. Liu, "Learning to teach," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–16.

[44] L. Wu *et al.*, "Learning to teach with dynamic loss functions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6466–6477.

[45] P. J. Huber, *Robust Statistics*. New York, NY, USA: Wiley, 1981.

[46] E. Veach and L. J. Guibas, "Optimally combining sampling techniques for Monte Carlo rendering," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1995, pp. 419–428.

[47] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. Statist. Planning Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[48] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation," *J. Mach. Learn. Res.*, vol. 8, pp. 985–1005, May 2007.

[49] J. Wen, C.-N. Yu, and R. Greiner, "Robust learning under uncertain test distributions: Relating covariate shift to model misspecification," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 631–639.

[50] Y. Wang, A. Kucukelbir, and D. M. Blei, "Robust probabilistic modeling with Bayesian data reweighting," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3646–3655.

[51] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 322–330.

[52] E. Amid, M. K. K. Warmuth, R. Anil, and T. Koren, "Robust bi-tempered logistic loss based on Bregman divergences," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates, 2019.

[53] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2233–2241.

[54] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10456–10465.

[55] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–11.

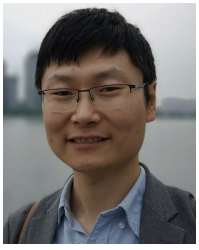
[56] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5552–5560.

[57] M. MacKay, P. Vicol, J. Lorraine, D. Duvenaud, and R. B. Grosse, "Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–25.

[58] Z. Tao, Y. Li, B. Ding, C. Zhang, J. Zhou, and Y. Fu, "Learning to mutate with hypergradient guided population," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates, 2020, pp. 17649–17659.

[59] S. Jadon, "An overview of deep learning architectures in few-shot learning domain," 2020, *arXiv:2008.06365*. [Online]. Available: http://arxiv.org/abs/2008.06365

[60] S. Jadon and A. A. Srinivasan, "Improving siamese networks for one-shot learning using kernel-based activation functions," in *Data Management, Analytics and Innovation*. Singapore: Springer, 2021, pp. 353–367.
[61] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Represent.*, 2014, pp. 1–14.
[62] M. Jankowiak and F. Obermeyer, "Pathwise derivatives beyond the reparameterization trick," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2235–2244.
[63] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.
[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, Y. Bengio and Y. LeCun, Eds., 2015, pp. 1–15.
[65] Y. Wang *et al.*, "Iterative learning with open-set noisy labels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8688–8696.
[66] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.
[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[68] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9268–9277.
[69] Y. Xu, P. Cao, Y. Kong, and Y. Wang, "L_DMI: A novel information-theoretic loss function for training deep nets robust to label noise," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6222–6233.
[70] J. Yao, H. Wu, Y. Zhang, I. W. Tsang, and J. Sun, "Safeguarded dynamic label regression for noisy supervision," in *Proc. Assoc. Adv. Artif. Intell. Conf. Artif. Intell. (AAAI)*, 2019, pp. 9103–9110.
[71] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Learning to learn from noisy labeled data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5046–5054.
[72] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6256–6268.
[73] F. Futami, I. Sato, and M. Sugiyama, "Variational inference based on robust divergences," in *Proc. 21st Int. Conf. Artif. Intell. Statist.*, A. Storkey and F. Perez-Cruz, Eds., vol. 84, 2018, pp. 813–822.
[74] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

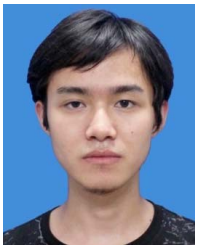


Qian Zhao received the B.Sc. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 2009 and 2015, respectively.

He was a Visiting Scholar with Carnegie Mellon University, Pittsburgh, PA, USA, from 2013 to 2014. He is currently an Associate Professor with the School of Mathematics and Statistics, Xi'an Jiaotong University. His current research interests include low-rank matrix/tensor analysis, Bayesian modeling, and meta-learning.

Jun Shu received the B.Sc. degree from Xi'an Jiaotong University, Xi'an, China, in 2016, where he is currently pursuing the Ph.D. degree with the School of Mathematics and Statistics.

His current research interests include machine learning, meta-learning, and computer vision.

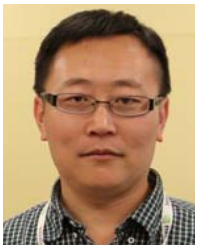


Xiang Yuan received the B.Sc. degree from Xi'an Jiaotong University, Xi'an, China, in 2020, where he is currently pursuing the master's degree with the School of Mathematics and Statistics.

His current research interest includes machine learning.

Ziming Liu received the B.Sc. degree from Xi'an Jiaotong University, Xi'an, China, in 2020, where he is currently pursuing the master's degree with the School of Mathematics and Statistics.

His current research interests include machine learning, Bayesian methods, and differential equations.



Deyu Meng (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. 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. His research interests include model-based deep learning, variational networks, and meta-learning.