Calibration of Few-Shot Classification Tasks: Mitigating Misconfidence from Distribution Mismatch

SUNGNYUN KIM\textsuperscript{1} AND SE-YOUNG YUN\textsuperscript{1}, (Member, IEEE)

\textsuperscript{1}Kim Jaechul Graduate School of Artificial Intelligence, Korea Advanced Institute of Science and Technology (KAIST), Seoul 02455, South Korea

Corresponding author: Se-Young Yun (e-mail: yunseyoung@kaist.ac.kr)

This work was conducted by Center for Applied Research in Artificial Intelligence(CARAI) grant funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD) (UD190031RD). This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)).

ABSTRACT As many meta-learning algorithms improve performance in solving few-shot classification problems for practical applications, the accurate prediction of uncertainty is considered essential. In meta-training, the algorithm treats all generated tasks equally and updates the model to perform well on training tasks. During the training, some of the tasks may make it difficult for the model to infer the query examples from the support examples, especially when a large mismatch between the support set and the query set exists. The distribution mismatch causes the model to have incorrect confidence, which causes a calibration problem. In this study, we propose a novel meta-training method that measures the distribution mismatch and enables the model to predict with more precise confidence. Moreover, our method is algorithm-agnostic and can be readily expanded to include a range of meta-learning models. Through extensive experimentation, including dataset shift, we show that our training strategy prevents the model from becoming indiscriminately confident, and thereby helps the model to produce calibrated classification results without the loss of accuracy.

INDEX TERMS Deep learning, distribution mismatch, few-shot image classification, task calibration

I. INTRODUCTION

WITH the great success of deep learning over the last decade, there has been growing interest in investigating more intuitive methods for mimicking human intelligence. One desirable characteristic of human cognition is the ability to learn new information quickly. Few-shot learning is a problem that requires machines to learn with only a few examples, and meta-learning algorithms have been studied as the solver. Meta-learning algorithms learn how to acclimate rapidly to new data with the help of prior across-task knowledge that is obtained through training on several other few-shot tasks. Noteworthy approaches include learning metric space \cite{1}–\cite{3}, learning update rules \cite{4}, \cite{5}, or learning an initialization \cite{6}, \cite{7}. For the meta-training to solve the few-shot classification (FSC) problem, it is generally assumed that each task randomly chooses classes and randomly samples the support and query examples of the selected classes, where all the generated tasks are treated equally. The meta-training process tries to extract the prior information so that the meta-learned model can perform well on every training task.

However, for some tasks, the model may have difficulty inferring the queries from the support examples, and this difficulty may cause a calibration problem. Fig. 1 depicts such a challenging example of a sampled batch of tasks from a real image set. In Task 1, each query image is easily matched to its corresponding support image (well-defined task), whereas Task 4 includes somewhat unclear matching between the support and the query images (ill-defined task). Due to the distribution mismatch, the model may learn wrong confidences for the predictions of the queries because common meta-learning approaches force the model to learn anyway.

Calibration in FSC is critical yet challenging. The over-/under-confidence of a network can result in dramatically different outcomes in the decision-making process, such as autonomous driving \cite{8} or medical diagnoses \cite{9}. Nevertheless, common calibration methods in classification, such as
We apply the novel meta-update to each sampled trajectory. However, they naïvely utilized the weights in the reinforcement learning context by prioritizing the support set, without considering the distribution match. This novel training allows the model to predict with precise confidence and ultimately obtain better calibration ability.

FIGURE 1. An example of a 4-task batch and their task calibration (our method) weights. $D_{spt}^{(1)}, D_{qry}^{(1)}$, and $w_1$ indicate the support set, query set, and task calibration weight of $i$-th task, respectively. $c_i$ is an $i$-th class in each task. Typical FSC algorithms force the model to classify the query set given the support set, without considering the distribution mismatch.

Platt scaling [10], histogram binning [11], or isotonic regression [12], are not applicable in FSC because they require an assumption that the target examples are from known (trained) classes. In FSC tasks where few training data are given, a model is vulnerable to misconfidence, but recent studies on FSC are focused on boosting accuracy and overlook the calibration of misconfident predictions. We thus design an algorithm-agnostic calibration method for an FSC model by reducing the learning signal from the ill-defined tasks. As Fig. 1 describes, our method gives a different weight to each task regarding the support-query discrepancy. The computed weight in the well-defined Task 1 is high. On the other hand, in Task 4, even the adapted model may not confidently distinguish the queries, which results in a lower weight. This novel training allows the model to predict with precise confidence and ultimately obtain better calibration ability.

To summarize, our contributions are as follows. 1) We propose a simple and straightforward method by which a few-shot classification model learns the ill-defined tasks in a conservative way, so as not to place the wrong confidence. The experiments demonstrate the improved calibration ability of our method. 2) We apply the novel meta-update to other few-shot learning algorithms, including a metric-based framework, and this confirms that our method is versatile and can easily be expanded to work across a range of few-shot learning models. Moreover, we validate our work through a dataset shift experiment, which is designed to be challenging due to the degradation of estimation ability [13].

Our work is the first that quantifies task difficulty and applies task calibration in the FSC problem. Whereas the “difficulty” is indeed an ambiguous term, and several different factors may create a difficult task, such as observation noise, data ambiguity, and data imbalance, in this study by “difficulty” we mean the support-query discrepancy within a task. The work of Xu et al. [14] is somewhat related to our work in that the researchers used rank-based meta-update weights in the reinforcement learning context by prioritizing each sampled trajectory. However, they naïvely utilized the scale of task-specific loss to estimate difficulty.

II. RELATED WORKS
A. FEW-SHOT CLASSIFICATION
Few-shot classification is an extremely challenging task for managing high degrees of uncertainty [15]. In an environment where there are sparse training examples for each task, models suffer from severe instability due to high uncertainty [16]. The estimation and judgment of uncertainty become more challenging when high-variance data, such as a large-scale image set, are used. To overcome this issue, several recent studies have relied on designing Bayesian neural networks. These networks often use posterior approximation to quantify parameter uncertainty [17]–[22].

Probabilistic MAML [23], a variant of MAML [7], approximates the distribution over model parameters by opti-
mizing the variational lower bound. Bayesian MAML [17] uses the Stein variational gradient descent (SVGD) for more flexible modeling of posterior. ABM [18], ML-PIP [24], and ABPML [20] all amortize the approximate inference. Neural Processes [25] and its variants [26], [27] are a combination of a stochastic process and a neural network that learns the distribution of functions while representing uncertainty, although Neural Processes have yet to be proven capable of solving large-scale classification problems. Ghadirzadeh et al. [21] applied the idea of Probabilistic MAML to the few-shot policy adaptation.

The Bayesian models must compute posterior inference, and most approaches use empirical approximation, which usually requires heavy computation. Conversely, our method is computationally advantageous and does not rely on posterior approximation. The reader may also refer to Section IV-D.

B. DISTRIBUTION MISMATCH

Our work was initially inspired by Malinin & Gales [28], which builds a Dirichlet Prior Network to explicitly model distributional uncertainty. Distributional uncertainty is ascribed to a mismatch between the training and test data [28]; in our setup, the mismatch corresponds to the discrepancy between the support and query set and is often caused by intra-class variations [29]. We quantified this discrepancy without requiring any additional parameters or networks. In parallel, Sensoy et al. [30] parameterized Dirichlet, using the evidence vector to regularize the predictive distribution by penalizing divergence from the uncertain state.

Bohdal et al. [31] proposed a novel meta-learning loss that directly minimizes the differentiable expected calibration error (ECE), but the loss function is not widely applicable to existing few-shot learning algorithms. The work of Xu et al. [14] is somewhat related to our work in that they gave rank-based weights to the sampled trajectories. By evaluating the importance of each trajectory and balancing the loss, they prevented the model from showing bias on some specific tasks. Our task calibration, in contrast, quantifies the distribution mismatch within each task and balances the loss to mitigate the misconfidence for the ill-defined tasks.

III. TASK CALIBRATION METHOD

A. PRELIMINARIES

1) Problem Setup

A meta-learning problem is composed of tasks from a task distribution \( p(\tau) \), and for each meta-training task \( \tau \sim p(\tau) \), there is a training set and test set. We denote each as a support set, \( D_{\text{sp}}(\tau) \), and a query set, \( D_{\text{qry}}(\tau) \), respectively. In typical FSC scenarios, each task is comprised of data from \( N \) randomly selected classes and \( k \) support examples per class, with a total of \( kN \) examples, \( \{(s^{(x)}_s, y^{(x)}_s)\}_{s=1}^{kN} := D_{\text{sp}}(\tau) \). This is called the \( N \)-way \( k \)-shot problem. Moreover, there is a disjoint query set, \( \{(q^{(x)}_q, y^{(x)}_q)\}_{q=1}^{kN} := D_{\text{qry}}(\tau) \). Meta-training is an iterative process of optimization on \( D_{\text{qry}}(\tau) \) after having seen \( D_{\text{sp}}(\tau) \). This setup is reminiscent of learning a novel task with only a few examples of training data. The adaptation on \( D_{\text{sp}}(\tau) \) corresponds to training with the novel task data. Given a few examples, a meta-learner is expected to perform predictions on unseen data, which in our setup is \( D_{\text{qry}}(\tau) \). That is, the meta-learner evolves by acquiring general information across the tasks by optimizing on \( D_{\text{qry}}(\tau) \).

2) Model-Agnostic Meta-Learning (MAML)

Before we introduce our new approach, we briefly review model-agnostic meta-learning (MAML) [7] to facilitate readers’ understanding. MAML learns a proper initialization of a globally shared parameter \( \theta \) that is quickly adjusted to each task \( \tau \) using a few gradient descent steps. After it is adapted to the support examples, MAML updates \( \theta \) by optimizing on the query set, and it minimizes task-specific loss \( L_\tau \) with a task-adapted parameter \( \phi_\tau \). The bi-level optimization is computed as follows:

\[
\phi_\tau \leftarrow \theta - \beta \nabla_{\theta} L(\theta; D_{\text{sp}}(\tau)) \tag{1}
\]

\[
\theta \leftarrow \theta - \frac{\alpha}{T} \sum_{\tau=1}^{T} \nabla_{\theta} L_{\tau}(\phi_\tau; D_{\text{qry}}(\tau)) \tag{2}
\]

With a task difficulty measure, we can easily modify the bi-level optimization such that task-specific losses are weighted adaptively according to difficulty. Note that the meta-update in (2) aggregates gradients over multiple tasks. By measuring the task difficulty, we can simply multiply a difficulty-dependent weight \( w_\tau \) to each task \( \tau \), i.e., \( L = \sum_{\tau=1}^{T} w_\tau L_{\tau}(\phi_\tau) \), where \( T \) is the number of tasks in a meta-batch. If a task \( \tau \) has a large distribution mismatch, a small value of \( w_\tau \) can decrease the contribution of \( \nabla_{\theta} L_{\tau} \).
becomes greater, it becomes more difficult to distinguish one from another.

We compute the inter-class query embeddings, which are measured in the embedding space mapped by $h_{\phi_{\tau}}$, the feature extractor before the classifier. With the pairs of query inputs and labels $(x_q^{(\tau)}, y_q^{(\tau)})$, we collect the average features for every class $c$,

$$\overline{h}_{\phi_{\tau},c} := \frac{1}{k^c} \sum_{(q,y_q^{(\tau)})=c} h_{\phi_{\tau}}(x_q^{(\tau)})$$

where $\overline{h}$ maintains the feature dimensions. $\overline{h}_{\phi_{\tau}}$ contains information about the support and query data because the average features of the query inputs are parameterized by the task parameter $\phi_{\tau}$, while $\phi_{\tau}$ is derived from the adaptation on the support data as (1). Therefore, the relation between the support and query data determines $\overline{h}_{\phi_{\tau}}$. By normalizing and concatenating $N$-way query embeddings, we obtain a matrix $H_{\tau} \in \mathbb{R}^{N \times m}$, where $m$ is the embedding dimension.

If we assume $\mathcal{L}$ as the cross-entropy loss, $\phi_{\tau}$ from the task adaptation (1) is an approximation of the maximum likelihood estimate (MLE) with observed support examples.

$$\phi_{\tau} = \arg \max_{\theta} \prod_{s} p(y_{s}^{(\tau)}|x_{s}^{(\tau)}, \theta); (x_{s}^{(\tau)}, y_{s}^{(\tau)}) \in D_{spt}^{(\tau)}$$

Let $\phi_{\tau}$ be the parameter from a task where there is no distribution mismatch. This is satisfied when $D_{spt}^{(\tau)}$ has exactly the same distribution as $D_{qry}^{(\tau)}$. We aim to maximize $p(D_{qry}^{(\tau)}|\theta)$ in meta-learning, which is achieved when $\theta = \phi_{\tau}$, from (4) and $D_{spt}^{(\tau)} \approx D_{qry}^{(\tau)}$. That is, the parameter that was adapted to the support set sufficiently distinguishes the query set. We can assume that to make a clear distinction between different classes, the embedded representation vectors should have low correlation. Therefore, in task $\tau^*$, the embedded vectors of the query inputs have low correlation because the classification boundary that $\phi_{\tau}$ establishes optimally distinguishes $D_{spt}^{(\tau^*)}$ (Fig. 2 left).

In contrast, if there is a significant distribution mismatch between $D_{spt}^{(\tau)}$ and $D_{qry}^{(\tau)}$, $\phi_{\tau}$ is a suboptimal solution for maximizing $p(D_{qry}^{(\tau)}|\theta)$, as the task-specific parameter cannot infer $D_{spt}^{(\tau)}$ from $D_{qry}^{(\tau)}$. This implies a higher correlation between the embedded queries (Fig. 2 right).

Thus, we can compare the inter-class cosine similarity between task $\tau^*$ and $\tau$ using the average features.

$$\|H_{\tau}^\top H_{\tau^*}\| \leq \|H_{\tau}^\top H_{\tau}\|$$

where $H_{\tau}H_{\tau^*}^\top$ is the inter-class cosine similarity matrix because

$$[H_{\tau}H_{\tau^*}^\top]_{ij} = \frac{\overline{h}_{\phi_{\tau},i}^\top \overline{h}_{\phi_{\tau^*},j}}{\|\overline{h}_{\phi_{\tau},i}\| \|\overline{h}_{\phi_{\tau^*},j}\|}$$

for all $i, j \in \{1, \ldots, N\}$. From now, we will simply use its norm as a similarity score $s_{\tau} := \|H_{\tau}H_{\tau^*}^\top\|$. For a general task $\tau$, a significant mismatch between the distribution of $D_{spt}^{(\tau)}$ and $D_{qry}^{(\tau)}$ makes the inequality gap (5) larger.

We use cosine similarity as it was used in the numerous contrastive learning literature (e.g., SimCLR [32] and MoCo [33]) to measure the distance between the feature representations. Contrastive learning studies use cosine similarity to embed positive samples close together and negative samples far apart. This has been shown to improve the alignment and uniformity of the representations on the hypersphere [34]. As such, we supposed the cosine similarity to be an appropriate criterion for measuring the distance between the features; its magnitude is bounded to $[-1, 1]$. We also experimented with Euclidean distance rather than cosine similarity, but the results were not significant in our experiments. We suppose that this is because the magnitude of Euclidean distance is not bounded, and the similarity scores were not accurate because some tasks have very large distance scales whereas others have very small distance scales.

2) Similarity Scores of Few-Shot Tasks

Fig. 3 shows the similarity scores for tasks from different distributions of real-world images. We measured the inter-class similarity scores of 1000 training tasks, where the tasks are sampled from some manipulated distribution. The dataset we used here was mini-ImageNet [35], and we followed a 5-way 1-shot MAML experimental protocol. We next describe the way that we manipulated the task sampling distribution.

- **Standard** is a general 1-shot experiment. For the support set, 1 image is sampled from each class. For the query set, 15 images are sampled from the corresponding class.
- **Well-defined** tasks are sampled so that there is no distribution mismatch as $\tau^*$. Although it is difficult to perfectly eliminate the mismatch, it is possible to make the distribution of $D_{spt}^{(\tau)}$ and $D_{qry}^{(\tau)}$ similar. To this end, we sampled 100 support images per class and 15 corresponding query images. By changing the few-shot tasks into data-rich many-shot tasks, we could simulate in the low uncertainty regime. The parameter was adapted to 100 support images (but we did not change the number of adaptation steps) to allow $\phi_{\tau}$ to be near-optimal to maximize the likelihood of the query set.
Algorithm 1: Task Calibration Model-Agnostic Meta-Learning (TCMAML, ours)

1. Require: \( p(\tau) \): distribution over tasks;
2. Require: \( T \): task meta-batch size;
3. Require: \( \alpha, \beta \): step size hyperparameters;
4. randomly initialize \( \theta \);
5. while not done do
6. Sample batch of tasks \( \tau \sim p(\tau) \);
7. for \( \tau = 1, \ldots, T \) do
8. Evaluate \( \nabla_{\theta} L(\theta; D_{spt}(\tau)) \) w.r.t. the support set;
9. Compute adapted parameter:
10. \( \phi_{\tau} \leftarrow \theta - \beta \nabla_{\theta} L(\theta; D_{spt}(\tau)) \);
11. for class \( c = 1, \ldots, N \) do
12. Compute average feature \( \overline{H}_{\phi_{\tau},c} \) as (3);
13. Normalize the feature vector;
14. end
15. Concatenate the normalized feature vectors:
16. \( H_{\tau} \in \mathbb{R}^{N \times m} \);
17. Set similarity score: \( s_{\tau} \leftarrow \|H_{\tau}H_{\tau}^T\| \);
18. end
19. Compute meta-update weight for each task as (8);
20. Update
21. \( \theta \leftarrow \theta - \alpha \sum_{\tau=1}^{T} \nabla_{\theta} \left[ w_{\tau} \cdot L_{\tau}(\phi_{\tau}; D_{qry}(\tau)) \right] \);
22. end

- **Poorly-defined** tasks are constructed to create an extremely large support-query discrepancy and a large distribution mismatch. Similar to the Standard setting, a Poorly-defined task samples 1 support image and 15 query images, but they are from different classes. For instance, the class labels of the support images could be \{hair slide, carousel, wok, photocopier, jellyfish\}, and the class labels of the query images could be \{ant, vase, dalmatian, school bus, king crab\}.

In Fig. 3, in the extremely discrepant case where the classes of the support images do not correspond to those of the query images, inter-class similarity is high because the model cannot infer the query set by observing the support set. On the other hand, in well-defined tasks, the adapted model is good at distinguishing the query representations because they are less correlated. General FSC tasks lie between these well-defined and poorly-defined tasks. The similarity score thus indicates whether a given input task has a large distribution mismatch or not.

### C. TASK CALIBRATION MAML

As we emphasized, our method is applicable to various few-shot learning algorithms; nevertheless, for better understanding, we begin by applying our method to the MAML algorithm [7]. We propose Task Calibration model-agnostic meta-learning (TCMAML), a variant of MAML, to estimate the task difficulty during meta-training and utilize it to modulate the optimization of global parameters. Specifically, we measure the distribution mismatch for each task and constrain the feedback from highly mismatched tasks (see Fig. 2). In Section III-B, we elaborated on how the similarity between the average features is related to the distribution mismatch.

The aggregated meta-loss in (2) assumes equal reliability for uniformly sampled tasks: \( 1/T \). We alternatively use different weights for each task, which allows for training with prior knowledge about task configuration. TCMAML estimates \( w_{\tau} \) as a normalized \(-s_{\tau}\) value. Consequently, we modify the meta-update rule (2) as follows:

\[
\theta \leftarrow \theta - \alpha \sum_{\tau=1}^{T} \nabla_{\theta} \left[ w_{\tau} \cdot L_{\tau}(\phi_{\tau}; D_{qry}(\tau)) \right], \quad w_{\tau} = \text{norm}(-s_{\tau})
\]

(7)

From (6) and (7), the inequality (5) leads to smaller weight values on the loss function of the misaligned tasks. Designing \( w_{\tau} \) as above implies there is a large weight used for the well-defined tasks, where the measured inter-class feature similarity is low, and a small weight used for the ill-defined tasks, where the similarity is high. In this sense, TCMAML considers the difficulty of a task by measuring the similarity between the query examples.

The full procedure of TCMAML is described in Algorithm 1. Starting from the randomly initialized model \( \theta \), we sample a few-shot task batch where each task poses an \( N \)-way classification (Line 6). Recall that we described an example of a task batch where \( T = 4 \) and \( N = 5 \) in Fig. 1. After we obtain \( \phi_{\tau} \) using \( D_{spt}(\tau) \) (Line 9), we aggregate five feature vectors using \( \phi_{\tau} \) and \( D_{qry}(\tau) \) (Lines 10–13), and then we compute the similarity score (Lines 14–16). After we repeat this procedure four times, we obtain the similarity scores of Task 1 to Task 4. Then, when we update the meta-parameter \( \theta \), we use calibration weights (Lines 17–18) to under-weight the ill-defined tasks (e.g., Task 4 in Fig. 1) and over-weight the well-defined tasks (e.g., Task 3 in Fig. 1).

Fortunately, in the complexity view, our task calibration does not add a large burden to the computation. Note that Algorithm 1 is based on the MAML algorithm, and the difference between TCMAML and MAML is on Lines 10–15 and Lines 17–18. (i.e., other operations are the same as MAML). Computing the average feature vectors for every class \( c \) is implemented in matrix form; thus, it does not result in additional \( \mathcal{O}(N) \) complexity. Computing the similarity score is also done by matrix multiplication. Moreover, in few-shot learning, \( N \) is typically small (e.g., 5), so the space complexity \( \mathcal{O}(N^2) \) can be ignored.

We can use any normalization function when computing \( w_{\tau} \). One of the common choices for normalization is the softmax function. The softmax uses exponential scaling (ES), general.
FIGURE 4. Overall pipeline of our algorithm. We generally used the meta-parameter $\theta$ as a Conv-4 backbone. The red color indicates the task calibration procedure in which we compute the similarity score ($s_{\cdot}$) and calibration weight ($w_{\cdot}$) for each task. The weighted sum of the task-specific losses is minimized.

however, linear scaling (LS) is also possible.

$$w_{\tau} = \begin{cases} \exp(-s_{\tau}/T) & (ES) \\ \sum_{\tau=1}^{T} \exp(-s_{\tau}/T) & (LS) \end{cases}$$

where $T$ is a scaling factor. ES needs to tune the scaling factor, although LS does not require any tuning. Unless $T$ is very large, ES imposes much larger weight variations than LS. LS may be more similar to the original update, i.e., equal weighting, than ES. In our experiments, we did some searching for $T$, and we empirically found that ES performed better than LS. Every result is based on ES by default. Refer to Section IV-C2 for the TCMAML with linear scaling.

D. HOW DOES TASK CALIBRATION WORK?

The example of a batch of tasks and their task calibration weights in Fig. 1 shows that for a difficult task, TCMAML imposes small weights to mitigate overconfident learning. To validate if the task calibration weighting strategy truly helps the calibration of a meta-learner, we investigated what happens during the meta-training of MAML. We sampled a meta-batch of size 4 ($T = 4$) from a training set and selected the most difficult task (with highest similarity score) and the easiest task (with lowest similarity score). We compared the prediction outcomes of the query set before and after the meta-update. For MAML, the meta-update corresponds to (2), as for TCMAML, it corresponds to (7).

In Fig. 5, we collected the prediction probabilities of the queries, sorted them in descending order, and then averaged over the queries: for 5-way $k$-shot learning the average prediction vector is 5-dimensional. Therefore, the leftmost value ($p_1$) on the $x$-axis of Fig. 5 is equivalent to the average confidence. Before the meta-update, the learner was adapted only to the support examples, so the confidence of the query predictions is low. After the meta-update, both MAML (After) and TCMAML (After TC) classify the queries more confidently. However, TCMAML’s behaviors on the two tasks are different. TCMAML obtained lower confidence than MAML on the difficult task, whereas on the easy task, TCMAML predicted with higher confidence. This implies that TCMAML learned the difficult task with a smaller step, which mitigated the overconfidence.

E. ALGORITHM-AGNOSTICISM

We emphasize that our task calibration approach can be easily extended to a metric-based approach. Unlike optimization-based models such as MAML, a metric-based model does not require a process of explicit adaptation to the task. Instead, the model calculates the distances between the query and support data in a non-parametric way. In this study, we use a prototypical network, i.e., ProtoNet [2], which is a high-performance metric-based few-shot learning method, and propose and apply our task-calibrated version, TCProtoNet.

Using the distance of each query to the support prototypes $p = [p_1, \ldots, p_N]^\top$, we collect the average distance vectors for each class, $d_{\phi, c} := \frac{1}{k} \sum_{q=1}^{k} d(x_q^{(r)}, p)$, where $(x_q^{(r)}, y_q^{(r)}) \in D_{\text{qry}}^{(r)}$ and $d(\cdot, p) \in \mathbb{R}_+^N$ hold. As in Section III-B, we assemble $N$-way $d_{\phi, c}$ vectors to form $D_{\tau}$. Meta-
update weights are computed based on the similarity score \( s_{\tau} = \| \mathbf{D}_\tau \mathbf{D}_\tau^\top \| \). We empirically demonstrate that TCPProtoNet is better calibrated than the vanilla ProtoNet.

In a similar manner, we extended the task calibration approach to several other algorithms, including Bayesian methods [17], [18] and convex optimization-based learning [36]. The details of the implementation for each algorithm are described in Appendix A. The empirical result implies that the task calibration is versatile and applicable to a range of few-shot learning algorithms.

IV. EXPERIMENTS AND DISCUSSION

A. EXPERIMENTAL DETAILS

We mainly used three data setups for the experiments: mini-ImageNet [35], CUB-200 [37], and dataset shift (mini-ImageNet → CUB-200).

mini-ImageNet dataset contains 100 classes and 600 images per class. We followed the standard protocol, which was initially provided by Ravi & Larochelle [4], by splitting the (train, validation, test) set into (64, 16, 20) non-overlapping classes.

CUB-200-2011 dataset is composed of 200 bird species classes that include 11,788 fine-grained images. According to Hilliard et al. [38], it is divided into (100, 50, 50) classes for the (train, validation, test) sets, respectively.

Thanks to the work they conducted on fair comparison in the FSC setup, most of our implementation details follow Chen et al. [29], both for 1-shot and 5-shot tasks. All accuracies reported are test accuracies, which were evaluated from 1000 tasks randomly drawn from the test set. By default, we used a Conv-4 backbone, CNN with four convolutional blocks [35], so as to not overestimate the performance by using deeper networks. Furthermore, all the MAML-based algorithms were trained with a first-order approximation of derivatives for the efficiency [7].

Dataset Shift. Additionally, we conducted dataset shift experiments [29], in which the train set was the entire mini-ImageNet dataset and the validation and test sets were from CUB-200. The shift mini-ImageNet → CUB-200 was implemented with a ResNet-18 backbone [39] for better convergence. Dataset shift is significantly more challenging than a typical FSC setting because the model has to predict the examples from distributions that are completely different from those of the train set. This is an intriguing situation from a practical perspective because for most real-world problems, it cannot be assumed that new tasks are sampled from the distribution which the model is familiar with.

Hyperparameters. In most experiments, we used a Conv-4 backbone, except for the dataset shift experiments. The Conv-4 architecture [35] stacks 4 blocks that are each comprised of (Convolution + BatchNorm + ReLU + MaxPool). During training, we used data augmentation (RandomResizedCrop, ColorJitter, RandomHorizontalFlip), and each image was resized to a resolution of 84 (224 in the dataset shift). For MAML and TCMAML, meta-training lasted 1600 epochs for 5-shot and 2400 epochs for 1-shot experiments, with 25 episodes per epoch. One episode sampled 4 tasks \( (T = 4) \) and each task sampled 15 queries per class \( (k' = 15) \). We used the Adam optimizer [40] with a learning rate of 5e-4. The learning rate in the inner-loop update was 1e-2, and the number of inner-loop steps was 5. In addition, computing weights for each task required a scaling factor, according to (8). \( T \) is a scaling hyperparameter that needs to be tuned, depending on the dataset, type of feature extractor, and the size of a matrix (number of classes), etc. Note that \( T \rightarrow \infty \) implies an equally weighted sum, leading to the original meta-update. For TCMAML, we searched \( T \) from

![FIGURE 5. Average prediction probabilities for 5 query images (1 per class) in a 5-way task, sorted in a descending order.](image-url)
TABLE 1. Calibration error (%) results for mini-ImageNet and CUB-200 experiments. A lower ECE or MCE value implies a better-calibrated model. The test accuracy is reported in Section IV-B4.

| Methods       | Backbone | mini-ImageNet | CUB-200         |
|---------------|----------|---------------|-----------------|
|               |          | 5-Way 1-Shot | 5-Way 5-Shot    | 5-Way 1-Shot | 5-Way 5-Shot |
|               |          | ECE  MCE     | ECE  MCE       | ECE  MCE     | ECE  MCE     |
| MAML [7]      | Conv-4   | 4.04 8.77    | 1.46 3.98      | 2.22 2.94    | 2.39 3.66    |
| TCMAAML       |          | 0.80 5.80    | 1.62 4.09      | 1.40 2.74    | 2.27 3.36    |
| ProtoNet [2]  | Conv-4   | 1.26 4.53    | 0.70 1.43      | 3.88 6.31    | 0.46 4.44    |
| TCProtoNet    |          | 0.89 3.00    | 0.55 1.27      | 2.87 5.37    | 0.34 0.85    |
| ABML [18]     | Conv-4   | 1.99 7.14    | 2.97 4.93      | 3.34 5.00    | 2.29 4.59    |
| TCAEML        |          | 1.14 7.00    | 0.95 1.69      | 3.09 8.41    | 2.20 4.45    |
| BMAML [17]    | Conv-4   | 3.97 8.02    | 7.40 9.60      | 10.04 14.34  | 12.92 16.94  |
| TCBMAML       |          | 2.06 4.07    | 4.56 6.51      | 9.31 12.76   | 11.59 15.59  |
| MetaOptNet [36] | ResNet-12 | 24.75 37.08  | 12.21 16.53    | 34.76 44.29  | 15.62 24.41  |
| TCMetaOptNet  |          | 23.56 34.23  | 10.23 13.06    | 27.22 34.79  | 8.49 13.83   |

1 We used MetaOptNet-SVM which uses a multi-class support vector machine as a convex base learner.

\[ \frac{1}{2}, 1, 2 \]. Refer to Appendix A for the hyperparameters of other algorithms.

**Baselines.** Our work aims to decrease the misconfidence that occurs in various few-shot learning algorithms. However, most few-shot learning studies have focused on ways to increase the few-shot learning accuracy, so they have evolved methods by using a sophisticated data augmentation technique [41]–[43], evaluating in a transductive manner [44], [45], or utilizing a self-supervised learning strategy [46], [47]. Due to analytical complexity, we had to choose representative methods for our experiments, especially those for which open-source code was available. Among the chosen methods, MetaOptNet [36] shows a few-shot performance comparable to that of the state-of-the-art algorithms in an inductive setting with a simple architecture. In addition, there are few works that have considered miscalibration in a few-shot learning framework, which makes it difficult to conduct comparative analysis.

### B. CALIBRATION RESULTS

#### 1) Evaluation

Our main concern is calibration error, rather than classification performance. The model is well-calibrated when it makes decisions based on the predicted uncertainty. The correctness of a perfectly calibrated model will match the confidence level (the largest value of the softmax output). For example, the model is miscalibrated or overconfident when the confidence is 0.8 on average, but only 70 out of 100 samples are correct. The quantified measures for miscalibration include expected calibration error (ECE) and maximum calibration error (MCE),

\[
ECE = \frac{1}{n} \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)| \quad (9)
\]

\[
MCE = \max_{m \in \{1, \ldots, M\}} \frac{|\text{acc}(B_m) - \text{conf}(B_m)|}{|B_m|} \quad (10)
\]

where \( \text{acc}(B_m) \) and \( \text{conf}(B_m) \), respectively, are the empirical accuracy and average confidence for the set \( B_m \). Partitioning the range \([0, 1]\) into \( M \) segments of identical lengths, we collect each sample’s prediction confidence, which is bounded by 0 and 1, and assign the sample to one of the \( M \) partitions according to the confidence level. Here, \( B_m \), referred to as bin, is a subset of the samples such that their confidence falls into the \( m \)-th partition. Thus, high calibration error implies that the model does not have confidence for the samples as the level of the model’s classification ability; high ECE means misconfidence in average, and high MCE means misconfidence for certain subset of samples. Refer to [48] for a detailed derivation.

We chose ECE and MCE for the evaluation metric for the purpose of finding a model that has a confidence that matches its predictions. Although the model shows poor classification performance on the target examples, we confirm that the model is well-calibrated if the calibration error is low: the model’s predictions and its predictive confidence are aligned. There is another view of uncertainty estimation that considers model uncertainty. This is often evaluated using proper scoring rules, such as NLL (negative log-likelihood) and the Brier score [49]. However, this kind of metric is related to the model uncertainty, which implies that a model with low prediction error is likely to have a low uncertainty estimate [13]. In FSC, classification accuracy is most of the time very low, and it largely differs depending on the meta-learning algorithm and backbone architecture. Therefore, we focus on lowering the gap between this accuracy and confidence, instead of reducing the prediction error.

#### 2) Calibration Improvements

In addition to MAML [7] and ProtoNet [2], we also compared with ABML [18], BMAML [17], and MetaOptNet [36]. ABML (amortized Bayesian meta-learning) and BMAML (Bayesian MAML) are Bayesian methods for few-shot learning, and both are based on the MAML algorithm. MetaOptNet is an optimization-based model, where in the adaptation step a linear classifier trains with the embeddings of the support set and returns the classification score on the query
TABLE 2. Calibration results (%) for dataset shift experiments. ABML and BMAML are not implemented in the dataset shift experiment, due to their inefficiency. Refer to Section IV-D and Appendix A regarding the reimplementation of ABML and BMAML.

| Methods      | Backbone      | mini-ImageNet → CUB-200     |
|--------------|---------------|-----------------------------|
|              |               | S-Way 1-Shot | ECE | MCE | S-Way 5-Shot | ECE | MCE |
| MAML [7]     | ResNet-18     | 18.02         | 35.80 | 18.28 | 25.27      |
| TCMAML       |               | 15.06         | 27.88 | 17.53 | 24.65      |
| ProtoNet [2] | ResNet-18     | 4.17          | 6.61  | 0.50  | 1.72       |
| TCPProtoNet  |               | 3.47          | 4.64  | 0.49  | 1.30       |
| MetaOptNet [36] | ResNet-18 | 12.09         | 22.80 | 6.39  | 8.66       |
| TCMetaOptNet | ResNet-12     | 9.31          | 17.81 | 4.36  | 5.56       |

set.

Table 1 summarizes the error rate on the mini-ImageNet and CUB-200 experiments. In most experiments, the task calibration method (TC-) exhibited a significantly decreased error rate than its counterpart; for example, TCPProtoNet outperformed ProtoNet. Even the Bayesian way, when equipped with task calibration (TCABML and TCBMAML), further reduced the miscalibration rate.

It is well known that Bayesian methods are capable of calibration. As for ABML [18], if the parameters sampled from the approximate posterior effectively classify the query set, the posterior is updated in the direction of producing point estimators. Otherwise, the approximate posterior produces a diverse parameter set, which results in uncertain predictions. Although our method does not deal with parameter distribution, the idea of mitigating overconfident learning on an uncertain task is similar. Whereas the Bayesian methods implicitly lead to calibration via parameter uncertainty, our task calibration directly targets the support-query discrepancy. Notably, in almost every experiment, the task calibration method yielded consistent accuracy without incurring a decrease in classification performance (refer to Section IV-B4). In contrast, as [18] denoted, ABML often produced poor test accuracy.

We could observe that different learning strategies highly influence the model’s calibration abilities, and our method orthogonally improves upon the diverse learning algorithms. Specifically, ProtoNet and TCPProtoNet have better calibration ability than other algorithms. ProtoNet updates the parameters using the distance between the support prototypes and the query set. Therefore, ProtoNet actually warps the embedding space toward reducing the support-query discrepancy.

3) Dataset Shift Results

Table 2 exhibits the calibration results for the dataset shift experiment. Although some experiments produced large miscalibration rates, task calibration consistently reduced them. From the perspective of accuracy, TCMAML is much worse than TCPProtoNet and TCMetaOptNet (refer also to Table 3). The general belief that a network with lower training error tends to be overconfident [48] is not always true. In some cases, sufficient amount of prediction capability is needed for better calibration.

4) Classification Accuracy

Calibrated models generally degrade classification ability for the sake of accurate risk prediction. However, our method does not degrade the classification accuracy. In Table 3, we display the test accuracy for every experiment. ProtoNet-based methods outperform MAML-based methods and do so by large margin in the dataset shift experiments. MetaOptNet is implemented using the ResNet-12 backbone, which is the reason for the high accuracy in mini-ImageNet and CUB-200 experiments.

Unfortunately, TCMetaOptNet shows an accuracy drop beyond the confidence interval in the CUB-200 and the dataset shift experiments. We suppose this happens because MetaOptNet itself has high miscalibration error and TCMetaOptNet reduces the error significantly (refer to Table 1 and Table 2). Although largely improving the calibration can possibly cause a drop in accuracy, we should use a calibrated model when we need a reliable model even if we give up some accuracy.

C. ABSTRACTION STUDIES

1) Temperature Scaling: A Classic Calibration Approach

Temperature scaling (TS) [48] is a calibration method that is widely used in general classification, and it learns a temperature parameter to control confidence. It functions as post-processing on the validation set. However, in a few-shot learning setup, the train, validation, and test set all have non-overlapped classes, and the unseen classes are given in future tasks. Existing calibration methods such as TS find the best parameter for known classes, which is not applicable in FSC.

This is why in Fig. 6, +TS produced somewhat random results; a few results are good, but the rest are extremely poor. Note that +TS has the same accuracy as its counterpart, because TS is a post-processing method on the validation set and does not change the prediction label (it changes only the smoothness of the predictive distribution).

2) TCMAML with Linear Scaling

In every experiment so far, we computed \( w_\tau \) using a softmax function over the similarity scores. While the softmax function uses an exponential scaling (ES), normalizing by linear scaling (LS) is also possible. In this case, we do not need a scaling factor.

\[
    w_\tau = \frac{1/s_\tau}{\sum_{\tau=1}^{T} 1/s_\tau} \tag{LS}
\]

Table 4 exhibits the results of TCMAML with linear scaling. However, in most experiments, TCMAML (ES) is comparable with or better than TCMAML (LS). Depending on the similarity score values, LS may have small weight variations across tasks and produce results similar to those of the vanilla MAML.
3) Inter-Class Support Embeddings

In Section III-C, we claimed that $\bar{h}_{\phi_e}$ is determined by the relation between the support set and the query set, because the support-adapted parameter $\phi_e$ embeds queries. To further verify this, we experimented with a different embedding method. Here, we used only the support set for computing inter-class similarity scores (TCMAML-SPT).

Table 4 summarizes the results. As we expected, using only the support set to compute inter-class similarity did not capture the distribution mismatch. Therefore, TCMAML-SPT did not calibrate MAML effectively and showed a few poor results. The support set was well-classified by $\phi_e$ due to the adaptation process, and the support set did not solely reflect the distribution mismatch. Therefore, TCMAML-SPT incorrectly estimated the task difficulty.

4) Reverse TCMAML

What would happen if TCMAML computes $w_\tau$ as a normalized +9σ value; large weight on the ill-defined tasks and small weight on the well-defined tasks? We hypothesized

**TABLE 3.** 5-way k-shot test classification accuracy (%) with 95% confidence interval. The experimental details follow those of Table 1 and Table 2.

| Methods         | mini-ImageNet | CUB-200 | mini-ImageNet → CUB-200 |
|-----------------|---------------|---------|-------------------------|
|                 | 1-Shot 5-Shot | 1-Shot 5-Shot | 1-Shot 5-Shot |
| MAML [7]        | 46.92±0.63   | 63.18±0.59 | 56.76±0.78  | 75.62±0.58 | 34.24±0.53  | 49.34±0.57 |
| TCMAML          | 47.20±0.63   | 62.96±0.59 | 56.55±0.75  | 76.03±0.57 | 33.48±0.52  | 50.44±0.56 |
| ProtoNet [2]    | 48.30±0.62   | 67.61±0.54 | 57.03±0.68  | 77.79±0.49 | 44.20±0.78  | 64.97±0.69 |
| TCPProtoNet     | 49.52±0.63   | 67.12±0.55 | 56.91±0.69  | 77.23±0.50 | 45.04±0.80  | 64.08±0.71 |
| ABLM [18]       | 42.34±0.58   | 60.00±0.53 | 54.99±0.70  | 69.59±0.56 | -            | -            |
| TCBABML         | 43.20±0.58   | 60.26±0.53 | 53.99±0.69  | 68.15±0.57 | -            | -            |
| BMAML [17]      | 45.12±0.61   | 60.81±0.58 | 57.79±0.73  | 73.19±0.57 | -            | -            |
| TCBMAML         | 45.52±0.63   | 60.27±0.58 | 57.29±0.72  | 72.92±0.60 | -            | -            |
| MetaOptNet [36] | 59.44±0.63   | 76.78±0.48 | 73.87±0.65  | 88.77±0.35 | 46.91±0.58  | 67.88±0.55 |
| TCMetaOptNet    | 59.00±0.64   | 76.22±0.50 | 68.59±0.69  | 84.71±0.43 | 44.57±0.57  | 64.49±0.54 |

**TABLE 4.** ECE (%) and MCE (%) comparisons between MAML, TCMAML with exponential scaling (ES), TCMAML with linear scaling (LS, Section IV-C2), and TCMAML-SPT (Section IV-C3).

| Methods         | mini-ImageNet | CUB-200 | mini-ImageNet → CUB-200 |
|-----------------|---------------|---------|-------------------------|
|                 | 5-Way 1-Shot  | 5-Way 5-Shot | 5-Way 1-Shot  | 5-Way 5-Shot | 5-Way 1-Shot  | 5-Way 5-Shot |
| MAML [7]        | 0.80±0.50     | 1.62±0.40 | 2.22±2.94   | 3.23±3.66   | 18.02±35.80   | 18.28±25.27  |
| TCMAML (ES)     | 1.25±0.96     | 4.66±4.90 | 2.63±3.34   | 2.67±3.17   | 16.46±39.46   | 18.24±24.39  |
| TCMAML (LS)     | 3.13±0.25     | 6.25±5.23 | 2.58±3.53   | 2.36±3.67   | 17.22±40.91   | 22.05±30.49  |
that this method would degrade the calibration in the reverse way of TCMAML. However, Table 5 shows an unexpected result: TCMAML-R had a still lower miscalibration rate than MAML in 1-shot tasks. Our conjecture is that assigning larger weights to difficult tasks moves the meta-parameter toward better solving these tasks. Then, those tasks will have smaller similarity scores than previous iterations and consequently get smaller weights. This happens alternatively and produces inconsistent calibration results.

5) Invariance to Backbone and Resolution
Table 6 shows the invariance of our method to the change in backbone architecture and the resolution (image size). Note that the convolutional backbone (e.g., Conv-4 and Conv-6) matches with the image size 84 and that the residual backbone (e.g., ResNet-10 and ResNet-18) matches with the image size 224. Recall that in Section IV-B2, we only presented the results on the Conv-4 backbone with the image size 84, except for MetaOptNet and TCMetaOptNet. The results in Table 6 demonstrate that TCMAML is consistently better than MAML for Conv-6, ResNet-10, and ResNet-18. In addition, there is a tendency for deeper backbones to have a higher miscalibration rate, and TCMAML is more likely to improve in these cases.

### D. COMPARISON WITH BAYESIAN METHOD

Bayesian meta-learning models are well known to capture uncertainty and calibrate the model, but task calibration can still improve them. In Table 1, we showed that task calibration on both ABML and BMAML further decreased the calibration error rate. In this Section, we compare our method with a Bayesian method, ABML [18]. ABML approximates the posterior distribution by $K$ SGD steps and obtains variational parameters $\{\mu^{(K)}, \sigma^{2(K)}\}$. This variational distribution should capture the uncertainty, that is, if the model is uncertain, it should produce a large standard deviation parameter. Then, the prediction probability can be largely different depending on the posterior samples,

\[
\{\mu^{(K)}, \sigma^{2(K)}\} = \text{SGD}_K(D_{\text{opt}}^{(\tau)}, \theta, \theta) \tag{12}
\]

\[
g_{\phi_T}(D_{\text{opt}}^{(\tau)}) = \mathcal{N}(\phi_T; \mu^{(K)}, \sigma^{2(K)}) \tag{13}
\]

To determine if ABML really captures uncertainty, we tracked $\sigma^{2(K)}$ values during the training of the ABML. However, contrary to our expectations, the variational standard deviation parameters were nearly 0 (see Fig. 7 (a)). This implies that posterior sampling produces similar samples and outputs, i.e., reducing the model to a point estimator. Moreover, noting that ABML sees 100 tasks per epoch, $\sigma^{2(K)}$ values hardly differed across 100 tasks. This observation implies that ABML does not consider the uncertainty of each task but rather captures global model uncertainty.

However, TCMAML shows a large difference in similarity scores between 100 tasks (see Fig. 7 (b)). We can interpret that TCMAML captures the uncertainty of each task in a more sophisticated manner than ABML.

Furthermore, we did not implement ABML in the dataset shift experiments due to its inefficient memory and computation. ABML is a Bayesian method, and this implies that ABML is trained by marginalizing over model parameters. During implementation, this requires a significant computational overhead because it should hold several models and receive their gradients in parallel. During the meta-training of ABML, 5 ensemble networks work in parallel, and during the meta-testing, 10 ensemble networks are employed. In Fig. 7 (c), TCMAML and ABML show an enormous difference between their GPU memory usage, even though they are both MAML-based methods. We also find that ABML takes a long time to compute during the meta-train phase. Because of ABML’s inefficiency, we could not apply it to a ResNet-18 model, which is the backbone for the dataset shift experiments. ABML with ResNet-18 exceeds the capacity of the single GPU we own, and also it requires an excessive amount of time to converge.

### E. CORRUPTED META-LEARNING

Our task calibration method is robust against data corruption. Meta-training with corrupted data is challenging to the meta-learner because the corruption introduces large noise that impedes the accumulation and transfer of meta-knowledge. Nevertheless, robustness is crucial because the model should still learn meta-knowledge, even on those tasks, to be transferred to the target tasks. Uncertainty-aware model as task calibration can prevent noisy learning and achieve robustness [50].

To this end, we experimented by training MAML and ProtoNet under data corruption. We manipulated the task sampling process in the meta-train phase. Specifically, we corrupted the tasks to include random Gaussian images using some fixed probability $p$. That is, a corrupted task consisted of 5 clean support images from the mini-ImageNet train set and 15 corrupted query images that had pixel values from $\mathcal{N}(0, 1)$.

We show in Table 7 that MAML is already robust against this attack due to informative feature reuse [51] and overfitting to the uncorrupted images; thus, there was no improvement with TCMAML. However, a metric-based model such as ProtoNet is fragile because this type of model learns directly from matching the query examples to the support set. If the support and query set are heterogeneous, ProtoNet is easily damaged by unwanted updates.

| Table 5. Comparisons of the calibration error rate (%) between TCMAML and TCMAML-R (Reverse TCMAML, Section IV-C4). |
|---|
| Methods | mini-ImageNet | 5-Way 1-Shot | 5-Way 2-Shot |
| MAML [7] | 4.04 | 8.37 | 1.46 | 3.98 |
| TCMAML | 0.80 | 5.80 | 1.62 | 4.09 |
| TCMAML-R | 3.81 | 8.35 | 1.97 | 5.61 |

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3176090, IEEE Access
TABLE 6. mini-ImageNet and CUB-200 test accuracy (%) and calibration errors (%) of the MAML algorithm for different backbone architectures and image sizes. TC (✓) denotes that the task-calibrated meta-update is applied.

| Backbone | Image size | TC | 5-Way 1-Shot | 5-Way 5-Shot |
|----------|------------|----|--------------|--------------|
|          |            |    | mini-ImageNet |              |
|          |            |    | Acc  | ECE  | MCE | Acc  | ECE  | MCE |
| Conv-6   | 84         | ✓  | 52.75±0.71  | 3.28 | 5.42 | 65.78±0.57 | 3.57 | 5.30 |
| ResNet-10| 224        | ✓  | 54.20±0.71  | 8.94 | 16.61| 65.88±0.61 | 11.13| 16.24|
| ResNet-18| 224        | ✓  | 51.80±0.68  | 7.35 | 14.54| 66.41±0.58 | 11.02| 15.36|

| Backbone | Image size | TC | 5-Way 1-Shot | 5-Way 5-Shot |
|----------|------------|----|--------------|--------------|
|          |            |    | CUB-200      |              |
|          |            |    | Acc  | ECE  | MCE | Acc  | ECE  | MCE |
| Conv-6   | 84         | ✓  | 66.29±0.75  | 2.18 | 3.96 | 78.68±0.54 | 1.36 | 2.05 |
| ResNet-10| 224        | ✓  | 71.06±0.73  | 6.88 | 10.15| 81.89±0.51 | 6.03 | 11.42|
| ResNet-18| 224        | ✓  | 70.68±0.81  | 9.50 | 13.87| 80.87±0.58 | 8.18 | 13.32|

Therefore, we observed that TCProtoNet gained considerable robustness. In Table 7, when 25% of tasks were corrupted (p = 0.25), the ProtoNet performance dropped to 61.38%. Not surprisingly, 50% corruption (p = 0.5) led to a further decrease, which was 24.01%p lower than when trained without corruption. TCProtoNet managed to exceed its counterpart, 63.45% and 48.65% for 25% and 50% corruption, respectively.

V. CONCLUSION

In this study, we asserted that the few-shot image classification problems should be examined by carefully estimat-
ing task difficulty. We detected that the FSC contains a large discrepancy between the support and the query set, although typical meta-learning algorithms do not consider this mismatch. During meta-training, the distribution mismatch causes the model to have wrong confidence, which causes a calibration problem. Therefore, we modified the meta-update rule based on similarity scores, and a simple experiment verified that giving different weights to the tasks actually affects the model’s prediction confidence, which consequently helps calibration. The calibration results verified reliable decision-making without degrading the model’s ability to classify. Although our method modifies the meta-training process, it would be helpful to develop a post-processing method in the future that could enhance calibration in the meta-test phase.

APPENDIX A REIMPLEMENTATION OF ABML, BMAML, AND METAOPTNET

A. ABML

ABML [18] is a Bayesian approach that estimates parameter uncertainty in meta-learning. We reimplemented the ABML algorithm because 1) open-source code was not available, 2) the authors only reported on the mini-ImageNet 5-way 1-shot experiment result, and 3) our experimental setup was slightly different from theirs. Therefore, we implemented ABML based on our setup. First, we sampled 4 tasks within an episode, as we did in the TCMAML experiments. Second, the authors of [18] did not use data augmentation, but we used the same data loaders as in other experiments of ours. This includes the same data augmentation scheme. Third, we used a first-order approximation of derivatives as TCMAML. For the other setups, such as the number of ensemble networks to train and validate, learning rates, and KL-divergence reweighting factor, we followed the original study (Table 8).

We implemented TCABML in a manner similar to that of TCMAML. After adaptation to a task, the model learns task-specific parameters, which are variational parameters for the approximate posterior distribution. Given these parameters, it samples network parameters that are used to predict the query examples; it samples $M$ times. Here, we can obtain feature vectors, as we did in computing the class-wise features. Then, we compute the similarity scores with $M$ different samples, and take the largest score out of the $M$ scores. We repeat this process for every task in a meta-batch, and then compute the weight in the meta-update. We use a scaling factor $T = 1/2$ for mini-ImageNet and $T = 2$ for CUB-200.

B. BMAML

BMAML [17] is also a Bayesian method that is based on the MAML algorithm. It also holds ensemble networks (they are denoted as “particles”) that are each updated in SVGD steps. We did not implement BMAML in the dataset shift experiment for the same reason on efficiency. As with TCABML, TCBMAML was implemented by obtaining the feature vectors from each particle and taking the largest value from 5 training particles. Most setups followed those of ABML and TCBABML, e.g., 4 tasks per episode, data augmentation, and first-order approximation. We used a scaling factor $T = 1$ for every experiment.

C. METAOPTNET

Recent studies have achieved excellent few-shot results by using a linear classifier that solves a convex optimization problem; the authors of [36] named it MetaOptNet. Specifically, we implemented MetaOptNet-SVM, which uses a multi-class SVM as the linear classifier. MetaOptNet-SVM learns the parameters of the SVM by optimizing on the embeddings of the support examples. The learned SVM parameters produce score logits for each query example. TCMetaOptNet uses these logit vectors when computing the class-wise average vectors, as we did in the TCProtoNet implementation.

We followed the protocol from [36], which used 15 support examples in meta-training, and then evaluated using 1-shot and 5-shot tasks in meta-testing. We trained for 30 epochs, where 1 epoch contained 1000 episodes. We set the scaling factor as $T = 0.1$. MetaOptNet produces highly uncorrelated logit vectors, due to the optimization process of the SVM. Therefore, we used a small scaling factor to create the weight variations between tasks. Also, we used the ResNet-12 architecture for every experiment, including the dataset shift.

APPENDIX B RELIABILITY DIAGRAMS

Reliability diagrams [52] visually present model calibration and plot empirical accuracy versus average prediction confidence, which is the largest value of the output distribution. Fig. 8 describes the reliability diagrams by comparing between MAML, ABML, and TCMAML. We can observe that TCMAML has its bars closer to the $y = x$ line than MAML and ABML, which implies more reliable predictions. Furthermore, the CUB experiments show slightly different results. MAML is diffident in its predictions, with the empirical accuracy higher than the confidence. A model that is not sufficiently confident can cause another type of problem. As opposed to overconfidence, low confidence often occurs when there are ambiguous images having low accuracy. This situation is vulnerable to adversarial attack or the perturbation of images [53], readily available to decrease overall accuracy due to the model’s low confidence. In this case, a model that can accurately tell easy tasks apart is needed. TC-MAML again succeeds in calibration with enough confidence in such tasks.

|Hyperparameters of ABML [18] reimplementation.|
|---|
|Inner learning rate | 0.01 |
|Outer learning rate | 0.001 |
|Gradient steps for $q$ (variational distribution) | 5 |
|Number of samples to train task-specific parameters | 5 |
|Number of samples to validate meta-parameters | 5 |
|Number of samples to compute expectation for evaluation | 10 |
|KL-divergence reweighting factor | 0.1 |
S. Kim, S-Y. Yun: Calibration of Few-Shot Classification Tasks: Mitigating Misconfidence from Distribution Mismatch

FIGURE 8. Reliability diagrams from MAML (left column), ABML (middle), and TCMAML (right) classification results. All experiments are 1-shot, and from the top row, the datasets are mini-ImageNet, CUB-200, and mini-ImageNet → CUB-200. The red bars indicate the empirical accuracy of the samples whose prediction confidence falls into a specific interval. A perfectly calibrated model should have its bars aligned to $y = x$ line (dashed). Deviation from the line is highlighted in blue if average confidence is higher than accuracy; overconfident, and dark if accuracy is higher; diffident. Note that the confidence is always above 0.2 because there are 5 classes. Best seen in color.

REFERENCES

[1] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in ICML deep learning workshop, vol. 2, 2015.
[2] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in Advances in Neural Information Processing Systems, pp. 4077–4087, 2017.
[3] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1199–1208, 2018.
[4] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in 5th International Conference on Learning Representations, 2017.
[5] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, “Learning to learn by gradient descent by gradient descent,” in Advances in Neural Information Processing Systems, pp. 3981–3989, 2016.
[6] D. Maclaurin, D. Duvenaud, and R. Adams, “Gradient-based hyperparameter optimization through reversible learning,” in Proceedings of the 32nd International Conference on Machine Learning, vol. 37, pp. 2113–2122, 2015.
[7] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1126–1135, JMLR. org, 2017.
[8] T. Helldin, G. Falkman, M. Riveiro, and S. Davidsson, “Presenting system uncertainty in automotive UIs for supporting trust calibration in autonomous driving,” in Proceedings of the 5th international conference on automotive user interfaces and interactive vehicular applications, pp. 210–217, 2013.
machine learning in medicine,” JAMA, vol. 318, no. 6, pp. 517–518, 2017.
[10] J. Platt et al., “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” Advances in large margin classifiers, vol. 10, no. 3, pp. 61–74, 1999.
[11] B. Zadrozny and C. Elkan, “Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers,” in ICML, vol. 1, pp. 609–616, Citeseer, 2001.
[12] B. Zadrozny and C. Elkan, “Transforming classifier scores into accurate multiclass probability estimates,” in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 694–699, 2002.
[13] J. Snoek, Y. Ovadia, E. Fertig, B. Lakshminarayanan, S. Nowozin, D. Sculley, J. Dillon, J. Ren, and Z. Nado, “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift,” in Advances in Neural Information Processing Systems, pp. 13969–13980, 2019.
[14] Z. Xu, L. Cao, and X. Chen, “Meta-learning via weighted gradient update,” IEEE Access, vol. 7, pp. 108864–110855, 2019.
[15] L. Fei-Fei et al., “A Bayesian approach to unsupervised one-shot learning of object categories,” in Proceedings Ninth IEEE International Conference on Computer Vision, pp. 1134–1141, IEEE, 2003.
[16] J. Zhang, C. Zhao, B. Ni, M. Xu, and X. Yang, “Variational few-shot learning,” in Proceedings of the IEEE International Conference on Computer Vision, pp. 1685–1694, 2019.
[17] T. Kim, J. Yoon, O. Dia, S. Kim, Y. Bengio, and S. Ahn, “Bayesian model-agnostic meta-learning,” in Advances in Neural Information Processing Systems, pp. 7343–7353, 2018.
[18] S. Ravi and A. Beatson, “Amortized Bayesian meta-learning,” in 7th International Conference on Learning Representations, 2019.
[19] Y. Zou and X. Lu, “Gradient-em bayesian meta-learning,” Advances in Neural Information Processing Systems, vol. 33, pp. 20865–20875, 2020.
[20] Z. Sun, J. Wu, X. Li, W. Yang, and J.-H. Xue, “Amortized bayesian prototype meta-learning: A new probabilistic meta-learning approach to few-shot image classification,” in International Conference on Artificial Intelligence and Statistics, pp. 1414–1422, PMLR, 2021.
[21] A. Ghadirzadeh, X. Chen, P. Poklakur, C. Finn, M. Björkman, and D. Krälig, “Bayesian meta-learning for few-shot policy adaptation across robotic platforms,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1274–1280, IEEE, 2021.
[22] M. Abdar, F. Pourpanah, S. Hussain, D. Rezaazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al., “A view of uncertainty quantification in deep learning: Techniques, applications and challenges,” Information Fusion, vol. 76, pp. 243–297, 2021.
[23] C. Finn, K. Xu, and S. Levine, “Probabilistic model-agnostic meta-learning,” in Advances in Neural Information Processing Systems, pp. 9516–9527, 2018.
[24] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. E. Turner, “Meta-learning probabilistic inference for prediction,” in 7th International Conference on Learning Representations, 2019.
[25] M. Garcelo, N. Schwarzenbach, D. Violato, D. J. Rezende, S. Eslami, and Y. W. Teh, “Neural processes,” ICML workshop on Theoretical Foundations and Applications of Deep Generative Models, 2018.
[26] G. Singh, J. Yoon, Y. Son, and S. Ahn, “Sequential neural processes,” Advances in Neural Information Processing Systems, vol. 32, 2019.
[27] Z. Ye and L. Yao, “Contrastive conditional neural processes,” arXiv preprint arXiv:2007.03978, 2022.
[28] A. Malinin and M. Gales, “Predictive uncertainty estimation via prior networks,” in Advances in Neural Information Processing Systems, pp. 7047–7058, 2018.
[29] W. Chen, Y. Liu, Z. Kira, Y. F. Wang, and J. Huang, “A closer look at few-shot classification,” in 7th International Conference on Learning Representations, 2019.
[30] M. Sensou, L. Kaplan, and M. Kandemir, “Evidential deep learning to quantify classification uncertainty,” in Advances in Neural Information Processing Systems, pp. 3179–3189, 2018.
[31] O. Bohdai, Y. Yang, and T. Hospedales, “Meta-calibration: Meta-learning of model calibration using differentiable expected calibration error,” arXiv preprint arXiv:2106.09613, 2021.
[32] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive contrastive learning of visual representations,” in International conference on machine learning, pp. 1597–1607, PMLR, 2020.
[33] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 9729–9738, 2020.
[34] T. Wang and P. Isola, “Understanding contrastive representation learning through alignment and uniformity on the hypersphere,” in International Conference on Machine Learning, pp. 9929–9939, PMLR, 2020.
[35] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., “Matching networks for one shot learning,” in Advances in Neural Information Processing Systems, pp. 3630–3638, 2016.
[36] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, “Meta-learning with differentiable convex optimization,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10657–10665, 2019.
[37] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, “The Caltech-UCSD birds-200-2011 dataset,” 2011.
[38] N. Hilliard, L. Phillips, S. Howland, A. Yankov, C. D. Corley, and N. O. Hodas, “Few-shot learning with metric-agnostic conditional embeddings,” arXiv preprint arXiv:1802.04376, 2018.
[39] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.
[40] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in International Conference on Learning Representations, 2015.
[41] P. Mangla, N. Kumari, A. Sinha, M. Singh, B. Krishnamurthy, and V. N. Balasubramanian, “Charting the right manifold: Manifold mixup for few-shot learning,” in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2218–2227, 2020.
[42] J. Liu, F. Chao, and C.-M. Lin, “Task augmentation by rotating for meta-learning,” arXiv preprint arXiv:2003.00804, 2020.
[43] H. Zhang, Z. Cao, Z. Yan, and C. Zhang, “Sill-net: Feature augmentation with separated illumination representation,” arXiv preprint arXiv:2102.03539, 2021.
[44] S. M. Kye, H. B. Lee, H. Kim, and S. J. Hwang, “Meta-learned confidence for few-shot learning,” arXiv preprint arXiv:2002.12017, 2020.
[45] M. Lichtenstein, P. Satgireti, R. Feris, R. Giryes, and L. Karlinsky, “Tafsil: Task-adaptive feature sub-space learning for few-shot classification,” in European Conference on Computer Vision, pp. 522–539, Springer, 2020.
[46] J. Rajasegaran, S. Khan, M. Hayat, F. S. Khan, and M. Shah, “Self-supervised knowledge distillation for few-shot learning,” arXiv preprint arXiv:2006.09785, 2020.
[47] D. Chen, Y. Chen, Y. Li, F. Mao, Y. He, and H. Xue, “Self-supervised learning for few-shot image classification,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1745–1749, IEEE, 2021.
[48] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1321–1330, JMLR. org, 2017.
[49] G. W. Brier et al., “Verification of forecasts expressed in terms of probability,” Monthly weather review, vol. 78, no. 1, pp. 1–3, 1950.
[50] J. Snell and R. Zemel, “Bayesian few-shot classification with one-vs-each poly-gamma augmented gaussian processes,” in International Conference on Learning Representations, 2021.
[51] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, “Rapid learning or feature reuse? Towards understanding the effectiveness of MAML,” in International Conference on Learning Representations, 2019.
[52] A. Niculescu-Mizil and R. Caruana, “Predicting good probabilities with the logistic regression classifier,” in Proceedings of the 22nd International Conference on Machine learning, vol. 119, pp. 562–569, ACM, 2005.
[53] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, “Adversarial attacks and defences: A survey,” arXiv preprint arXiv:1810.00069, 2018.
SUNGYUN KIM was born in Seoul, South Korea in 1996. He received the B.S. degrees in chemical and biomolecular engineering and industrial and systems engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2019, and the M.S. degree in artificial intelligence from Kim Jaechul Graduate School of Artificial Intelligence, KAIST, Seoul, South Korea, in 2021. His research interests include machine learning, unsupervised deep learning, self-supervised learning, few-shot learning, and computer vision.

SE-YOUNG YUN is an Associate Professor with Kim Jaechul Graduate School of Artificial Intelligence, KAIST, South Korea. He received the B.S. and Ph.D. degrees in Electrical Engineering from KAIST, in 2006 and 2012. He was a Post-Doc Researcher with KTH, Sweden, from 2013 to 2014, MSR-INRIA, France, from 2014 to 2015, and LANL, New Mexico, US, from 2016 to 2017. From 2015 to 2016, he was a Visiting Researcher with Microsoft Research, Cambridge, UK. He was a recipient of ACM MobiHoc Best Paper Award in 2013, and NeurIPS Outstanding Reviewer Award in 2016.