An ensemble of epoch-wise empirical bayes for few-shot learning

Yaoyao LIU

Bernt SCHIELE

Qianru SUN
Singapore Management University, qianrusun@smu.edu.sg

Citation
1

An Ensemble of Epoch-wise Empirical Bayes
for Few-shot Learning

Yaoyao Liu[1], Bernt Schiele[1], and Qianru Sun[2]

{yaoyao.liu, schiele, qsun}@mpi-inf.mpg.de   qianrusun@smu.edu.sg

[1]Max Planck Institute for Informatics, Saarland Informatics Campus
[2] School of Information Systems, Singapore Management University

Abstract. Few-shot learning aims to train efficient predictive models with a few examples. The lack of training data leads to poor models that perform high-variance or low-confidence predictions. In this paper, we propose to meta-learn the ensemble of epoch-wise empirical Bayes models (E^3BM) to achieve robust predictions. "Epoch-wise" means that each training epoch has a Bayes model whose parameters are specifically learned and deployed. "Empirical" means that the hyperparameters, e.g., used for learning and ensembling the epoch-wise models, are generated by hyperprior learners conditional on task-specific data. We introduce four kinds of hyperprior learners by considering inductive *vs.* transductive, and epoch-dependent *vs.* epoch-independent, in the paradigm of meta-learning. We conduct extensive experiments for five-class few-shot tasks on three challenging benchmarks: *mini*ImageNet, *tiered*ImageNet, and FC100, and achieve top performance using the epoch-dependent transductive hyperprior learner, which captures the richest information. Our ablation study shows that both "epoch-wise ensemble" and "empirical" encourage high efficiency and robustness in the model performance[1].

1 Introduction

The ability of learning new concepts from a handful of examples is well-handled by humans, while in contrast, it remains challenging for machine models whose typical training requires a significant amount of data for good performance [34]. However, in many real-world applications, we have to face the situations of lacking a significant amount of training data, as e.g., in the medical domain. It is thus desirable to improve machine learning models to handle few-shot settings where each new concept has very scarce examples [13, 30, 39, 70].

Meta-learning methods aim to tackle the few-shot learning problem by transferring experience from similar few-shot tasks [7]. There are different meta strategies, among which the gradient descent based methods are particularly promising for today's neural networks [1, 13–15, 20, 25, 38, 70, 74, 81, 83, 84, 86]. These methods follow a unified meta-learning procedure that contains two loops. The
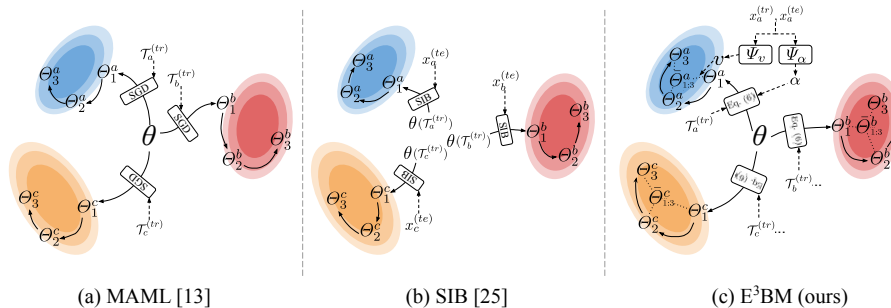
---

[1] Our code is open-sourced at https://gitlab.mpi-klsb.mpg.de/yaoyaoliu/e3bm.

(a) MAML [13]   (b) SIB [25]   (c) E³BM (ours)

Fig. 1. Conceptual illustrations of the model adaptation on the blue, red and yellow tasks. (a) MAML [13] is the classical inductive method that meta-learns a network initialization θ that is used to learn a single base-learner on each task, e.g., Θ_3^a in the blue task. (b) SIB [25] is a transductive method that formulates a variational posterior as a function of both labeled training data $\mathcal{T}^{(tr)}$ and unlabeled test data $x^{(te)}$. It also uses a single base-learner and optimizes the learner by running several synthetic gradient steps on $x^{(te)}$. (c) Our E³BM is a generic method that learns to combine the epoch-wise base-learners (e.g., Θ_1, Θ_2, and Θ_3), and to generate task-specific learning rates α and combination weights v that encourage robust adaptation. $\bar{\Theta}_{1:3}$ denotes the ensemble result of three base-learners; Ψ_α and Ψ_v denote the hyperprior learners learned to generate α and v, respectively. Note that figure (c) is based on E³BM+MAML, i.e., plug-in our E³BM to MAML baseline. Other plug-in versions are introduced in Sec. 4.4.

inner loop learns a base-learner for each individual task, and the outer loop uses the validation loss of the base-learner to optimize a meta-learner. In previous works [1, 13, 14, 70], the task of the meta-learner is to initialize the base-learner for the fast and efficient adaptation to the few training samples in the new task.

In this work, we aim to address two shortcomings of the previous works. First, the learning process of a base-learner for few-shot tasks is quite unstable [1], and often results in high-variance or low-confidence predictions. An intuitive solution is to train an ensemble of models and use the combined prediction which should be more robust [6, 29, 54]. However, it is not obvious how to obtain and combine multiple base-learners given the fact that a very limited number of training examples are available. Rather than learning multiple independent base-learners [79], we propose a novel method of utilizing the sequence of epoch-wise base-learners (while training a single base-learner) as the ensemble. Second, it is well-known that the values of hyperparameters, e.g., for initializing and updating models, are critical for best performance, and are particularly important for few-shot learning. In order to explore the optimal hyperparameters, we propose to employ the empirical Bayes method in the paradigm of meta-learning. In specific, we meta-learn hyperprior learners with meta-training tasks, and use them to generate task-specific hyperparameters, e.g., for updating and ensembling multiple base-learners. We call the resulting novel approach **E³BM**, which learns the **E**nsemble of **E**poch-wise **E**mpirical **B**ayes **M**odels for each few-shot task.

Our "epoch-wise models" are *different models* since each one of them is resulted from a specific training epoch and is trained with a specific set of hyperparameter values. During test, E^3BM combines the ensemble of models' predictions with soft ensembling weights to produce more robust results. In this paper, we argue that during model adaptation to the few-shot tasks, the most active adapting behaviors actually happen in the early epochs, and then converge to and even overfit to the training data in later epochs. Related works use the single base-learner obtained from the last epoch, so their meta-learners learn only partial adaptation experience [13, 14, 25, 70]. In contrast, our E^3BM leverages an ensemble modeling strategy that adapts base-learners at different epochs and each of them has task-specific hyperparameters for updating and ensembling. It thus obtains the optimized combinational adaptation experience. Figure 1 presents the conceptual illustration of E^3BM, compared to those of the classical method MAML [13] and the state-of-the-art SIB [25].

Our main contributions are three-fold. (1) A novel few-shot learning approach E^3BM that learns to learn and combine an ensemble of epoch-wise Bayes models for more robust few-shot learning. (2) Novel hyperprior learners in E^3BM to generate the task-specific hyperparameters for learning and combining epoch-wise Bayes models. In particular, we introduce four kinds of hyperprior learner by considering inductive [13, 70] and transductive learning methods [25], and each with either epoch-dependent (e.g., LSTM) or epoch-independent (e.g., epoch-wise FC layer) architectures. (3) Extensive experiments on three challenging few-shot benchmarks, *mini*ImageNet [73], *tiered*ImageNet [58] and Fewshot-CIFAR100 (FC100) [53]. We plug-in our E^3BM to the state-of-the-art few-shot learning methods [13, 25, 70] and obtain consistent performance boosts. We conduct extensive model comparison and observe that our E^3BM employing an epoch-dependent transductive hyperprior learner achieves the top performance on all benchmarks.

2   Related Works

Few-shot learning & meta-learning. Research literature on few-shot learning paradigms exhibits a high diversity from using data augmentation techniques [9, 75, 77] over sharing feature representation [2, 76] to meta-learning [18, 72]. In this paper, we focus on the meta-learning paradigm that leverages few-shot learning experiences from similar tasks based on the episodic formulation (see Section 3). Related works can be roughly divided into three categories. (1) *Metric learning methods* [12, 24, 40, 41, 64, 71, 73, 78, 82] aim to learn a similarity space, in which the learning should be efficient for few-shot examples. The metrics include Euclidean distance [64], cosine distance [8, 73], relation module [24, 41, 71] and graph-based similarity [45, 62]. Metric-based task-specific feature representation learning has also been presented in many related works [12, 24, 41, 78]. (2) *Memory network methods* [50, 52, 53] aim to learn training "experience" from the seen tasks and then aim to generalize to the learning of the unseen ones. A model with external memory storage is designed specifically for fast learning in a

few iterations, e.g., Meta Networks [52], Neural Attentive Learner (SNAIL) [50], and Task Dependent Adaptive Metric (TADAM) [53]. (3) *Gradient descent based methods* [1, 13, 14, 20, 25, 37, 38, 43, 57, 70, 86] usually employ a meta-learner that learns to fast adapt an NN base-learner to a new task within a few optimization steps. For example, Rusu *et al.* [61] introduced a classifier generator as the meta-learner, which outputs parameters for each specific task. Lee *et al.* [37] presented a meta-learning approach with convex base-learners for few-shot tasks. Finn *et al.* [13] designed a meta-learner called MAML, which learns to effectively initialize the parameters of an NN base-learner for a new task. Sun *et al.* [69, 70] introduced an efficient knowledge transfer operator on deeper neural networks and achieved a significant improvement for few-shot learning models. Hu *et al.* [25] proposed to update base-learner with synthetic gradients generated by a variational posterior conditional on unlabeled data. Our approach is closely related to gradient descent based methods [1, 13, 25, 69, 70, 70]. An important difference is that we learn how to combine an ensemble of epoch-wise base-learners and how to generate efficient hyperparameters for base-learners, while other methods such as MAML [13], MAML++ [1], LEO [61], MTL [69, 70], and SIB [25] use a single base-learner.

Hyperparameter optimization. Building a model for a new task is a process of exploration-exploitation. Exploring suitable architectures and hyperparameters are important before training. Traditional methods are model-free, e.g., based on grid search [4, 28, 42]. They require multiple full training trials and are thus costly. Model-based hyperparameter optimization methods are adaptive but sophisticated, e.g., using random forests [27], Gaussian processes [65] and input warped Gaussian processes [67] or scalable Bayesian optimization [66]. In our approach, we meta-learn a hyperprior learner to output optimal hyperparameters by gradient descent, without additional manual labor. Related methods using gradient descent mostly work for single model learning in an inductive way [3, 10, 15, 44, 46–49]. While, our hyperprior learner generates a sequence of hyperparameters for multiple models, in either the inductive or the transductive learning manner.

Ensemble modeling. It is a strategy [26, 85] to use multiple algorithms to improve machine learning performance, and which is proved to be effective to reduce the problems related to overfitting [35, 68]. Mitchell et al. [51] provided a theoretical explanation for it. Boosting is one classical way to build an ensemble, e.g., AdaBoost [16] and Gradient Tree Boosting [17]. Stacking combines multiple models by learning a combiner and it applies to both tasks in supervised learning [6, 29, 54] and unsupervised learning [63]. Bootstrap aggregating (i.e., Bagging) builds an ensemble of models through parallel training [6], e.g., random forests [22]. The ensemble can also be built on a temporal sequence of models [36]. Some recent works have applied ensemble modeling to few-shot learning. Yoon et al. proposed Bayesian MAML (BMAML) that trains multiple instances of base-model to reduce mete-level overfitting [80]. The most recent work [11] encourages multiple networks to cooperate while keeping predictive diversity. Its networks are trained with carefully-designed penalty functions, dif-

ferent from our automated method using empirical Bayes. Besides, its method needs to train much more network parameters than ours. Detailed comparisons are given in the experiment section.

3   Preliminary

In this section, we introduce the unified episodic formulation of few-shot learning, following [13, 57, 73]. This formulation was proposed for few-shot classification first in [73]. Its problem definition is different from traditional classification in three aspects: (1) the main phases are not training and test but meta-training and meta-test, each of which includes training and test; (2) the samples in meta-training and meta-testing are not datapoints but episodes, i.e. few-shot classification tasks; and (3) the objective is not classifying unseen datapoints but to fast adapt the meta-learned knowledge to the learning of new tasks.

Given a dataset $\mathcal{D}$ for meta-training, we first sample few-shot episodes (tasks) $\{\mathcal{T}\}$ from a task distribution $p(\mathcal{T})$ such that each episode $\mathcal{T}$ contains a few samples of a few classes, e.g., 5 classes and 1 shot per class. Each episode $\mathcal{T}$ includes a training split $\mathcal{T}^{(tr)}$ to optimize a specific base-learner, and a test split $\mathcal{T}^{(te)}$ to compute a generalization loss to optimize a global meta-learner. For meta-test, given an unseen dataset $\mathcal{D}_{un}$ (i.e., samples are from unseen classes), we sample a test task $\mathcal{T}_{un}$ to have the same-size training/test splits. We first initiate a new model with meta-learned network parameters (output from our hyperprior learner), then train this model on the training split $\mathcal{T}_{un}^{(tr)}$. We finally evaluate the performance on the test split $\mathcal{T}_{un}^{(te)}$. If we have multiple tasks, we report average accuracy as the final result.

4   An Ensemble of Epoch-wise Empirical Bayes Models

As shown in Fig. 2, E³BM trains a sequence of epoch-wise base-learners $\{\Theta_m\}$ with training data $\mathcal{T}^{(tr)}$ and learns to combine their predictions $\{z_m^{(te)}\}$ on test data $x^{(te)}$ for the best performance. This ensembling strategy achieves more robustness during prediction. The hyperparameters of each base-learner, i.e., learning rates α and combination weights v, are generated by the hyperprior learners conditional on task-specific data, e.g., $x^{(tr)}$ and $x^{(te)}$. This approach encourages the high diversity and informativeness of the ensembling models.

4.1   Empirical Bayes method

Our approach can be formulated as an empirical Bayes method that learns two levels of models for a few-shot task. The first level has hyperprior learners that generate hyperparameters for updating and combining the second-level models. More specifically, these second-level models are trained with the loss derived from the combination of their predictions on training data. After that, their loss
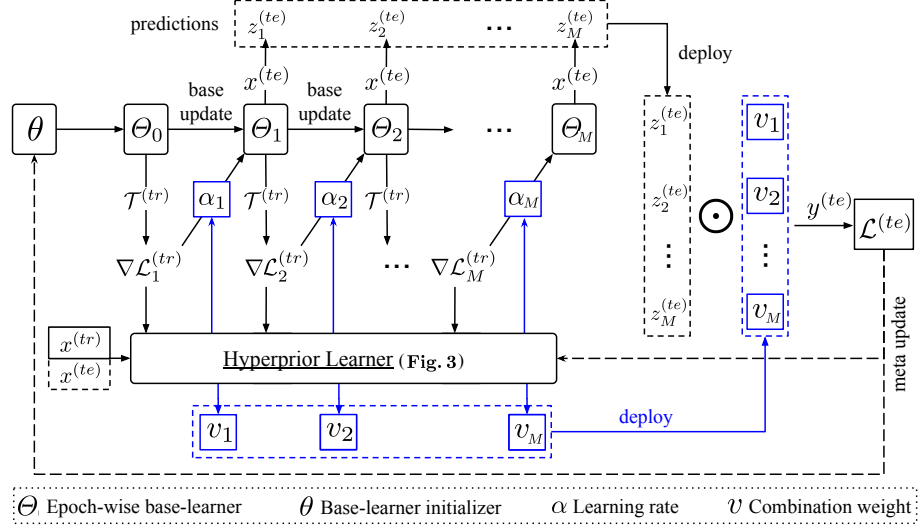
Fig. 2. The computing flow of the proposed E^3BM approach in one meta-training episode. For the meta-test task, the computation will be ended with predictions. Hyper-learner predicts task-specific hyperparameters, i.e., learning rates and multi-model combination weights. When its input contains $x^{(te)}$, it is transductive, otherwise inductive. Its detailed architecture is given in Fig. 3.

of test data are used to optimize the hyperprior learners. This process is also called meta update, see the dashed arrows in Fig. 2.

In specific, we sample K episodes $\{\mathcal{T}_k\}_{k=1}^{K}$ from the meta-training data $\mathcal{D}$. Let Θ denote base-learner and ψ represent its hyperparameters. An episode $\mathcal{T}_k$ aims to train Θ to recognize different concepts, so we consider to use concepts related (task specific) data for customizing the Θ through a hyperprior $p(\psi_k)$. To achieve this, we first formulate the empirical Bayes method with marginal likelihood according to hierarchical structure among data as follows,

$$p(\mathcal{T}) = \prod_{k=1}^{K} p(\mathcal{T}_k) = \prod_{k=1}^{K} \int_{\psi_k} p(\mathcal{T}_k|\psi_k)p(\psi_k)d\psi_k. \tag{1}$$

Then, we use variational inference [23] to estimate $\{p(\psi_k)\}_{k=1}^{K}$. We parametrize distribution $q_{\varphi_k}(\psi_k)$ with φ_k for each $p(\psi_k)$, and update φ_k to increase the similarity betweeen $q_{\varphi_k}(\psi_k)$ and $p(\psi_k)$. As in standard probabilistic modeling, we derive an evidence lower bound on the log version of Eq. (1) to update φ_k,

$$\log p(\mathcal{T}) \geqslant \sum_{k=1}^{K} \Big[\mathbb{E}_{\psi_k \sim q_{\varphi_k}} \big[\log p(\mathcal{T}_k|\psi_k) \big] - D_{\mathrm{KL}}(q_{\varphi_k}(\psi_k)||p(\psi_k)) \Big]. \tag{2}$$

Therefore, the problem of using $q_{\varphi_k}(\psi_k)$ to approach to the best estimation of $p(\psi_k)$ becomes equivalent to the objective of maximizing the evidence lower

bound [5, 23, 25] in Eq. (2), with respect to $\{\varphi_k\}_{k=1}^K$, as follows,

$$\min_{\{\varphi_k\}_{k=1}^K} \frac{1}{K} \sum_{k=1}^K \Big[\mathbb{E}_{\psi_k \sim q_{\varphi_k}} \big[-\log p(\mathcal{T}_k|\psi_k) \big] + D_{\mathrm{KL}}(q_{\varphi_k}(\psi_k)||p(\psi_k)) \Big]. \qquad (3)$$

To improve the robustness of few-shot models, existing methods sample a significant amount number of episodes during meta-training [13, 70]. Each episode employing its own hyperprior $p(\psi_k)$ causes a huge computation burden, making it difficult to solve the aforementioned optimization problem. To tackle this, we leverage a technique called "amortized variational inference" [25, 32, 59]. We parameterize the KL term in $\{\varphi_k\}_{k=1}^K$ (see Eq. (3)) with a unified deep neural network $\Psi(\cdot)$ taking $x_k^{(tr)}$ (inductive learning) or $\{x_k^{(tr)}, x_k^{(te)}\}$ (transductive learning) as inputs, where $x_k^{(tr)}$ and $x_k^{(te)}$ respectively denote the training and test samples in the k-th episode. In this paper, we call $\Psi(\cdot)$ hyperprior learner. As shown in Fig. 3, we additionally feed the hyperprior learner with the training gradients $\nabla \mathcal{L}_\Theta(\mathcal{T}_k^{(tr)})$ to $\Psi(\cdot)$ to encourage it to "consider" the cur-

(a) Epoch-independent   (b) Epoch-dependent

Fig. 3. Two options of hyperprior learner at the m-th base update epoch. In terms of the mapping function, we deploy either FC layers to build epoch-independent hyperprior learners, or LSTM to build an epoch-dependent learner. Values in dashed box were learned from previous tasks.

rent state of the training epoch. We mentioned in Sec. 1 that base-learners at different epochs are adapted differently, so we expect the corresponding hyperprior learner to "observe" and "utilize" this information to produce effective hyperparameters. By replacing q_{φ_k} with $q_{\Psi(\cdot)}$, Problem (3) can be rewritten as:

$$\min_{\Psi} \frac{1}{K} \sum_{k=1}^K \Big[\mathbb{E}_{\psi_k \sim q_{\Psi(\cdot)}} \big[-\log p(\mathcal{T}_k|\psi_k) \big] + D_{\mathrm{KL}}(q_{\Psi(\cdot)}(\psi_k)||p(\psi_k)) \Big]. \qquad (4)$$

Then, we solve Problem (4) by optimizing $\Psi(\cdot)$ with the meta gradient descent method used in classical meta-learning paradigms [13, 25, 70]. We elaborate the details of learning $\{\Theta_m\}$ and meta-learning $\Psi(\cdot)$ in the following sections.
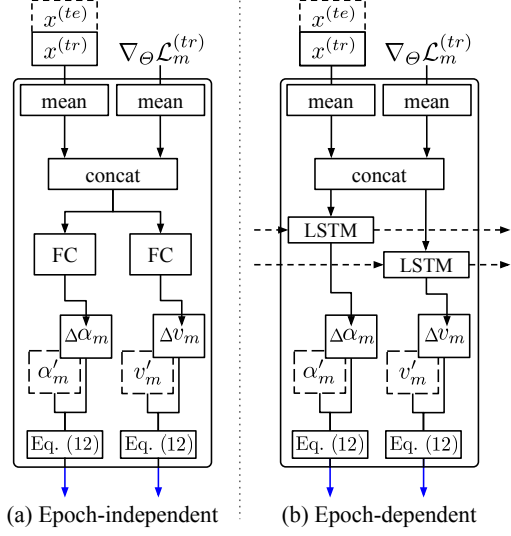
4.2   Learning the ensemble of base-learners

Previous works have shown that training multiple instances of the base-learner is helpful to achieve robust few-shot learning [12,79]. However, they suffer from the computational burden of optimizing multiple copies of neural networks in parallel, and are not easy to generalize to deeper neural architectures. If include the computation of second-order derivatives in meta gradient descent [13], this burden becomes more unaffordable. In contrast, our approach is free from this problem, because it is built on top of optimization-based meta-learning models, e.g., MAML [13], MTL [70], and SIB [25], which naturally produce a sequence of models along the training epochs in each episode.

Given an episode $\mathcal{T} = \{\mathcal{T}^{(tr)}, \mathcal{T}^{(te)}\} = \{\{x^{(tr)}, y^{(tr)}\}, \{x^{(te)}, y^{(te)}\}\}$, let Θ_m denote the parameters of the base-learner working at epoch m (w.r.t. m-th base-learner or BL-m), with $m \in \{1, ..., M\}$. Basically, we initiate BL-1 with parameters θ (network weights and bias) and hyperparameters (e.g., learning rate α), where θ is meta-optimized as in MAML [13], and α is generated by the proposed hyperprior learner Ψ_α. We then adapt BL-1 with normal gradient descent on the training set $\mathcal{T}^{(tr)}$, and use the adapted weights and bias to initialize BL-2. The general process is thus as follows,

$$\Theta_0 \leftarrow \theta, \tag{5}$$

$$\Theta_m \leftarrow \Theta_{m-1} - \alpha_m \nabla_\Theta \mathcal{L}_m^{(tr)} = \Theta_{m-1} - \Psi_\alpha(\tau, \nabla_\Theta \mathcal{L}_m^{(tr)}) \nabla_\Theta \mathcal{L}_m^{(tr)}, \tag{6}$$

where α_m is the learning rate outputted from Ψ_α, and $\nabla_\Theta \mathcal{L}_m^{(tr)}$ are the derivatives of the training loss, i.e, gradients. τ represents either $x^{(tr)}$ in the inductive setting, or $\{x^{(tr)}, x^{(te)}\}$ in the transductive setting. Note that Θ_0 is introduced to make the notation consistent, and a subscript m is omitted from Ψ_α for conciseness. Let $F(x; \Theta_m)$ denote the prediction scores of input x, so the base-training loss $\mathcal{T}^{(tr)} = \{x^{(tr)}, y^{(tr)}\}$ can be unfolded as,

$$\mathcal{L}_m^{(tr)} = L_{ce}\big(F(x^{(tr)}; \Theta_{m-1}), y^{(tr)}\big), \tag{7}$$

where L_{ce} is the softmax cross entropy loss. During episode test, each base-learner BL-m infers the prediction scores z_m for test samples $x^{(te)}$,

$$z_m = F(x^{(te)}; \Theta_m). \tag{8}$$

Assume the hyperprior learner Ψ_v generates the combination weight v_m for BL-m. The final prediction score is initialized as $\hat{y}_1^{(te)} = v_1 z_1$. For the m-th base epoch, the prediction z_m will be calculated and added to $\hat{y}^{(te)}$ as follows,

$$\hat{y}_m^{(te)} \leftarrow v_m z_m + \hat{y}_{m-1}^{(te)} = \Psi_v(\tau, \nabla_\Theta \mathcal{L}_m^{(tr)}) F(x^{(te)}; \Theta_m) + \hat{y}_{m-1}^{(te)}. \tag{9}$$

In this way, we can update prediction scores without storing base-learners or feature maps in the memory.

4.3   Meta-learning the hyperprior learners

As presented in Fig. 3, we introduce two architectures, i.e., LSTM or individual FC layers, for the hyperprior learner. FC layers at different epochs are independent. Using LSTM to "connect" all epochs is expected to "grasp" more task-specific information from the overall training states of the task. In the following, we elaborate the meta-learning details for both designs.

Assume before the k-th episode, we have meta-learned the base learning rates $\{\alpha'_m\}_{m=1}^M$ and combination weights $\{v'_m\}_{m=1}^M$. Next in the k-th episode, specifically at the m-th epoch as shown in Fig. 3, we compute the mean values of τ and $\nabla_{\Theta_m}\mathcal{L}_m^{(tr)}$, respectively, over all samples[2]. We then input the concatenated value to FC or LSTM mapping function as follows,

$$\Delta\alpha_m, \Delta v_m = \text{FC}_m(\text{concat}[\bar{\tau}; \overline{\nabla_{\Theta_m}\mathcal{L}_m^{(tr)}}]), \textbf{ or} \tag{10}$$

$$[\Delta\alpha_m, \Delta v_m], h_m = \text{LSTM}(\text{concat}[\bar{\tau}; \overline{\nabla_{\Theta_m}\mathcal{L}_m^{(tr)}}], h_{m-1}), \tag{11}$$

where h_m and h_{m-1} are the hidden states at epoch m and epoch $m-1$, respectively. We then use the output values to update hyperparameters as,

$$\alpha_m = \lambda_1\alpha'_m + (1-\lambda_1)\Delta\alpha, \ v_m = \lambda_2 v'_m + (1-\lambda_2)\Delta v, \tag{12}$$

where λ_1 and λ_2 are fixed fractions in $(0,1)$. Using learning rate α_m, we update BL-$(m-1)$ to be BL-m with Eq. (6). After M epochs, we obtain the combination of predictions $\hat{y}_M^{(te)}$ (see Eq. (9)) on test samples. In training tasks, we compute the test loss as,

$$\mathcal{L}^{(te)} = L_{ce}(\hat{y}_M^{(te)}, y^{(te)}). \tag{13}$$

We use this loss to calculate meta gradients to update Ψ as follows,

$$\Psi_\alpha \leftarrow \Psi_\alpha - \beta_1\nabla_{\Psi_\alpha}\mathcal{L}^{(te)}, \ \ \Psi_v \leftarrow \Psi_v - \beta_2\nabla_{\Psi_v}\mathcal{L}^{(te)}, \tag{14}$$

where β_1 and β_2 are meta-learning rates that determine the respective stepsizes for updating Ψ_α and Ψ_v. These updates are to back-propagate the test gradients till the input layer, through unrolling all base training gradients of $\Theta_1 \sim \Theta_M$. The process thus involves a gradient through a gradient [13,14,70]. Computationally, it requires an additional backward pass through $\mathcal{L}^{(tr)}$ to compute Hessian-vector products, which is supported by standard numerical computation libraries such as TensorFlow [19] and PyTorch [55].

4.4   Plugging-in E³BM to baseline methods

The optimization of Ψ relies on meta gradient descent method which was first applied to few-shot learning in MAML [13]. Recently, MTL [70] showed more efficiency by implementing that method on deeper pre-trained CNNs (e.g., ResNet-12 [70], and ResNet-25 [69]). SIB [25] was built on even deeper and wider networks (WRN-28-10), and it achieved top performance by synthesizing gradients

---

[2] In the inductive setting, training images are used to compute $\bar{\tau}$; while in the transductive setting, test images are additionally used.

in transductive learning. These three methods are all optimization-based, and use the single base-learner of the last base-training epoch. In the following, we describe how to learn and combine multiple base-learners in MTL, SIB and MAML, respectively, using our E^3BM approach.

According to [25, 70], we pre-train the feature extractor f on a many-shot classification task using the whole set of $\mathcal{D}$. The meta-learner in MTL is called scaling and shifting weights Φ_{SS}, and in SIB is called synthetic information bottleneck network $\phi(\lambda, \xi)$. Besides, there is a common meta-learner called base-learner initializer θ, i.e., the same θ in Fig. 2, in both methods. In MAML, the only base-learner is θ and there is no pre-training for its feature extractor f.

Given an episode $\mathcal{T}$, we feed training images $x^{(tr)}$ and test images $x^{(te)}$ to the feature extractor $f \odot \Phi_{SS}$ in MTL (f in SIB and MAML), and obtain the embedding $e^{(tr)}$ and $e^{(te)}$, respectively. Then in MTL, we use $e^{(tr)}$ with labels to train base-learner Θ for M times to get $\{\Theta_m\}_{m=1}^M$ with Eq. (6). In SIB, we use its multilayer perceptron (MLP) net to synthesize gradients conditional on $e^{(te)}$ to indirectly update $\{\Theta_m\}_{m=1}^M$. During these updates, our hyperprior learner Ψ_α derives the learning rates for all epochs. In episode test, we feed $e^{(te)}$ to $\{\Theta_m\}_{m=1}^M$ and get the combined prediction $\{z_m\}_{m=1}^M$ with Eq. (9). Finally, we compute the test loss to meta-update $[\Psi_\alpha; \Psi_v; \Phi_{SS}; \theta]$ in MTL, $[\Psi_\alpha; \Psi_v; \phi(\lambda, \xi); \theta]$ in SIB, and $[f; \theta]$ in MAML. We call the resulting methods MTL+E^3BM, SIB+E^3BM, and MAML+E^3BM, respectively, and demonstrate their improved efficiency over baseline models [13, 25, 70] in experiments.

5 Experiments

We evaluate our approach in terms of its overall performance and the effects of its two components, i.e. ensembling epoch-wise models and meta-learning hyperprior learners. In the following sections, we introduce the datasets and implementation details, compare our best results to the state-of-the-art, and conduct an ablation study.

5.1 Datasets and implementation details

Datasets. We conduct few-shot image classification experiments on three benchmarks: *mini*ImageNet [73], *tiered*ImageNet [58] and FC100 [53]. *mini*ImageNet is the most widely used in related works [13, 24, 25, 25, 70, 71]. *tiered*ImageNet and FC100 are either with a larger scale or a more challenging setting with lower image resolution, and have stricter training-test splits.

***mini*ImageNet** was proposed in [73] based on ImageNet [60]. There are 100 classes with 600 samples per class. Classes are divided into 64, 16, and 20 classes respectively for sampling tasks for meta-training, meta-validation and meta-test. ***tiered*ImageNet** was proposed in [58]. It contains a larger subset of ImageNet [60] with 608 classes ($779, 165$ images) grouped into 34 super-class nodes. These nodes are partitioned into 20, 6, and 8 disjoint sets respectively for meta-training, meta-validation and meta-test. Its super-class based training-test split

results in a more challenging and realistic regime with test tasks that are less similar to training tasks. **FC100** is based on the CIFAR100 [33]. The few-shot task splits were proposed in [53]. It contains 100 object classes and each class has 600 samples of 32×32 color images per class. On these datasets, we consider the (5-class, 1-way) and (5-class, 5-way) classification tasks. We use the same task sampling strategy as in related works [1, 13, 25].

Backbone architectures. In MAML+E^{3}BM, we use a 4-layer convolution network (4CONV) [1, 13]. In MTL+E^{3}BM, we use a 25-layer residual network (ResNet-25) [56, 69, 78]. Followed by convolution layers, we apply an average pooling layer and a fully-connected layer. In SIB+E^{3}BM, we use a 28-layer wide residual network (WRN-28-10) as SIB [25].

The configuration of base-learners. In MTL [70] and SIB [25], the base-learner is a single fully-connected layer. In MAML [13], the base-learner is the 4-layer convolution network. In MTL and MAML, the base-learner is randomly initialized and updated during meta-learning. In SIB, the base-learner is initialized with the averaged image features of each class. The number of base-learners M in MTL+E^{3}BM and SIB+E^{3}BM are respectively 100 and 3, i.e., the original numbers of training epochs in [25, 70].

The configuration of hyperprior learners. In Fig. 3, we show two options for hyperprior learners (i.e., Ψ_α and Ψ_v). Fig. 3(a) is the epoch-independent option, where each epoch has two FC layers to produce α and v respectively. Fig. 3(b) is the epoch-dependent option which uses an LSTM to generate α and v at all epochs. In terms of the learning hyperprior learners, we have two settings: inductive learning denoted as "Ind.", and transductive learning as "Tra.". "Ind." is the supervised learning in classical few-shot learning methods [13,37,64,70,73]. "Tra." is semi-supervised learning, based on the assumption that all test images of the episode are available. It has been applied to many recent works [24,25,45].

Ablation settings. We conduct a careful ablative study for two components, i.e., "ensembling multiple base-learners" and "meta-learning hyperprior learners". We show their effects indirectly by comparing our results to those of using arbitrary constant or learned values of v and α. **In terms of v**, we have 5 ablation options: (v1) "E^{3}BM" is our method generating v from Ψ_v; (v2) "learnable" is to set v to be update by meta gradient descent same as θ in [13]; (v3) "optimal" means using the values learned by option (a2) and freezing them during the actual learning; (v4) "equal" is an simple baseline using equal weights; (v5) "last-epoch" uses only the last-epoch base-learner, i.e., v is set to $[0, 0, ..., 1]$. In the experiments of (v1)-(v5), we simply set α as in the following (a4) [13,25,70]. **In terms of α**, we have 4 ablation options: (a1) "E^{3}BM" is our method generating α from Ψ_α; (a2) "learnable" is to set α to be update by meta gradient descent same as θ in [13]; (a3) "optimal" means using the values learned by option (a2) and freezing them during the actual learning; (a4) "fixed" is a simple baseline that uses manually chosen α following [13,25,70]. In the experiments of (a1)-(a4), we simply set v as in (v5), same with the baseline method [70].

Methods	Backbone	*mini*ImageNet		*tiered*ImageNet		FC100	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MatchNets [73]	4CONV	43.44	55.31	–	–	–	–
ProtoNets [64]	4CONV	49.42	68.20	53.31	72.69	–	–
MAML$^\diamond$ [13]	4CONV	48.70	63.11	49.0	66.5	38.1	50.4
MAML++$^\diamond$ [1]	4CONV	52.15	68.32	51.5	70.6	38.7	52.9
TADAM [53]	ResNet-12	58.5	76.7	–	–	40.1	56.1
MetaOptNet [37]	ResNet-12	62.64	78.63	65.99	81.56	41.1	55.5
CAN [24]	ResNet-12	63.85	79.44	**69.89**	84.23	–	–
CTM [40]	ResNet-18	**64.12**	80.51	68.41	**84.28**	–	–
MTL [70]	ResNet-12	61.2	75.5	–	–	**45.1**	57.6
MTL$^\diamond$ [70]	ResNet-25	63.4	80.1	69.1	84.2	43.7	**60.1**
LEO [61]	WRN-28-10	61.76	77.59	66.33	81.44	–	–
Robust20-dist‡ [12]	WRN-28-10	63.28	**81.17**	–	–	–	–
MAML+E^{3}BM	4CONV	53.2(↑4.5)	65.1(↑2.0)	52.1(↑3.1)	70.2(↑3.7)	39.9(↑1.8)	52.6(↑2.2)
(+time, +param)	–	(8.9, 2.2)	(9.7, 2.2)	(10.6, 2.2)	(9.3, 2.2)	(7.8, 2.2)	(12.1, 2.2)
MTL+E^{3}BM	ResNet-25	**64.3**(↑0.9)	**81.0**(↑0.9)	**70.0**(↑0.9)	**85.0**(↑0.8)	**45.0**(↑1.3)	**60.5**(↑0.4)
(+time, +param)	–	(5.9, 0.7)	(10.2, 0.7)	(6.7, 0.7)	(9.5, 0.7)	(5.7, 0.7)	(7.9, 0.7)

(a) Inductive Methods

EGNN [31]	ResNet-12	64.02	77.20	65.45	82.52	–	–
CAN+T [24]	ResNet-12	67.19	**80.64**	73.21	**84.93**	–	–
SIB$^{\diamond\ddagger}$ [25]	WRN-28-10	**70.0**	79.2	**72.9**	82.8	**45.2**	**55.9**
SIB+E^{3}BM‡	WRN-28-10	**71.4**(↑1.4)	**81.2**(↑2.0)	**75.6**(↑2.7)	**84.3**(↑1.5)	**46.0**(↑0.8)	**57.1**(↑1.2)
(+time, +param)	–	(2.1, 0.04)	(5.7, 0.04)	(5.2, 0.04)	(4.9, 0.04)	(6.1, 0.04)	(7.3, 0.04)

(b) Transductive Methods

$^\diamond$Our implementation on *tiered*ImageNet and FC100. ‡Input image size: $80 \times 80 \times 3$.

Table 1. The 5-class few-shot classification accuracies (%) on *mini*ImageNet, *tiered*ImageNet, and FC100. "(+time, +param)" denote the additional computational time (%) and parameter size (%), respectively, when plugging-in E^{3}BM to baselines (MAML, MTL and SIB). "–" means no reported results in original papers. The **best** and **second best** results are highlighted.

5.2   Results and analyses

In Table 1, we compare our best results to the state-of-the-arts. In Table 2, we present the results of using different kinds of hyperprior learner, i.e., regarding two architectures (FC and LSTM) and two learning strategies (inductive and transductive). In Fig. 4(a)(b), we show the validation results of our ablative methods, and demonstrate the change during meta-training iterations. In Fig. 4(c)(d), we plot the generated values of v and α during meta-training.

Comparing to the state-of-the-arts. Table 1 shows that the proposed E^{3}BM achieves the best few-shot classification performance in both 1-shot and 5-shot settings, on three benchmarks. Please note that [12] reports the results of using different backbones and input image sizes. We choose its results under the same setting of ours, i.e., using WRN-28-10 networks and $80 \times 80 \times 3$ images, for fair comparison. In our approach, plugging-in E^{3}BM to the state-of-the-art model SIB achieves 1.6% of improvement on average, based on the identical network architecture. This improvement is significantly larger as 2.9% when taking MAML

No.	Setting			*mini*ImageNet		*tiered*ImageNet		FC100	
	Method	Hyperprior	Learning	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
1	MTL [70]	–	Ind.	63.4	80.1	69.1	84.2	43.7	60.1
2	MTL+E^{3}BM	FC	Ind.	64.3	80.9	69.8	84.6	44.8	60.5
3	MTL+E^{3}BM	FC	Tra.	64.7	80.7	69.7	84.9	44.7	**60.6**
4	MTL+E^{3}BM	LSTM	Ind.	64.3	81.0	70.0	**85.0**	45.0	60.4
5	MTL+E^{3}BM	LSTM	Tra.	64.5	**81.1**	70.2	**85.3**	45.1	**60.6**
6	SIB [25]	–	Tra.	70.0	79.2	72.9	82.8	45.2	55.9
7	SIB+E^{3}BM	FC	Tra.	**71.3**	81.0	**75.2**	83.8	**45.8**	56.3
8	SIB+E^{3}BM	LSTM	Tra.	**71.4**	**81.2**	**75.6**	84.3	**46.0**	57.1

Table 2. The 5-class few-shot classification accuracies (%) of using different hyper-prior learners, on the *mini*ImageNet, *tiered*ImageNet, and FC100. "Ind." and "Tra." denote the inductive and transductive settings, respectively. The **best** and **second best** results are highlighted.

as baseline. All these show to be more impressive if considering the tiny overheads from pluging-in. For example, using E^{3}BM adds only 0.04% learning parameters to the original SIB model, and it gains only 5.2% average overhead regarding the computational time. It is worth mentioning that the amount of learnable parameters in SIB+E^{3}BM is around 80% less than that of model in [12] which ensembles 5 deep networks in parallel (and later learns a distillation network).

Hyperprior learners. In Table 2, we can see that using transductive learning clearly outperforms inductive learning, e.g., No. 5 *vs.* No. 4. This is because the "transduction" leverages additional data, i.e., the episode-test images (no labels), during the base-training. In terms of the network architecture, we observe that LSTM-based learners are slightly better than FC-based (e.g., No. 3 *vs.* No. 2). LSTM is a sequential model and is indeed able to "observe" more patterns from the adaptation behaviors of models at adjacent epochs.

Ablation study. Fig. 4(a) shows the comparisons among α related ablation models. Our E^{3}BM (orange) again performs the best, over the models of using any arbitrary α (red or light blue), as well as over the model with α optimized by the meta gradient descent (blue) [13]. Fig. 4(b) shows that our approach E^{3}BM works consistently better than the ablation models related to v. We should emphasize that E^{3}BM is clearly more efficient than the model trained with meta-learned v (blue) through meta gradient descent [13]. This is because E^{3}BM hyperprior learners generate empirical weights conditional on task-specific data. The LSTM-based learners can leverage even more task-specific information, i.e., the hidden states from previous epochs, to improve the efficiency.

The values of α and v learned by E^{3}BM. Fig. 4(c)(d) shows the values of α and v during the meta-training iterations in our approach. Fig. 4(c) show the base-learners working at later training epochs (e.g., BL-100) tend to get smaller values of α. This is actually similar to the common manual schedule, i.e. monotonically decreasing learning rates, of conventional large-scale network training [21]. The difference is that in our approach, this is "scheduled" in a total automated way by hyperprior learners. Another observation is that the

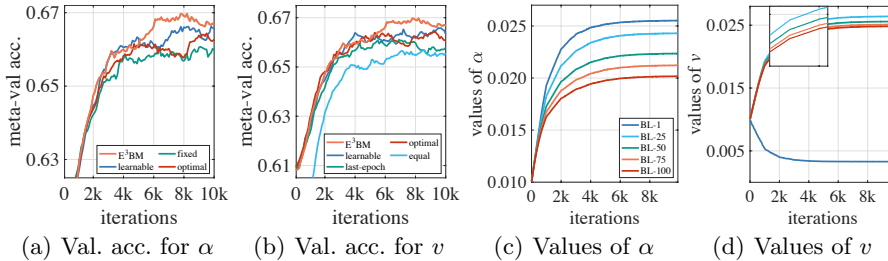(a) Val. acc. for α (b) Val. acc. for v (c) Values of α (d) Values of v

Fig. 4. (a)(b): The meta-validation accuracies of ablation models. The legends are explained in (a1)-(a4) and (v1)-(v5) in Sec. 5.1 **Ablation settings**. All curves are smoothed with a rate of 0.9 for a better visualization. (c)(d): The values of α and v generated by Ψ_α and Ψ_v, respectively. The setting is using MTL+E^{3}BM, ResNet-25, on *mini*ImageNet, 1-shot.

highest learning rate is applied to BL-1. This actually encourages BL-1 to make an influence as significant as possible. It is very helpful to reduce meta gradient diminishing when unrolling and back-propagating gradients through many base-learning epochs (e.g., 100 epochs in MTL). Fig. 4(d) shows that BL-1 working at the initial epoch has the lowest values of v. In other words, BL-1 is almost disabled in the prediction of episode test. Intriguingly, BL-25 instead of BL-100 gains the highest v values. Our explanation is that during the base-learning, base-learners at latter epochs get more overfitted to the few training samples. Their functionality is thus suppressed. Note that our empirical results revealed that including the overfitted base-learners slightly improves the generalization capability of the approach.

6 Conclusions

We propose a novel E^{3}BM approach that tackles the few-shot problem with an ensemble of epoch-wise base-learners that are trained and combined with task-specific hyperparameters. In specific, E^{3}BM meta-learns the hyperprior learners to generate such hyperparameters conditional on the images as well as the training states for each episode. Its resulting model allows to make use of multiple base-learners for more robust predictions. It does not change the basic training paradigm of episodic few-shot learning, and is thus *generic* and easy to plug-and-play with existing methods. By applying E^{3}BM to multiple baseline methods, e.g., MAML, MTL and SIB, we achieved top performance on three challenging few-shot image classification benchmarks, with little computation or parametrization overhead.

References

1. Antoniou, A., Edwards, H., Storkey, A.: How to train your maml. In: ICLR (2019) 1, 2, 4, 11, 12
2. Bart, E., Ullman, S.: Cross-generalization: Learning novel classes from a single example by feature replacement. In: CVPR. pp. 672–679 (2005) 3
3. Bengio, Y.: Gradient-based optimization of hyperparameters. Neural Computation **12**(8), 1889–1900 (2000) 4
4. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**, 281–305 (2012) 4
5. Blei, D.M., Kucukelbir, A., McAuliffe, J.D.: Variational inference: A review for statisticians. Journal of the American statistical Association **112**(518), 859–877 (2017) 7
6. Breiman, L.: Stacked regressions. Machine Learning **24**(1), 49–64 (1996) 2, 4
7. Caruana, R.: Learning many related tasks at the same time with backpropagation. In: NIPS. pp. 657–664 (1995) 1
8. Chen, W.Y., Liu, Y.C., Kira, Z., Wang, Y.C., Huang, J.B.: A closer look at few-shot classification. In: ICLR (2019) 3
9. Chen, Z., Fu, Y., Zhang, Y., Jiang, Y., Xue, X., Sigal, L.: Multi-level semantic feature augmentation for one-shot learning. IEEE Transactions Image Processing **28**(9), 4594–4605 (2019) 3
10. Domke, J.: Generic methods for optimization-based modeling. In: AISTATS. pp. 318–326 (2012) 4
11. Dvornik, N., Schmid, C., Julien, M.: f-VAEGAN-D2: A feature generating framework for any-shot learning. In: ICCV. pp. 10275–10284 (2019) 4
12. Dvornik, N., Schmid, C., Mairal, J.: Diversity with cooperation: Ensemble methods for few-shot classification. In: ICCV. pp. 3722–3730 (2019) 3, 8, 12, 13
13. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML. pp. 1126–1135 (2017) 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13
14. Finn, C., Xu, K., Levine, S.: Probabilistic model-agnostic meta-learning. In: NeurIPS. pp. 9537–9548 (2018) 1, 2, 3, 4, 9
15. Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M.: Bilevel programming for hyperparameter optimization and meta-learning. In: ICML. pp. 1563–1572 (2018) 1, 4
16. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55**(1), 119–139 (1997) 4
17. Friedman, J.H.: Stochastic gradient boosting. Computational Statistics & Data Analysis **38**(4), 367–378 (2002) 4
18. Geoffrey, H.E., David, P.C.: Using fast weights to deblur old memories. In: CogSci. pp. 177–186 (1987) 3
19. Girija, S.S.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. Software available from tensorflow. org **39** (2016) 9
20. Grant, E., Finn, C., Levine, S., Darrell, T., Griffiths, T.L.: Recasting gradient-based meta-learning as hierarchical bayes. In: ICLR (2018) 1, 4
21. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: CVPR. pp. 558–567 (2019) 13
22. Ho, T.K.: Random decision forests. In: ICDAR. vol. 1, pp. 278–282 (1995) 4

23. Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J.: Stochastic variational inference. The Journal of Machine Learning Research **14**(1), 1303–1347 (2013) 6, 7
24. Hou, R., Chang, H., Bingpeng, M., Shan, S., Chen, X.: Cross attention network for few-shot classification. In: NeurIPS. pp. 4005–4016 (2019) 3, 10, 11, 12
25. Hu, S.X., Moreno, P.G., Xiao, X.S.Y., Lawrence, N.D., Obozinski, G., Damianou, A., Champs-sur Marne, F.: Empirical bayes meta-learning with synthetic gradients. In: ICLR (2020) 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13
26. Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J.E., Weinberger, K.Q.: Snapshot ensembles: Train 1, get m for free. In: ICLR (2017) 4
27. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: LION. pp. 507–523 (2011) 4
28. Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W.M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., Kavukcuoglu, K.: Population based training of neural networks. arXiv **1711.09846** (2017) 4
29. Ju, C., Bibaut, A., van der Laan, M.: The relative performance of ensemble methods with deep convolutional neural networks for image classification. Journal of Applied Statistics **45**(15), 2800–2818 (2018) 2, 4
30. Jung, H.G., Lee, S.W.: Few-shot learning with geometric constraints. IEEE Transactions on Neural Networks and Learning Systems (2020) 1
31. Kim, J., Kim, T., Kim, S., Yoo, C.D.: Edge-labeling graph neural network for few-shot learning. In: CVPR. pp. 11–20 (2019) 12
32. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014) 7
33. Krizhevsky, A.: Learning multiple layers of features from tiny images. University of Toronto (2009) 11
34. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. pp. 1097–1105 (2012) 1
35. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning **51**(2), 181–207 (2003) 4
36. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: ICLR (2017) 4
37. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. In: CVPR. pp. 10657–10665 (2019) 4, 11, 12, 3
38. Lee, Y., Choi, S.: Gradient-based meta-learning with learned layerwise metric and subspace. In: ICML. pp. 2933–2942 (2018) 1, 4
39. Li, F., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(4), 594–611 (2006) 1
40. Li, H., Eigen, D., Dodge, S., Zeiler, M., Wang, X.: Finding task-relevant features for few-shot learning by category traversal. In: CVPR. pp. 1–10 (2019) 3, 12
41. Li, H., Dong, W., Mei, X., Ma, C., Huang, F., Hu, B.: Lgm-net: Learning to generate matching networks for few-shot learning. In: ICML. pp. 3825–3834 (2019) 3
42. Li, L., Jamieson, K.G., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research **18**, 185:1–185:52 (2017) 4
43. Li, X., Sun, Q., Liu, Y., Zhou, Q., Zheng, S., Chua, T.S., Schiele, B.: Learning to self-train for semi-supervised few-shot classification. In: NeurIPS. pp. 10276–10286 (2019) 4
44. Li, Z., Zhou, F., Chen, F., Li, H.: Meta-sgd: Learning to learn quickly for few shot learning. arXiv **1707.09835** (2017) 4

45. Liu, Y., Lee, J., Park, M., Kim, S., Yang, Y.: Learning to propagate labels: Transductive propagation network for few-shot learning. In: ICLR (2019) 3, 11
46. Liu, Y., Su, Y., Liu, A.A., Schiele, B., Sun, Q.: Mnemonics training: Multi-class incremental learning without forgetting. In: CVPR. pp. 12245–12254 (2020) 4
47. Luketina, J., Raiko, T., Berglund, M., Greff, K.: Scalable gradient-based tuning of continuous regularization hyperparameters. In: ICML. pp. 2952–2960 (2016) 4
48. Maclaurin, D., Duvenaud, D.K., Adams, R.P.: Gradient-based hyperparameter optimization through reversible learning. In: ICML. pp. 2113–2122 (2015) 4
49. Metz, L., Maheswaranathan, N., Cheung, B., Sohl-Dickstein, J.: Meta-learning update rules for unsupervised representation learning. In: ICLR (2019) 4
50. Mishra, N., Rohaninejad, M., Chen, X., Abbeel, P.: Snail: A simple neural attentive meta-learner. In: ICLR (2018) 3, 4
51. Mitchell, T.: Machine learning, mcgraw-hill higher education. New York (1997) 4
52. Munkhdalai, T., Yu, H.: Meta networks. In: ICML. pp. 2554–2563 (2017) 3, 4
53. Oreshkin, B.N., Rodríguez, P., Lacoste, A.: TADAM: task dependent adaptive metric for improved few-shot learning. In: NeurIPS. pp. 719–729 (2018) 3, 4, 10, 11, 12
54. Ozay, M., Vural, F.T.Y.: A new fuzzy stacked generalization technique and analysis of its performance. arXiv **1204.0171** (2012) 2, 4
55. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS. pp. 8024–8035 (2019) 9
56. Qiao, S., Liu, C., Shen, W., Yuille, A.L.: Few-shot image recognition by predicting parameters from activations. In: CVPR. pp. 7229–7238 (2018) 11
57. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: ICLR (2017) 4, 5
58. Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S.: Meta-learning for semi-supervised few-shot classification. In: ICLR (2018) 3, 10
59. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: ICML. pp. 1278–1286 (2014) 7
60. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015) 10
61. Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R.: Meta-learning with latent embedding optimization. In: ICLR (2019) 4, 12, 3
62. Satorras, V.G., Estrach, J.B.: Few-shot learning with graph neural networks. In: ICLR (2018) 3
63. Smyth, P., Wolpert, D.: Linearly combining density estimators via stacking. Machine Learning **36**(1-2), 59–83 (1999) 4
64. Snell, J., Swersky, K., Zemel, R.S.: Prototypical networks for few-shot learning. In: NIPS. pp. 4077–4087 (2017) 3, 11, 12
65. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: NIPS. pp. 2951–2959 (2012) 4
66. Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M.M.A., Prabhat, Adams, R.P.: Scalable bayesian optimization using deep neural networks. In: ICML. pp. 2171–2180 (2015) 4
67. Snoek, J., Swersky, K., Zemel, R.S., Adams, R.P.: Input warping for bayesian optimization of non-stationary functions. In: ICML. pp. 1674–1682 (2014) 4

68. Sollich, P., Krogh, A.: Learning with ensembles: How overfitting can be useful. In: NIPS. pp. 190–196 (1996) 4
69. Sun, Q., Liu, Y., Chen, Z., Chua, T., Schiele, B.: Meta-transfer learning through hard tasks. arXiv **1910.03648** (2019) 4, 9, 11
70. Sun, Q., Liu, Y., Chua, T.S., Schiele, B.: Meta-transfer learning for few-shot learning. In: CVPR. pp. 403–412 (2019) 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13
71. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H.S., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: CVPR. pp. 1199–1208 (2018) 3, 10
72. Thrun, S., Pratt, L.: Learning to learn: Introduction and overview. In: Learning to learn, pp. 3–17. Springer (1998) 3
73. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. In: NIPS. pp. 3630–3638 (2016) 3, 5, 10, 11, 12
74. Wang, X., Huang, T.E., Darrell, T., Gonzalez, J.E., Yu, F.: Frustratingly simple few-shot object detection. In: ICML (2020) 1
75. Wang, Y., Girshick, R.B., Hebert, M., Hariharan, B.: Low-shot learning from imaginary data. In: CVPR. pp. 7278–7286 (2018) 3
76. Wang, Y.X., Hebert, M.: Learning from small sample sets by combining unsupervised meta-training with cnns. In: NIPS. pp. 244–252 (2016) 3
77. Xian, Y., Sharma, S., Schiele, B., Akata, Z.: f-VAEGAN-D2: A feature generating framework for any-shot learning. In: CVPR. pp. 10275–10284 (2019) 3
78. Ye, H.J., Hu, H., Zhan, D.C., Sha, F.: Learning embedding adaptation for few-shot learning. arXiv **1812.03664** (2018) 3, 11
79. Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., Ahn, S.: Bayesian model-agnostic meta-learning. In: NeurIPS. pp. 7343–7353 (2018) 2, 8
80. Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., Ahn, S.: Bayesian model-agnostic meta-learning. In: NeurIPS. pp. 7343–7353 (2018) 4
81. Zhang, C., Cai, Y., Lin, G., Shen, C.: Deepemd: Differentiable earth mover's distance for few-shot learning. arXiv **2003.06777** (2020) 1, 3
82. Zhang, C., Cai, Y., Lin, G., Shen, C.: Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In: CVPR. pp. 12203–12213 (2020) 3
83. Zhang, C., Lin, G., Liu, F., Guo, J., Wu, Q., Yao, R.: Pyramid graph networks with connection attentions for region-based one-shot semantic segmentation. In: ICCV. pp. 9587–9595 (2019) 1
84. Zhang, C., Lin, G., Liu, F., Yao, R., Shen, C.: Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In: CVPR. pp. 5217–5226 (2019) 1
85. Zhang, L., Shi, Z., Cheng, M.M., Liu, Y., Bian, J.W., Zhou, J.T., Zheng, G., Zeng, Z.: Nonlinear regression via deep negative correlation learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019) 4
86. Zhang, R., Che, T., Grahahramani, Z., Bengio, Y., Song, Y.: Metagan: An adversarial approach to few-shot learning. In: NeurIPS. pp. 2371–2380 (2018) 1, 4

Supplementary Materials

These supplementary materials include E³BM algorithms, results with confidence intervals, the supplementary plots to Fig. 4, backbone architectures, implementation details, ablation results for MAML, the inference time and the number of parameters, and the execution steps of our source code with PyTorch.

A   E³BM algorithms

Algorithm 1 summarizes the meta-training (line 1-10) and meta-testing (line 11-16) procedures in our E³BM approach. For clarity, the base-learning steps within a single episode are moved to Algorithm 2.

Algorithm 1: An Ensemble of Epoch-wise Empirical Bayes Models (E³BM)

Input: Meta-train episode distribution $p_{tr}(\mathcal{T})$, Meta-test episode distribution $p_{te}(\mathcal{T})$, and meta-train stepsizes β_1 and β_2.
Output: The average accuracy of meta-test.
% Meta-train phase:
1 Randomly initialize θ;
2 **for** *all meta iterations* **do**
3   Sample a batch of meta-train episodes $\{\mathcal{T}_i\} \in p_{tr}(\mathcal{T})$;
4   **for** $\mathcal{T}_i$ *in* $\{\mathcal{T}_i\}$ **do**
5   Train the sequence of base-learners on $\mathcal{T}_i$ by **Algorithm 2**;
6   **end**
7   Evaluate $\mathcal{L}^{(te)}$ with Eq. (13) ;
8   Optimize Ψ_α, and Ψ_v with Eq. (14) using β_1 and β_2;
9   Optimize other meta components, e.g., θ.
10 **end**
    % Meta-test phase:
11 Sample meta-test episodes $\{\mathcal{T}_i\} \in p_{te}(\mathcal{T})$;
12 **for** $\mathcal{T}_i$ *in* $\{\mathcal{T}_i\}$ **do**
13   Train the sequence of base-learners on $\mathcal{T}_i$ and obtain the prediction scores $\hat{y}^{(te)}$ by **Algorithm 2**;
14   Compute episode test accuracy Acc_i;
15 **end**
16 Return the average accuracy of $\{Acc_i\}$.

B   Results with confidence intervals

In Table S1, we supplement the few-shot classification accuracy (%) on *mini*ImageNet, *tiered*ImageNet, and FC100 (5-class) with confidence intervals.

Algorithm 2: Learning the ensemble of base-learners in one episode

---

 Input: An episode $\mathcal{T}$, hyperprior learners Ψ_α and Ψ_v.
 Output: Prediction $\hat{y}^{(te)}$, and episode test loss $\mathcal{L}^{(te)}$.

1 Initialize $\Theta_0 = \theta$;
2 **for** m in $\{1, ..., M\}$ **do**
3   Evaluate $\mathcal{L}_m^{(tr)}$ with Eq. (7) and compute $\nabla_\Theta \mathcal{L}_m^{(tr)}$;
4   Get α_m from Ψ_α and get v_m from Ψ_v;
5   Get Θ_m using α_m with Eq. (6);
6   Compute z_m with Eq. (8);
7   **if** $m = 1$ **then**
8   Initialize $\hat{y}_1^{(te)} = v_1 z_1$;
9   **else**
10   Compute $\hat{y}_m^{(te)}$ using v_m with Eq. (9);
11   **end**
12 **end**
13 Evaluate $\mathcal{L}^{(te)}$ with Eq. (13).

Methods	Backbone	*mini*ImageNet		*tiered*ImageNet		FC100	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML+E³BM	4CONV	53.2 ± 1.8	65.1 ± 0.9	52.1 ± 1.8	70.2 ± 0.9	39.9 ± 1.8	52.6 ± 0.9
MTL+E³BM	ResNet-12	63.8 ± 0.4	80.1 ± 0.3	71.2 ± 0.4	85.3 ± 0.3	43.2 ± 0.3	60.2 ± 0.3
MTL+E³BM	ResNet-25	64.3 ± 0.4	81.0 ± 0.3	70.0 ± 0.4	85.0 ± 0.3	45.0 ± 0.4	60.5 ± 0.3
SIB+E³BM	WRN-28-10	71.4 ± 0.5	81.2 ± 0.4	75.6 ± 0.6	84.3 ± 0.4	46.0 ± 0.6	57.1 ± 0.4

Table S1. Supplementary to Table 1. Few-shot classification accuracy (%) on *mini*ImageNet, *tiered*ImageNet, and FC100 (5-class).

C   Supplementary figures

Supplementary to Fig. 4(a)(b). In Fig. S1, we supplement the meta-validation accuracies for the 1-shot and 5-shot cases on *mini*ImageNet, *tiered*ImageNet, and FC100 (Note that Fig. 4 already has the *mini*ImageNet 1-shot results).

Supplementary to Fig. 4(c)(d). On the *mini*ImageNet, we supplement the plots of α and v in the 5-shot case in Fig. S2(a)(b). On the *tiered*ImageNet, we show the plots of α and v in Fig. S2(c)(d) and (e)(f), respectively for 1-shot and 5-shot cases. On the FC100, we show the plots of α and v in Fig. S2(g)(h) and (i)(j), respectively for 1-shot and 5-shot cases. Each figure demonstrates the values of α (or v) generated by the model "MTL+E³BM" as in Table 1.

D   Backbone architectures

4CONV consists of 4 layers with 3×3 convolutions and 32 filters, followed by batch normalization (BN), a ReLU nonlinearity, and 2×2 max-pooling.

ResNet-12 has 3 residual blocks. Each block has 4 convolution layers with 3×3 kernels. The number of filters starts from 160 and is doubled every next block. After a global average pooling layer, it gets a 640-dim embedding. This architecture follows [37].

ResNet-25 has 3 residual blocks after an initial convolution layer. Each block has 8 convolution layers with 3×3 kernels. The number of filters starts from 160 and is doubled every next block. After a global average pooling layer, it gets a 640-dim embedding. This architecture follows [78].

WRN-28-10 has its depth and width set to 28 and 10, respectively. After a global average pooling in the last layer of the backbone, it gets a 640-dimensional embedding. For this backbone, we resize the input image to $80 \times 80 \times 3$ for a fair comparison with related methods [25, 70]. Other details are the same as those with ResNet-25 [61, 78].

E   Implementation details

MTL+E^{3}BM. The meta learning rates for the scaling and shifting weights Φ_{SS} and the base-learner initializer θ are set to 1×10^{-4} uniformly. The base learning rates $\{\alpha'_m\}_{m=1}^M$ (Fig. 3) are initialized as 1×10^{-2} [70,78]. We meta-train MTL+E^{3}BM for $10,000$ iterations and use the model, which has the highest meta-validation accuracy, for meta-test.

SIB+E^{3}BM. The meta learning rates for both SIB network $\phi(\lambda, \xi)$ and base-learner initializer θ are set to 1×10^{-3} uniformly. The base learning rates $\{\alpha'_m\}_{m=1}^M$ (Fig. 3) are initialized as 1×10^{-3} [25]. We meta-train SIB+E^{3}BM for $50,000$ iterations and use the model, which has the highest meta-validation accuracy, for meta-test.

MAML+E^{3}BM. MAML only contains a model initializer θ, and we set its meta-learning rate as 1×10^{-3} [13]. The base learning rates $\{\alpha'_m\}_{m=1}^M$ (Fig. 3) are initialized as 1×10^{-3}. We meta-train MAML+E^{3}BM for $60,000$ iterations and use the model, which has the highest meta-validation accuracy, for meta-test.

Shared hyperparameters. The meta learning rates for Ψ_α and Ψ_v are set to 1×10^{-6} uniformly. For initializing $\{v'_m\}_{m=1}^M$ (Fig. 3), we have two options. One is each v'_m is initialized as 1/(number of base-learners), and the other one is that $\{v'_m\}_{m=1}^{M-1}$ are initialized as 0 and v'_M as 1. In Eq. (12) in Sec. 4.3, λ_1 and λ_2 are set to 1×10^{-4}. For the rest of the hyperparameters, we follow the original settings of baselines [13, 25, 70].

Constraints for v and α. In the constraint mode, we applied the constraints on v and α to force them to be positive and smaller than 1. We did not have any constraint for Δv or $\Delta \alpha$. Please note that the constraints are not applied in the default setting.

Dataloader For MAML, we use the same dataloader as [13]. For MTL, we follow [78, 81, 82]. For SIB, we follow [25].

F   Ablation results for MAML

In Table S2, we supplement the ablation results for "MAML+E^{3}BM" on *mini*ImageNet, *tiered*ImageNet, and FC100 (5-class).

No.	Setting			*mini*ImageNet		*tiered*ImageNet		FC100	
	Method	Hyperprior	Learning	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
1	MAML [13]	–	Ind.	48.70	63.11	49.0	66.5	38.1	50.4
2	5x MAML	–	Ind.	52.1	65.1	51.1	68.8	40.1	50.8
3	MAML+E^{3}BM	FC	Ind.	52.1	65.1	51.1	68.8	39.5	51.7
4	MAML+E^{3}BM	FC	Tra.	52.8	65.3	52.2	69.5	40.4	51.8
5	MAML+E^{3}BM	LSTM	Ind.	53.2	65.0	52.1	70.2	39.9	52.6
6	MAML+E^{3}BM	LSTM	Tra.	53.8	65.2	52.7	70.5	40.4	52.3

Table S2. Supplementary to Table 2. Results (%) for different hyperprior learners on *mini*ImageNet, *tiered*ImageNet, and FC100 (5-class). "Ind." and "Tra." denote inductive and transductive settings, respectively.

G   The inference time and the number of parameters

In Table S3, we supplement the the inference time and the number of parameters of baselines (100 epochs, *mini*ImageNet, 5-way 1-shot, on NVIDIA V100 GPU)

No.	Method	Backbone	# Param	Time (min)
1	MTL	ResNet-25	4,321k	73.3
2	MTL+E^{3}BM (ours)	ResNet-25	4,351k	77.6
3	SIB	WRN-28-10	36,475k	325.0
4	SIB+E^{3}BM (ours)	WRN-28-10	36,490k	331.8
5	ProtoNets	ResNet-12	12,424k	35.2
6	MatchNets	ResNet-12	12,424k	37.3
7	ProtoNets	ResNet-25	36,579k	70.5
7	ProtoNets	WRN-28-10	36,482k	350.1

Table S3. Supplementary to Table 1. The inference time and the number of parameters of baselines (100 epochs, *mini*ImageNet, 5-way 1-shot, on NVIDIA V100 GPU).

H   Executing the source code with PyTorch

We provide our PyTorch code at https://gitlab.mpi-klsb.mpg.de/yaoyaoliu/e3bm. To run this repository, we kindly advise you to install python 3.6 and PyTorch

1.2.0 with Anaconda. You may download Anaconda and read the installation instruction on the official website (https://www.anaconda.com/download/). Create a new environment and install PyTorch and torchvision on it:

```
conda create --name e3bm-pytorch python=3.6
conda activate e3bm-pytorch
conda install pytorch=1.2.0
conda install torchvision -c pytorch
```

Install other requirements:

```
pip install -r requirements.txt
```

Run meta-training with default settings (data and pre-trained model will be downloaded automatically):

```
python main.py -backbone resnet12 -shot 1 -way 5 -mode
    meta_train -dataset miniimagenet
python main.py -backbone resnet12 -shot 5 -way 5 -mode
    meta_train -dataset miniimagenet
python main.py -backbone resnet12 -shot 1 -way 5 -mode
    meta_train -dataset tieredimagenet
python main.py -backbone resnet12 -shot 5 -way 5 -mode
    meta_train -dataset tieredimagenet
```

Run pre-training with default settings:

```
python main.py -backbone resnet12 -mode pre_train -dataset
    miniimagenet
python main.py -backbone resnet12 -mode pre_train -dataset
    tieredimagenet
```
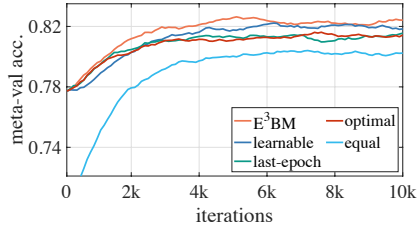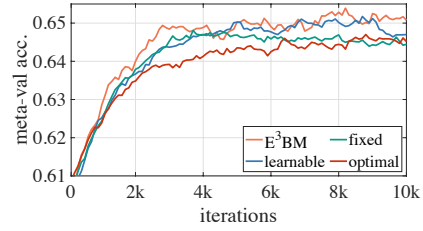
(a) v, *mini*ImageNet, 5-shot
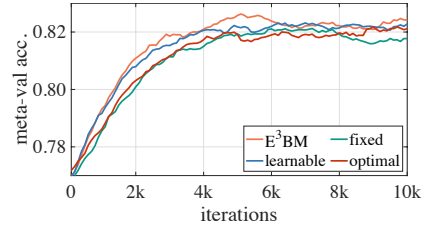
(b) α, *mini*ImageNet, 5-shot
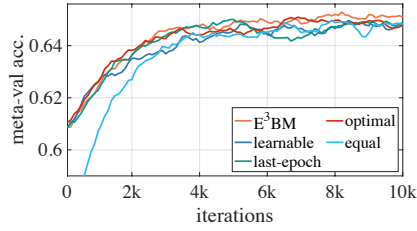
(c) v, *tiered*ImageNet, 1-shot

(d) α, *tiered*ImageNet, 1-shot
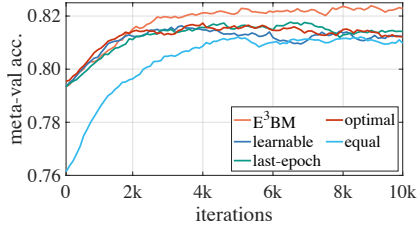
(e) v, *tiered*ImageNet, 5-shot
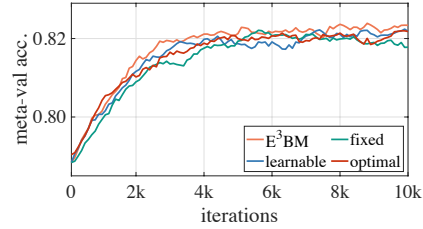
(f) α, *tiered*ImageNet, 5-shot
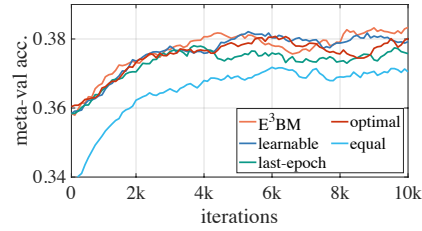
(g) v, FC100, 1-shot
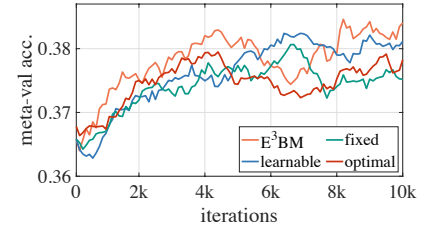
(h) α, FC100, 1-shot
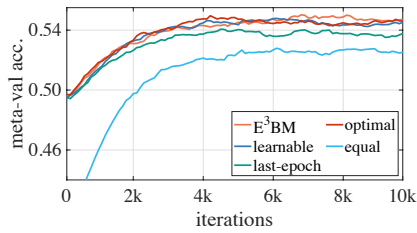
(i) v, FC100, 5-shot
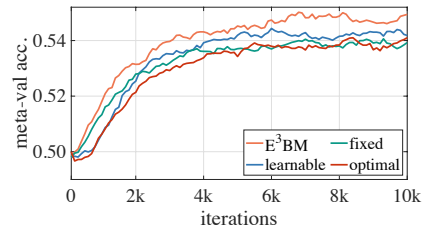
(j) α, FC100, 5-shot

Fig. S1. Supplementary to Fig. 4(a)(b). The meta-validation accuracies of ablation models. Each figure demonstrates the results using the same model "MTL+E^{3}BM" as in Table 1. All curves are smoothed with a rate of 0.9 for a better visualization.
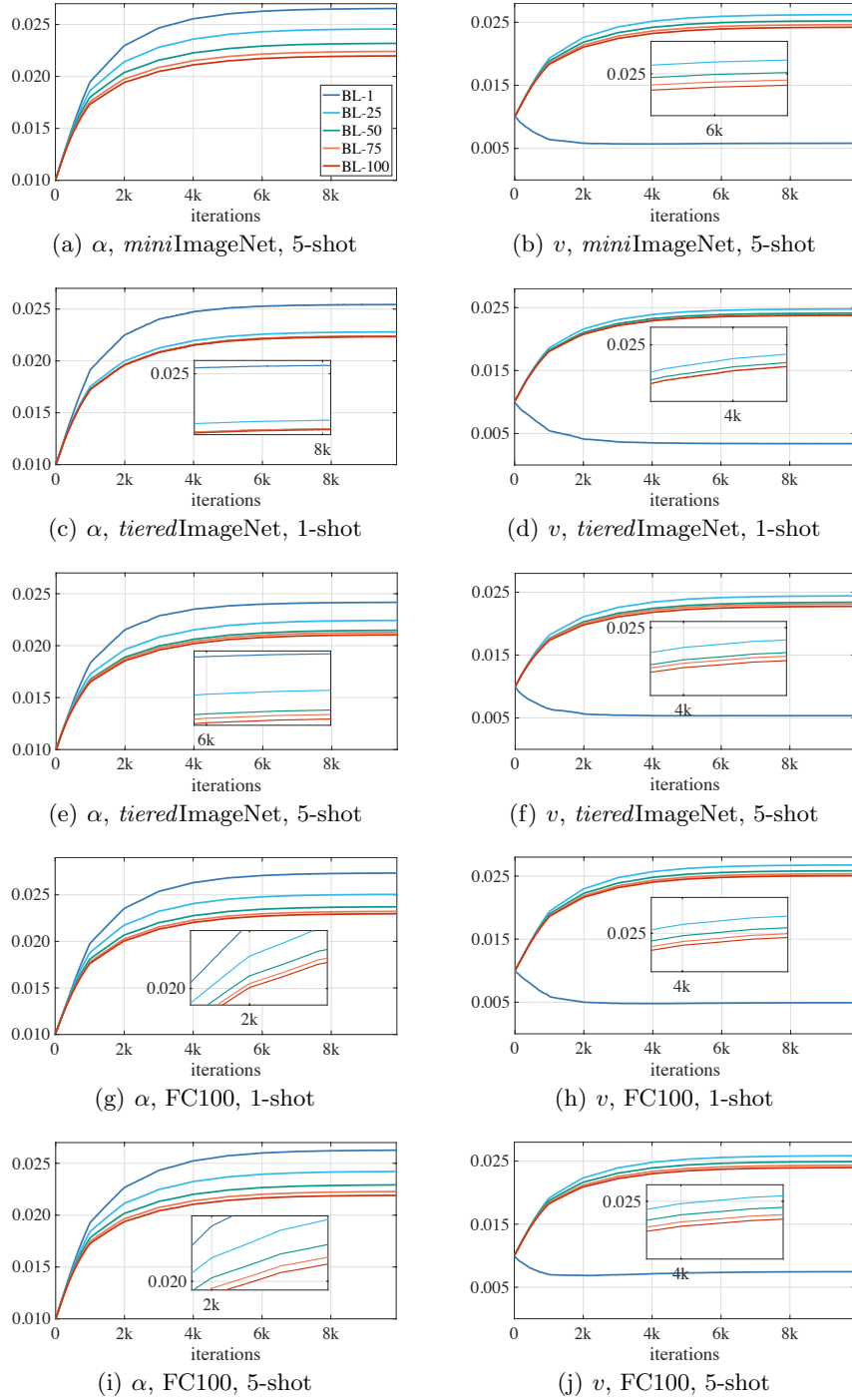
(a) α, *mini*ImageNet, 5-shot

(b) v, *mini*ImageNet, 5-shot

(c) α, *tiered*ImageNet, 1-shot

(d) v, *tiered*ImageNet, 1-shot

(e) α, *tiered*ImageNet, 5-shot

(f) v, *tiered*ImageNet, 5-shot

(g) α, FC100, 1-shot

(h) v, FC100, 1-shot

(i) α, FC100, 5-shot

(j) v, FC100, 5-shot

Fig. S2. Supplementary to Fig. 4(c)(d). The values of α and v generated by Ψ_α and Ψ_v, respectively. Each figure demonstrates the results using the same model "MTL+E^{3}BM" as in Table 1.