HYPERNETWORK APPROACH TO BAYESIAN MAML

Anonymous authors
Paper under double-blind review

ABSTRACT

The main goal of Few-Shot learning algorithms is to enable learning from small amounts of data. One of the most popular and elegant Few-Shot learning approaches is Model-Agnostic Meta-Learning (MAML). The main idea behind this method is to learn shared universal weights of a meta-model, which then are adapted for specific tasks. However, due to limited data size, the method suffers from over-fitting and poorly quantifies uncertainty. Bayesian approaches could, in principle, alleviate these shortcomings by learning weight distributions in place of point-wise weights. Unfortunately, previous Bayesian modifications of MAML are limited in a way similar to the classic MAML, e.g., task-specific adaptations must share the same structure and can not diverge much from the universal meta-model. Additionally, task-specific distributions are considered as posteriors to the universal distributions working as priors and optimizing them jointly with gradients is hard and poses a risk of getting stuck in local optima.

In this paper, we propose BayesHMAML, a novel generalization of Bayesian MAML, which employs Bayesian principles along with Hypernetworks for MAML. We achieve better convergence than the previous methods by classically learning universal weights. Furthermore, Bayesian treatment of the specific tasks enables uncertainty quantification, and high flexibility of task adaptations is achieved using Hypernetworks instead of gradient-based updates. Consequently, the proposed approach not only improves over the previous methods, both classic and Bayesian MAML in several standard Few-Shot learning benchmarks but also benefits from the properties of the Bayesian framework.

1 INTRODUCTION

Deep neural networks work perfectly when trained on large data sets. These, however, are rarely available in real-world settings. Hence, approaches able to learn from small amounts of data are needed. In particular, Few-Shot learning models can easily adapt to previously unseen tasks based on a few labeled samples. Among Few-Shot learning approaches, one of the most popular and elegant is Model-Agnostic Meta-Learning (MAML) Finn et al. (2017). The main idea behind this method is to produce universal weights which can be rapidly updated to solve new small tasks. However, limited data sets lead to two main problems. Firstly, the method tends to overfit training data, preventing us from using deep architectures with large numbers of weights. Secondly, it lacks good quantification of uncertainty, e.g., the model does not know how reliable its predictions are. Both problems can be addressed by employing Bayesian Neural Networks (BNNs) (MacKay, 1992; Jospin et al., 2022), which in place of point-wise estimates, learn distributions. BNNs may rely on the same network structure as the classic NNs. Still, their parameters (i.e., network's weights) have assumed prior distributions, which later are updated to posterior distributions when training data is observed. The Bayesian treatment allows for obtaining uncertain information principally and prevents over-fitting.

Bayesian modification has also been previously proposed for MAML. Bayesian MAML (Yoon et al., 2018) or Amortized bayesian meta-learning (Ravi & Beatson, 2018), PACOH (Rothfuss et al., 2021; 2020), FO-MAML (Nichol et al., 2018), MLAP-M (Amit & Meir, 2018), Meta-Mixture (Jerfel et al., 2019) learn distributions for the common universal weights, which are then updated to per-task local weights distributions. Although BNNs help these methods to regularize training and quantify uncertainty, they still suffer from the same shortages as the classic MAML, as well as from additional challenges coming from solving a harder modeling and optimization task (due to increased dimensionality and complexity of a variational objective).

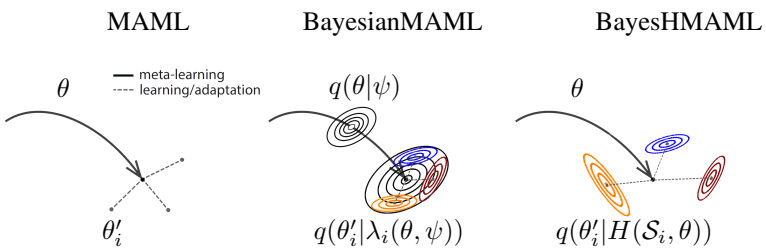

Figure 1: Visualization of MAML, BayesianMAML and BayesHMAML. For the classic MAML, we have universal weights θ which are then adjusted to θ'_i for individual tasks. For BayesianMAML, the posterior distributions for individual small tasks $q(\theta'|\lambda_i(\theta,\phi))$ are obtained in a few gradient-based updates from the universal distribution $q(\theta|\psi)$. For BayesHMAML, we learn the universal weights analogically as for MAML, but then, a hypernetwork $H(\mathcal{S}_i,\theta)$ produces parameters of the Bayesian posteriors. Such an approach enables significantly larger updates in the adaptation phase.

The previous Bayesian modifications of MAML (see the middle plot in Fig. 1), similar to the original MAML, rely on gradient-based updates. Weights specialized for small tasks are obtained by taking a fixed number of gradient steps from the common universal weights. Such the procedure binds tightly all the tasks by an overly rigid structure of possible solutions. The same effect is observed for the methods using the Bayesian approach, despite them acting on distributions instead of individual weights. All the specialized weight distributions must stay within a distance of a few gradient updates. Additionally, as proposed for example by Ravi & Beatson (2018) the universal distribution is usually employed as a prior for the per-task specializations, amplifying the above problem by furthermore limiting how the final weights for the small tasks may look like. Although such a hierarchical structure is not uncommon (see for example the work by (Amit & Meir, 2018)), it noticeably complicates the variational objective and the optimized loss surface. We argue that for practical problems with limited datasets, the benefits from the usage of the hierarchical model are outweighed by the optimization challenges.

A natural way to allow a more flexible structure of solutions and to enable better adaptations of weights is to employ non-gradient-based updates, for example, by using hypernetworks. Hypernetworks, introduced by Ha et al. (2016), are neural models, separate from a main model, which generate weights for it. In our paper, we propose to use a hypernetwork to adapt specialized per-task distributions starting from the universal weights. In particular, besides the main network, we have a side hypernetwork responsible for modeling Bayesian posteriors for individual tasks. Similar to the previous approaches, the final posterior is obtained by updating the universal weights, but otherwise, we propose multiple modifications. For instance, we avoid the aforementioned hierarchical structure by modeling the universal weights in a point-wise manner. Then, by using the hypernetwork in our amortization scheme, we also can allow arbitrary variances for the specialized distributions while at the same time remaining robust against overfitting. In our model, variances for the per-task weights are regularized by using the common hypernetwork. Hence, the distribution of the universal weights does not need to be used as a common prior and we achieve a simpler optimization objective. A schematic illustration of the approach, we present on the right-most plot in Fig. 1.

Our contributions can be summarized as follows:

- We introduce BayesHMAML, a novel Bayesian approach to the Few-Shot learning problem, which directly produces posterior distributions of weights specialized for small tasks by aggregating information from support sets and common universal weights.
- BayesHMAML handles universal weights and their updates in a new way: in the adaptation procedure, it transforms the classical neural network into its Bayesian counterpart.
- Compared to the previous Bayesian modifications to MAML, BayesHMAML by employing hypernetworks achieves significantly more flexible weight updates.
- To the best of our knowledge, the proposed solution is the first approach using the Hypernetwork paradigm for Bayesian Few-Shot learning.

2 BACKGROUND

In this section, we introduce all the notions necessary for understanding our method. First, we start by presenting background and notations for Few-Shot learning. Then we describe how the MAML algorithm works. Finally, we introduce general idea of Hypernetworks dedicated for MAML updates.

The terminology describing the Few-Shot learning setup is dispersive due to the colliding definitions used in the literature. Here, we use the nomenclature derived from the Meta-Learning literature, which is the most prevalent at the time of writing. Let $\mathcal{S} = \{(\mathbf{x}_l, \mathbf{y}_l)\}_{l=1}^{L}$ be a support-set containing input-output pairs, with L examples with the equal class distribution. In the *one-shot* scenario, each class is represented by a single example, and $L = K$, where K is the number of the considered classes in the given task. Whereas, for *Few-Shot* scenarios, each class usually has from 2 to 5 representatives in the support set $\mathcal{S}$.

Let $\mathcal{Q} = \{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^{M}$ be a query-set (sometimes referred to in the literature as a target-set), with M examples, where M is typically one order of magnitude greater than K. For clarity of notation, the support and query sets are grouped in a task $\mathcal{T} = \{\mathcal{S}, \mathcal{Q}\}$. During the training stage, the models for Few-Shot applications are fed by randomly selected examples from training set $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^{N}$, defined as a collection of such tasks.

During the inference stage, we consider task $\mathcal{T}_* = \{\mathcal{S}_*, \mathcal{X}_*\}$, where $\mathcal{S}_*$ is a support set with the known class values for a given task, and $\mathcal{X}_*$ is a set of query (unlabeled) inputs. The goal is to predict the class labels for query inputs $\mathbf{x} \in \mathcal{X}_*$, assuming support set $\mathcal{S}_*$ and using the model trained on $\mathcal{D}$.

Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) is one of the current standard algorithms for Few-Shot learning, which learns the parameters of a model so that it can adapt to a new task in a few gradient steps.

We consider a model represented by a parametrized function f_θ with parameters θ. In practice, the architecture consists of a future extractor (backbone) $E(\cdot)$ and one fully connected layer. The universal weights $\theta = (\theta^E, \theta^H)$ include θ^E for feature extractor and θ^H for classification head.

In adaptation to a new task $\mathcal{T}_i = \{\mathcal{S}_i, \mathcal{Q}_i\}$, all model's parameters θ are updated to θ_i'. Such an update is modeled by one or more gradient descent updates on task T_i. In the simplest case of one gradient update, the parameters are updated as follows:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

where the step size α is a hyperparameter and the loss function for a set of observations $\mathcal{Z}$ is defined as $\mathcal{L}_\mathcal{Z}$ for the few shot scenario is represented as a simple cross-entropy:

$$\mathcal{L}_\mathcal{Z}(f_\theta) = \sum_{(\mathbf{x}_{i,l}, \mathbf{y}_{i,l}) \in \mathcal{Z}} \sum_{k=1}^{K} -y_{i,l}^k \log f_{\theta,k}(\mathbf{x}_{i,j}),$$

where $f_{\theta,k}(\mathbf{x}_{i,j})$ denotes k-th output of the model f_θ, for a given input $\mathbf{x}_{i,l}$, and $\mathbf{y}_{i,l}$ is corresponding class in one-hot coding. For simplicity of notation, we will consider one gradient update for the rest of this section, but using multiple gradient updates is a straightforward extension.

The meta-optimization across tasks is performed via stochastic gradient descent (SGD):

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(T)} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

where β is the meta step size, see Fig. 1.

HyperMAML (Przewięźlikowski et al., 2022) is a generalization of the MAML algorithm which uses hypernetwork paradigm to model non-gradient based updates. We aim at predicting the class distribution $p(\mathbf{y}|\mathbf{x}_q, \mathcal{S})$, assuming given single query example $\mathbf{x}_q$, and the set of support examples $\mathcal{S}$.

Analogically to MAML we consider a model represented by a parameterized function f_θ with parameters θ. When adapting to a new task T_i, the model's parameters θ become θ_i'. In HyperMAML, contrary to MAML, the updated parameter vector θ_i' is computed using Hypernetwork. In practice,

Hypernetwork is a neural network that consists of feature extractor $E(\cdot)$, that transforms support set to low-dimensional representation and fully connected layers which aggregate such lower reprehension. Before aggregation, we add true labels and predictions given by universal weights. Hypernetwork produces an update for universal weights

$$\theta_i' = \theta + H_\phi(S_i, f_\theta(S_i)).$$

Analogically to MAML universal weights $\theta = (\theta_E, \theta_H)$ consist of θ_E from feature extractor and θ_H from classification head. But in HyperMAML we produce updates only for θ_H:

$$\theta_i' = (\theta_i'^E, \theta_i'^H) = (\theta_i'^E, \theta_i'^H + H_\phi(S_i, f_\theta(S_i)).$$

The meta-optimization across tasks is performed via stochastic gradient descent (SGD) such that the model parameters θ are updated as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(T)} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

where β is the meta step size.

3 BAYESIAN PERSPECTIVE ON MAML

In this section, we present BayesHMAML – a Bayesian extension of the classical MAML and discuss its use for Few-Shot learning.

A generic predictive model $p(y|f_{\theta'}(x))$ with latent weights θ' can be trained in a point-wise manner as explained above, which however may lead to over-fitting and takes into account only aleatoric uncertainty, entirely overlooking model uncertainty. The most straightforward Bayesian treatment for such a model is to pose priors for the model parameters and learn their posteriors. In particular, for MAML one needs to learn posterior distributions for both θ and θ'. However, the observed data $\mathcal{D}$ depends directly only on θ' (by $p(y|f_{\theta'}(x))$) and on θ only through θ'. This naturally hints towards a hierarchical Bayesian model: $\theta \to \theta_i' \to \mathcal{T}_i$, which indeed was previously proposed by Ravi & Beatson (2018) and later studied by Chen & Chen (2022). Since $f_{\theta'}(x)$ is an arbitrary neural network, posterior inference for such a model is intractable. Hence, the variational inference along with reparametrization gradients (i.e., Bayes by backpropagation (Blundell et al., 2015)) is typically used and the following objective (evidence lower bound) maximized w.r.t variational parameters λ_i and ψ:

$$\mathcal{L}_{\mathcal{D}} = \mathbb{E}_{q(\theta|\psi)} \left[\underbrace{\sum_i^N \mathbb{E}_{q(\theta_i'|\lambda_i)} \left[\log p(\mathcal{T}_i|\theta_i') - KL\left(q(\theta_i'|\lambda_i)|p(\theta_i'|\theta)\right)\right]}_{\mathcal{L}_{\mathcal{T}_i}} \right] - KL(q(\theta|\psi)|p(\theta))$$

where $q(\theta_i'|\lambda_i)$ and $q(\theta|\psi)$ are respectively per-task posterior approximation and approximate posterior for the universal weights. They are tied together by the prior $p(\theta_i'|\theta)$.

The above formulation raises a few challenges. Due to potentially large number of tasks $\mathcal{T}$ and their limited size (few data points per task) it is hard to choose the prior $p(\theta_i'|\theta)$, which would not dominate learning $q(\theta_i'|\lambda_i)$ for individual tasks and yet sufficiently modulate $q(\theta|\psi)$. The problem becomes even more prominent with an increasing number of tasks with the parameters λ_i learned separately for each of the tasks. A solution is amortized inference (Kingma & Welling, 2014) where instead of learning a separate λ_i for each task, $q(\theta_i'|\cdot)$ is conditioned on data.

Ravi & Beatson (2018) proposed a strategy inspired once again by the standard MAML, e.g., they used $q(\theta_i'|\lambda_i) = SGD(\mathcal{T}_i, \theta)$, where SGD denotes a few steps of optimization with the gradient $\nabla_{\lambda_i}\mathcal{L}_{\mathcal{T}_i}$, starting from the distribution for the universal parameters θ. From an implementation perspective, the difference is rather minor: instead of moving θ to θ_i' for MAML, for Amortized Bayesian MAML one now transforms parameters controlling the distribution for θ into parameters controlling the distribution for θ_i'. On the other hand, gradient-based optimization of the objective

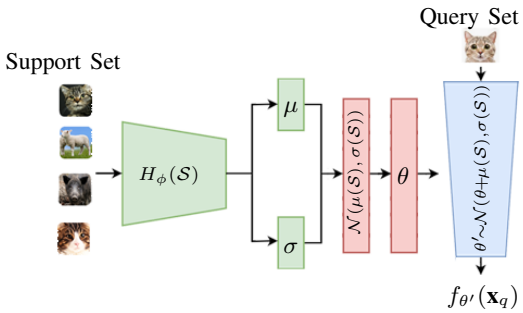

Figure 2: Schematic of our BayesHMAML model. Instead of updating the weights with gradient descent, we use Hypernetwork to aggregate information from the support set and production parameters for the probability distribution $\mathcal{N}(\mu(\mathcal{S}), \sigma(\mathcal{S}))$ dedicated to a specific task. We move such distribution by using universal weights θ to obtain final task specific distribution $\mathcal{N}(\theta{+}\mu(\mathcal{S}), \sigma(\mathcal{S}))$. Finally, we sample θ' and use a cost function containing cross-entropy and Kullback–Leibler regularization. In BayesHMAML, we have universal weight and Bayesian updates. Such an approach allows production to model significantly significant updates in the adaptation phase.

$\mathcal{L}_\mathcal{D}$ w.r.t jointly ψ and λ_i poses a strong risk of finding a local optimum and suboptimal solutions, especially when the variational parameters include the hard-to-fit distribution variances. Finally, the prior $p(\theta_i'|\theta)$ as specified in the original work, prevents $q(\theta_i'|\cdot)$ to diverge significantly from $q(\theta|\cdot)$ which may often impose a too strong regularization.

3.1 BayesHMAML: Bayesian Hypernetwork for Few-Shot learning

In this work, we propose an alternative approach that alleviates the problems of the previous attempts at Bayesian MAML. In particular, the posterior $q(\theta|\cdot)$ for the universal parameters serves no purpose beyond regularizing per-task posterior approximations (via the prior $p(\theta_i'|\theta)$ and by providing initialization for the SGD updates) but as explained above may complicate the optimization task. For example, Ravi & Beatson (2018) hinted toward considering simple distributions (e.g. delta distributions, which however could be overly limiting for $q(\theta_i'|\cdot)$). Hence, in our method, we learn θ in a point-wise manner instead. In particular, we don't learn variance for the universal parameters θ, but we learn them for individual θ_i' independently. Furthermore, we remove the coupling prior between θ and θ_i' and propose a basic non-hierarchical prior $p(\theta_i')$ instead. All these modifications simplify the optimization landscape and, taken together along with our Hypernetwork-based adjustment strategy (details below) should enable better optima for our objective:

$$\mathcal{L}_D^{our} = \sum_i^N \mathbb{E}_{q(\theta_i'|\lambda_i(\theta,\mathcal{S}_i))} \left[\log p(\mathcal{T}_i|\theta_i') - \gamma \cdot KL\left(q(\theta_i'|\lambda_i(\theta,\mathcal{S}_i))|p(\theta_i')\right)\right]$$

In practice, we use the standard normal priors for the weights of the neural network f, i.e., $p(\theta_i') = \mathcal{N}(\theta_i'|0, \mathbb{I})$, and the hyperparameter γ allows controlling impact of the priors and compensating for model mispecification.

The key component of BayesHMAML is however our amortization scheme, e.g., implementation details of $q(\theta_i'|\lambda_i(\theta, \mathcal{S}_i))$, see Fig. 2. In particular, we propose $q(\theta_i'|\lambda_i(\theta, \mathcal{S}_i)) = \mathcal{N}(\theta_i'|\theta + \mu_i(\mathcal{S}_i), \sigma_i(\mathcal{S}_i))$, where $\mu_i(\mathcal{S}_i)$ and $\sigma_i(\mathcal{S}_i)$ are outputs of a hypernetwork H_ϕ. Parameters of the posterior approximation $q(\theta_i'|\cdot)$ are constructed by combining the point-wise learned universal parameters θ and outputs of the hypernetwork (for the mean of the distribution) or just directly from the hypernetwork outputs (for the variance/standard deviation parameter). Optimization of $\mathcal{L}_D^{our}$ is performed w.r.t to θ and hypernetwork weights ϕ, which have fixed sizes and do not grow with the number of tasks N. Hence, BayesHMAML scales well. Also, hypernetworks can provide flexible adjustments and BayesHMAML better than the previous methods adapt for individual tasks.

Optimization details are presented in Algorithm 1. We use learning with minibatches and applied an annealing scheme for the hyperparameter γ (Bowman et al., 2016). For a batch of meta-task, our hypernetwork aggregates information from the support set and produces parameters of probability distribution dedicated to a specific task $(\mu(\mathcal{S}_i), \sigma(\mathcal{S}_i)) = H_\phi(S_i, f_\theta(S_i))$. In Bayesian training, we sample weights from updated distributions $\theta_i' \sim \mathcal{N}(\theta + \mu(S_i), \sigma(S_i))$. For simplicity of notation, we

Algorithm 1 BayesHMAML

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, γ: step size hyper parameters

while not done **do**
  Sample task $\mathcal{T}_i \sim p(\mathcal{T})$
  for each $\mathcal{T}_i$ **do**
    Compute probability distribution of adapted parameters:
    $(\mu(\mathcal{S}_i), \sigma(\mathcal{S}_i)) = H_\phi(S_i, f_\theta(S_i))$
    Sample $\theta_i' \sim \mathcal{N}(\theta + \mu(S_i), \sigma(S_i))$
  end for
  Update
  $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_\mathcal{T}$
  $\phi \leftarrow \phi - \beta \nabla_\phi \mathcal{L}_\mathcal{T}$
  where $\mathcal{L}_\mathcal{T} = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) - \gamma KL[\mathcal{N}(\theta + \mu(\mathcal{S}_i), \sigma(\mathcal{S}_i)), \mathcal{N}(0, \mathbb{I})] \right]$

will consider one sample for the rest of this section, but in practice, we use multiple samples. The number of samples is a hyperparameter.

The meta-optimization across tasks is obtained by minimization of cross-entropy loss and Kullback–Leibler regularization:

$$\mathcal{L}_\mathcal{T}^{our} = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) - \gamma KL[\mathcal{N}(\theta + \mu(\mathcal{S}_i), \sigma(\mathcal{S}_i)), \mathcal{N}(0, \mathbb{I})] \right],$$

where parameter γ changes from zero to fixed constant during training. The final value of γ is also a hyperparameter of the model.

4 Related Work

The problem of Meta-Learning and Few-Shot learning (Hospedales et al., 2020; Schmidhuber, 1992; Bengio et al., 1992) is currently one of the most important topics in deep learning, with the abundance of methods emerging as a result. They can be roughly categorized into three groups: Model-based methods, Metric-based methods, Optimization-based methods. In all these groups, we can find methods that use Hypernetworks and Bayesian models. We briefly describe such approaches. To the best of our knowledge, it is the first approach that uses the Hypernetwork paradigm for bayesian few-shot learning. In the end, we concentrate on Bayesian models proposed for few-shot learning models.

Model-based methods aim to adapt to novel tasks quickly by utilizing mechanisms such as memory (Ravi & Larochelle, 2017; Santoro et al., 2016; Mishra et al., 2018; Zhen et al., 2020), Gaussian Processes (Rasmussen, 2003; Patacchiola et al., 2020; Wang et al., 2021; Sendera et al., 2021), or generating fast weights based on the support set with set-to-set architectures (Qiao et al., 2017; Bauer et al., 2017; Ye et al., 2018; Zhmoginov et al., 2022). Other approaches maintain a set of weight templates and, based on those, generate target weights quickly through gradient-based optimization such as (Zhao et al., 2020). The fast weights approaches can be interpreted as using Hypernetworks (Ha et al., 2016) – models which learn to generate the parameters of neural networks performing the designated tasks.

Metric-based methods learn a transformation to a feature space where the distance between examples from the same class is small. The earliest examples of such methods are Matching Networks (Vinyals et al., 2016) and Prototypical Networks (Snell et al., 2017). Subsequent works show that metric-based approaches can be improved by techniques such as learnable metric functions (Sung et al., 2018), conditioning the model on tasks (Oreshkin et al., 2018) or predicting the parameters of the kernel function to be calculated between support and query data with Hypernetworks (Sendera et al., 2022).

Optimization-based methods such as MetaOptNet (Lee et al., 2019) is based on the idea of an optimization process over the support set within the Meta-Learning framework. Arguably, the most

popular of this family of methods is Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017). In literature, we have various techniques for stabilizing its training and improving performance, such as Multi-Step Loss Optimization (Antoniou et al., 2018), or using the Bayesian variant of MAML (Yoon et al., 2018).

Due to a need for calculating second-order derivatives when computing the gradient of the meta-training loss, training the classical MAML introduces a significant computational overhead. The authors show that in practice, the second-order derivatives can be omitted at the cost of small gradient estimation error and minimally reduced accuracy of the model (Finn et al., 2017; Nichol et al., 2018). Methods such as iMAML and Sign-MAML propose to solve this issue with implicit gradients or Sign-SGD optimization (Rajeswaran et al., 2019; Fan et al., 2021). The optimization process can also be improved by training the base initialization (Munkhdalai & Yu, 2017; Li et al., 2017; Rajasegaran et al., 2020). In (Przewięźlikowski et al., 2022) propose a generalization of the MAML algorithm, which uses hypernetwork paradigm to model non-gradient based updates.

Bayesian approaches Classical MAML-based algorithms have problems with over-fitting. To solve a such problem we can use the Bayesian model (Ravi & Beatson, 2018; Yoon et al., 2018; Grant et al., 2018; Jerfel et al., 2019; Nguyen et al., 2020). In practice, the Bayesian model contains two levels of probability distribution on weights. We have Bayesian universal weights, which are updated for different tasks Grant et al. (2018). Its leads to a hierarchical Bayes formulation. Bayesian networks perform better in few-shot settings and reduce over-fitting. Several variants of the hierarchical Bayes model have been proposed based on different Bayesian inference methods (Finn et al., 2018; Yoon et al., 2018; Gordon et al., 2018; Nguyen et al., 2020). Another branch of probabilistic methods is represented by PAC-Bayes based method (Chen & Chen, 2022; Amit & Meir, 2018; Rothfuss et al., 2021; 2020; Ding et al., 2021; Farid & Majumdar, 2021). In the PAC-Bayes framework, we use the Gibbs error when sampling priors. But still, we have a double level of Bayesian networks.

In the paper, we propose BayesHMAML, which uses probability distribution only for weight dedicated for small tasks. Thanks to such a solution, we produce significantly larger updates.

5 EXPERIMENTS

In the typical Few-Shot learning setting, making a valuable and fair comparison between proposed models is often complicated because of the significant differences in architectures and implementations of known methods. To limit the influence of the deeper backbone (feature extractor) architectures, we follow the unified procedure proposed by Chen et al. (2019)[1].

In all the reported experiments, the tasks consist of 5 classes (5-way) and 1 or 5 support examples (1 or 5-shot). Unless indicated otherwise, all compared models use a known and widely utilized backbone consisting of four convolutional layers (each consisting of a 2D convolution, a batch-norm layer, and a ReLU non-linearity; each layer consists of 64 channels) and have been trained from scratch Chen et al. (2019). In all experiments, the query set of each task consists of 16 samples for each class (80 in total). We split the data sets into the standard train, validation, and test class subsets, used commonly in the literature Ravi & Larochelle (2017); Chen et al. (2019); Patacchiola et al. (2020). We report the performance of BayesHMAML averaged over three training runs for each setting. We provide the additional training details in the Appendix.

We report the performance of two variants of BayesHMAML:

- **BayesHMAML** - Bayesian models generated by the hypernetworks for each task.
- **BayesHMAML + adaptation** - Bayesian models generated by hypernetworks adapted to the support examples of each task with a few training steps.

In the case of **BayesHMAML + adaptation**, we tune a copy of the hypernetwork on the support set separately for each validation task. This way, we ensure that our model does not take unfair advantage of the validation tasks. In the case of hypernetwork-based approaches, such adaptation is a common strategy introduced by Sendera et al. (2022).

[1]We shall release the code with our experiments after the end of the review period.

5.1 CLASSIFICATION

First, we consider a classical Few-Shot learning scenario. We benchmark the performance of the BayesHMAML and other methods on two challenging and widely considered data sets: Caltech-USCD Birds (**CUB**) Wah et al. (2011) and **mini-ImageNet** Ravi & Larochelle (2017). The following experiments are in the most popular setting, 5-way, with five random classes. We compare BayesHMAML to a vast pool of state-of-the-art algorithms in the tasks of 1-shot and 5-shot classification.

In **CUB**, we have the second score in 1-shot and 5-shot. In the case of **mini-ImageNet** we have comparable results to other methods. It should be highlighted that we obtained the best score in the area of Bayesian and the MAML-based method.

Table 1: The classification accuracy results for the inference tasks on **CUB** and **mini-ImageNet** data sets in the 1-shot and 5-shot settings. The highest results are in bold and the second-highest in italic (the larger, the better).

	CUB		**mini-ImageNet**	
Method	1-shot	5-shot	1-shot	5-shot
ML-LSTM Ravi & Larochelle (2017)	–	–	43.44 ± 0.77	60.60 ± 0.71
LLAMA Grant et al. (2018)	–	–	49.40 ± 1.83	
VERSA Gordon et al. (2018)	–	–	48.53 ± 1.84	67.37 ± 0.86
Amortized VI Gordon et al. (2018)	–	–	44.13 ± 1.78	55.68 ± 0.91
Meta-Mixture Jerfel et al. (2019)	–	–	49.60 ± 1.50	64.60 ± 0.92
Feature Transfer Zhuang et al. (2020)	46.19 ± 0.64	68.40 ± 0.79	39.51 ± 0.23	60.51 ± 0.55
Baseline++ Chen et al. (2019)	61.75 ± 0.95	78.51 ± 0.59	47.15 ± 0.49	66.18 ± 0.18
ProtoNet Snell et al. (2017)	52.52 ± 1.90	75.93 ± 0.46	44.19 ± 1.30	64.07 ± 0.65
RelationNet Sung et al. (2018)	62.52 ± 0.34	78.22 ± 0.07	48.76 ± 0.17	64.20 ± 0.28
DKT + BNCosSim Patacchiola et al. (2020)	62.96 ± 0.62	77.76 ± 0.62	49.73 ± 0.07	64.00 ± 0.09
VAMPIRE Nguyen et al. (2020)	–	–	51.54 ± 0.74	64.31 ± 0.74
ABML Ravi & Beatson (2018)	49.57 ± 0.42	68.94 ± 0.16	45.00 ± 0.60	–
FO-MAML Nichol et al. (2018)	–	–	48.70 ± 1.84	63.11 ± 0.92
Reptile Nichol et al. (2018)	–	–	49.97 ± 0.32	65.99 ± 0.58
HyperShot Sendera et al. (2022)	65.27 ± 0.24	79.80 ± 0.16	52.42 ± 0.46	68.78 ± 0.29
HyperShot+ adaptation Sendera et al. (2022)	66.13 ± 0.26	80.07 ± 0.22	53.18 ± 0.45	69.62 ± 0.2
FEAT Ye et al. (2018)	$\mathbf{68.87 \pm 0.22}$	$\mathbf{82.90 \pm 0.15}$	$\mathbf{55.15 \pm 0.20}$	$\mathbf{71.61 \pm 0.16}$
MAML Finn et al. (2017)	56.11 ± 0.69	74.84 ± 0.62	45.39 ± 0.49	61.58 ± 0.53
MAML++ Antoniou et al. (2018)	–	–	52.15 ± 0.26	68.32 ± 0.44
iMAML-HF Rajeswaran et al. (2019)	–	–	49.30 ± 1.88	–
SignMAML Fan et al. (2021)	–	–	42.90 ± 1.50	60.70 ± 0.70
Bayesian MAML Yoon et al. (2018)	55.93 ± 0.71		53.80 ± 1.46	64.23 ± 0.69
Unicorn-MAML Ye & Chao (2021)	–	–	54.89	–
Meta-SGD Li et al. (2017)	–	–	50.47 ± 1.87	64.03 ± 0.94
PAMELA Rajasegaran et al. (2020)	–	–	53.50 ± 0.89	$\mathit{70.51 \pm 0.67}$
HyperMAML Przewięźlikowski et al. (2022)	66.11 ± 0.28	78.89 ± 0.19	51.84 ± 0.57	66.29 ± 0.43
BayesHMAML	66.57 ± 0.47	79.86 ± 0.31	52.54 ± 0.46	67.39 ± 0.35
BayesHMAML + adaptation	$\mathit{66.92 \pm 0.38}$	$\mathit{80.47 \pm 0.38}$	52.69 ± 0.38	68.24 ± 0.47

Table 2: The classification accuracy results for the inference tasks on cross-domain tasks (**Omniglot→EMNIST** and **mini-ImageNet→CUB**) data sets in the 1-shot and 5-shot setting. The highest results are bold and second-highest in italic (the larger, the better).

	Omniglot→EMNIST		**mini-ImageNet→CUB**	
Method	**1-shot**	**5-shot**	**1-shot**	**5-shot**
Feature Transfer Zhuang et al. (2020)	64.22 ± 1.24	86.10 ± 0.84	32.77 ± 0.35	50.34 ± 0.27
Baseline++ Chen et al. (2019)	56.84 ± 0.91	80.01 ± 0.92	39.19 ± 0.12	57.31 ± 0.11
ProtoNet Snell et al. (2017)	72.04 ± 0.82	87.22 ± 1.01	33.27 ± 1.09	52.16 ± 0.17
RelationNet Sung et al. (2018)	75.62 ± 1.00	87.84 ± 0.27	37.13 ± 0.20	51.76 ± 1.48
DKT Patacchiola et al. (2020)	75.40 ± 1.10	$\mathit{90.30 \pm 0.49}$	$\mathbf{40.14 \pm 0.18}$	56.40 ± 1.34
HyperShot Sendera et al. (2022)	78.06 ± 0.24	89.04 ± 0.18	39.09 ± 0.28	$\mathit{57.77 \pm 0.33}$
HyperShot + adaptation Sendera et al. (2022)	80.65 ± 0.30	$\mathbf{90.81 \pm 0.16}$	$\mathit{40.03 \pm 0.41}$	$\mathbf{58.86 \pm 0.38}$
OVE PG GP + Cosine (ML) Snell & Zemel (2020)	68.43 ± 0.67	86.22 ± 0.20	39.66 ± 0.18	55.71 ± 0.31
OVE PG GP + Cosine (PL) Snell & Zemel (2020)	77.00 ± 0.50	87.52 ± 0.19	37.49 ± 0.11	57.23 ± 0.31
MAML Finn et al. (2017)	74.81 ± 0.25	83.54 ± 1.79	34.01 ± 1.25	48.83 ± 0.62
Bayesian MAML Yoon et al. (2018)	63.94 ± 0.47	65.26 ± 0.30	33.52 ± 0.36	51.35 ± 0.16
HyperMAML Przewięźlikowski et al. (2022)	79.07 ± 1.09	89.22 ± 0.78	36.32 ± 0.61	49.43 ± 0.14
BayesHMAML	$\mathit{80.95 \pm 0.46}$	89.21 ± 0.27	36.90 ± 0.34	49.24 ± 0.38
BayesHMAML + adaptation	$\mathbf{81.05 \pm 0.47}$	89.76 ± 0.26	37.23 ± 0.44	50.79 ± 0.59

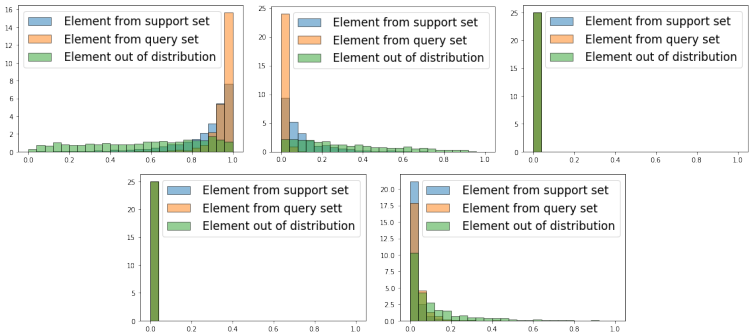

Figure 3: We train BayesHMAML on cross-domain adaptation setting **Omniglot→EMNIST**. Then we sample one thousand different weights from the distribution dedicated to the support set. We present predictions of BayesHMAML in three cases: for an element from the support set, an element from the query set, and an element from **EMNIST** but from classes that were not sampled for the support set. As we can see for elements from support and query sets, our model always gives a similar prediction. We have high uncertainty in the case of elements from out of distribution.

5.2 Cross-domain adaptation

In the cross-domain adaptation setting, the model is evaluated on tasks from a different distribution than the one on which it had been trained. Therefore, such a task is more challenging than standard classification and is a plausible indicator of a model's ability to generalize. To benchmark the performance of BayesHMAML in cross-domain adaptation, we combine two data sets so that the training fold is drawn from the first data set and validation and the testing fold – from another. We report the results in Table 2. In the task of 1-shot **Omniglot→EMNIST** classification, BayesHMAML achieves the best result. In the 5-shot **Omniglot→EMNIST** classification task BayesHMAML yields comparable results to baseline methods. In **mini-ImageNet→CUB** classification, our method performs comparably to baseline methods such as MAML, ProtoNet and Matching Net. Again we obtained the best score in the area of Bayesian and the MAML-based method.

5.3 Ablation studies: uncertainty

One of the most important advantages of a Bayesian Neural network is the natural uncertainty of the model. To visualize it, we train BayesHMAML on cross-domain adaptation setting **Omniglot→EMNIST**. We sample testing tasks from **EMNIST** during the evaluation. Then we sample one thousand different weights from the distribution dedicated to our support set. In Fig. 3 we present predictions of BayesHMAML in three cases: for an element from the support set, an element from the query set, and an element out of the distribution of the support set (element from **EMNIST** but from classes which was not sampled for support set). As we can see, our model always gives a similar prediction for elements from support and query sets. In the case of elements from out of distribution, we have high uncertainty.

6 Conclusions

In this work, we introduced BayesHMAML – a novel Bayesian Meta-Learning algorithm strongly motivated by MAML. In BayesHMAML, we have universal weight trained analogically to MAML in a point-wise manner and Bayesian updates. Such an approach allows for significantly larger updates in the adaptation phase. Our experiments show that BayesHMAML outperforms all Bayesian and MAML-based methods in several standard Few-Shot learning benchmarks and in most cases achieves results better or comparable to various other state-of-the-art methods. What is crucial, BayesHMAML can be used to estimate uncertainty of the prediction effectively, enabling possible applications in critical areas of deep learning, such as medical diagnosis or autonomous driving.

REFERENCES

Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended pac-bayes theory. In *International Conference on Machine Learning*, pp. 205–214. PMLR, 2018.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml, 2018. URL https://arxiv.org/abs/1810.09502.

Matthias Bauer, Mateo Rojas-Carulla, Jakub Bartlomiej Świątkowski, Bernhard Schölkopf, and Richard E. Turner. Discriminative k-shot learning using probabilistic models, 2017.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. 1992.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*, pp. 10–21. Association for Computational Linguistics (ACL), 2016.

Lisha Chen and Tianyi Chen. Is bayesian model-agnostic meta learning better than model-agnostic meta learning, provably? In *International Conference on Artificial Intelligence and Statistics*, pp. 1733–1774. PMLR, 2022.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Nan Ding, Xi Chen, Tomer Levinboim, Sebastian Goodman, and Radu Soricut. Bridging the gap between practice and pac-bayes theory in few-shot meta-learning. *Advances in Neural Information Processing Systems*, 34:29506–29516, 2021.

Chen Fan, Parikshit Ram, and Sijia Liu. Sign-maml: Efficient model-agnostic meta-learning by signsgd. *CoRR*, abs/2109.07497, 2021. URL https://arxiv.org/abs/2109.07497.

Alec Farid and Anirudha Majumdar. Generalization bounds for meta-learning via pac-bayes and uniform stability. *Advances in Neural Information Processing Systems*, 34:2173–2186, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2018.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

Ghassen Jerfel, Erin Grant, Thomas L Griffiths, and Katherine Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 9122–9133, 2019.

Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning, 2017.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pp. 2554–2563. PMLR, 2017.

Cuong Nguyen, Thanh-Toan Do, and Gustavo Carneiro. Uncertainty in model-agnostic meta-learning using variational inference. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3090–3100, 2020.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *arXiv preprint arXiv:1805.10123*, 2018.

Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33, 2020.

M Przewięźlikowski, P Przybysz, J Tabor, M Zięba, and P Spurek. Hypermaml: Few-shot adaptation of deep models with hypernetworks. *arXiv preprint arXiv:2205.15745*, 2022.

Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan Yuille. Few-shot image recognition by predicting parameters from activations, 2017.

Jathushan Rajasegaran, Salman H. Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. Meta-learning the learning trends shared across tasks. *CoRR*, abs/2010.09291, 2020. URL `https://arxiv.org/abs/2010.09291`.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in Neural Information Processing Systems*, 32:113–124, 2019.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003.

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2018.

Sachin Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Jonas Rothfuss, Martin Josifoski, and Andreas Krause. Meta-learning bayesian neural network priors based on pac-bayesian theory. 2020.

Jonas Rothfuss, Vincent Fortuin, Martin Josifoski, and Andreas Krause. Pacoh: Bayes-optimal meta-learning with pac-guarantees. In *International Conference on Machine Learning*, pp. 9116–9126. PMLR, 2021.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp. 1842–1850. PMLR, 2016.

Jürgen Schmidhuber. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation*, 4(1):131–139, 01 1992. ISSN 0899-7667.

Marcin Sendera, Jacek Tabor, Aleksandra Nowak, Andrzej Bedychaj, Massimiliano Patacchiola, Tomasz Trzcinski, Przemysław Spurek, and Maciej Zieba. Non-gaussian gaussian processes for few-shot regression. *Advances in Neural Information Processing Systems*, 34:10285–10298, 2021.

Marcin Sendera, Marcin Przewięźlikowski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. Hypershot: Few-shot learning by kernel hypernetworks. *arXiv preprint arXiv:2203.11378*, 2022.

Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each pólya-gamma augmented gaussian processes. In *International Conference on Learning Representations*, 2020.

Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy 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.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

Ze Wang, Zichen Miao, Xiantong Zhen, and Qiang Qiu. Learning to learn dense gaussian processes for few-shot learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Han-Jia Ye and Wei-Lun Chao. How to train your maml to excel in few-shot classification, 2021.

Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Learning embedding adaptation for few-shot learning. *CoRR*, abs/1812.03664, 2018. URL `http://arxiv.org/abs/1812.03664`.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 7343–7353, 2018.

Dominic Zhao, Seijin Kobayashi, João Sacramento, and Johannes von Oswald. Meta-learning via hypernetworks. 12 2020.

Xiantong Zhen, Ying-Jun Du, Huan Xiong, Qiang Qiu, Cees Snoek, and Ling Shao. Learning to learn variational semantic memory. In *NeurIPS*, 2020.

Andrey Zhmoginov, Mark Sandler, and Max Vladymyrov. Hypertransformer: Model generation for supervised and semi-supervised few-shot learning, 2022.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.

A Appendix: Training details

In this section, we present details of the training and architecture overview.

A.1 Architecture overview

The architecture of BayesHMAML consists of the following parts:

Encoder For each experiment described in the main body of this work, we utilize a shallow convolutional encoder (feature extractor), commonly used in the literature Finn et al. (2017); Chen et al. (2019); Patacchiola et al. (2020). This encoder consists of four convolutional layers, each consisting of a convolution, batch normalization, and ReLU nonlinearity. Each of the convolutional layers has an input and output size of 64, except for the first layer, where the input size is equal to the number of image channels. We also apply max-pooling between each convolution, by which the resolution of the processed feature maps is decreased by half. The output of the encoder is flattened to process it in the next layers.

In the case of the **Omniglot** and **EMNIST** images, such encoder compresses the images into 64-element embedding vectors, which serve as input to the Hypernetwork (with the above-described enhancements) and the classifier. However, in the case of substantially larger **mini-ImageNet** and **CUB** images, the backbone outputs a feature map of shape $[64 \times 5 \times 5]$, which would translate to 1600-element embeddings and lead to an over parametrization of the Hypernetwork and classifier which processes them and increase the computational load. Therefore, we apply an average pooling operation to the obtained feature maps and ultimately also obtain embeddings of shape 64. Thus, we can use significantly smaller Hypernetworks.

Hypernetwork The Hypernetwork transforms the enhanced embeddings of the support examples of each class in a task into the updates for the portion of classifier weights predicting that class. It consists of three fully-connected layers with ReLU activation function between each consecutive pair of layers. In the hypernetwork, we use a hidden size of 256 or 512.

Classifier The universal classifier is a single fully-connected layer with the input size equal to the encoder embedding size (in our case 64) and the output size equal to the number of classes. When using the strategy with embeddings enhancement, we freeze the classifier to get only the information about the behavior of the classifier. This means we do not calculate the gradient for the classifier in this step of the forward pass. Instead, gradient calculation for the classifier takes place during the classification of the query data.

A.2 Training details

In all of the experiments described in the main body of this work, we utilize the switch and the embedding enhancement mechanisms. During training, we use the Adam optimizer and the MultiStepLR learning rate scheduler with the decay of 0.3 and learning rate starting from 0.01 or 0.001. We train BayesHMAML for 4000 epochs on all the data sets, save for the simpler **Omniglot** $\rightarrow$ **EMNIST** classification task, where we train for 2048 epochs instead.

A.3 Hyperparameters

Below, we outline the hyperparameters of architecture and training procedures used in each experiment.

hyperparameter	**CUB**	**mini-ImageNet**	**mini-ImageNet** $\rightarrow$ **CUB**	**Omniglot** $\rightarrow$ **EMNIST**
learning rate	0.01	0.001	0.001	0.01
Hyper Network depth	3	3	3	3
Hyper Network width	512	256	256	512
epochs no.	4000	4000	4000	2048
milestones	$51, 550$	$101, 1100$	$101, 1100$	$51, 550$
γ	$1e-4$	$1e-4$	$1e-5$	0.001
weight set num. (train)	5	7	5	5

Table 3: Hyperparameters for each of conducted 1-shot experiments.

hyperparameter	**CUB**	**mini-ImageNet**	**mini-ImageNet → CUB**	**Omniglot → EMNIST**
learning rate	0.001	0.001	0.001	0.01
Hyper Network depth	3	3	3	3
Hyper Network width	256	256	256	512
epochs no.	4000	4000	4000	2048
milestones	$101, 1100$	$101, 1100$	$101, 1100$	$51, 550$
γ	$1e-5$	$1e-5$	$1e-4$	0.001
weight set num. (train)	5	5	5	5

Table 4: Hyperparameters for each of the conducted 5-shot experiments.