HYPERMAML: FEW-SHOT ADAPTATION OF DEEP MODELS WITH HYPERNETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The aim of Few-Shot learning methods is to train models which can easily adapt to previously unseen tasks, based on 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 the general weights of the meta-model, which are further adapted to specific problems in a small number of gradient steps. However, the model's main limitation lies in the fact that the update procedure is realized by gradient-based optimisation. In consequence, MAML cannot always modify weights to the essential level in one or even a few gradient iterations. On the other hand, using many gradient steps results in a complex and time-consuming optimization procedure, which is hard to train in practice, and may lead to overfitting. In this paper, we propose HyperMAML, a novel generalization of MAML, where the training of the update procedure is also part of the model. Namely, in HyperMAML, instead of updating the weights with gradient descent, we use for this purpose a trainable Hypernetwork. Consequently, in this framework, the model can generate significant updates whose range is not limited to a fixed number of gradient steps. Experiments show that HyperMAML outperforms MAML in most cases and performs comparably to other state-of-the-art techniques in a number of standard Few-Shot learning benchmarks.

1 INTRODUCTION

In the typical Few-Shot learning setting, the aim is to adapt to new tasks under the assumption that only a few examples are given. As we know, people typically learn new tasks easily by using only a few training examples. On the contrary, a standard deep neural network must be trained on an extensive amount of data to obtain a similar accuracy. Thus, the aim of Few-Shot learning models is to bring neural networks closer to the human brain's capabilities. The most famous and, in our opinion, the most elegant approach to Few-Shot learning is Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), where the model is trained to adapt universal weights to new Few-Shot learning tasks quickly. It seems that the brain's neural networks can adapt to new tasks too, by applying the fact that during the process of evolution, some of its parts have developed universal weights which are easily adaptable to typical tasks we encounter in real life. Thus, the idea behind MAML gives us a possible insight into the working of the brain.

The fascinating factor of human intelligence is that the human learning process, although still not understood, is clearly not based on the gradient descent algorithm, as we cannot in general backpropagate the information (Lillicrap et al., 2020; Song et al., 2020; Whittington & Bogacz, 2019). Thus, from the biological point of view, the main limitation of MAML is the fact that it uses the gradient descent method for weight updates. The main research problem that we set for ourselves is whether one can modify MAML to be more biologically feasible, i.e. keep its ability to find universal weight but remove the necessity of using gradient-based update methods.

We solve this problem by constructing HyperMAML, a model which replaces the gradient optimization in the update of weights by trainable update procedure with the use of the Hypernetwork paradigm. Hypernetworks, introduced in (Ha et al., 2016) are defined as neural models that generate weights for a separate target network solving a specific task. In our model, HyperMAML, the Hypernetwork aggregates the information from the support set and produces an update to the main model. Thanks to such an approach, we can create various types of updates that are not limited to a

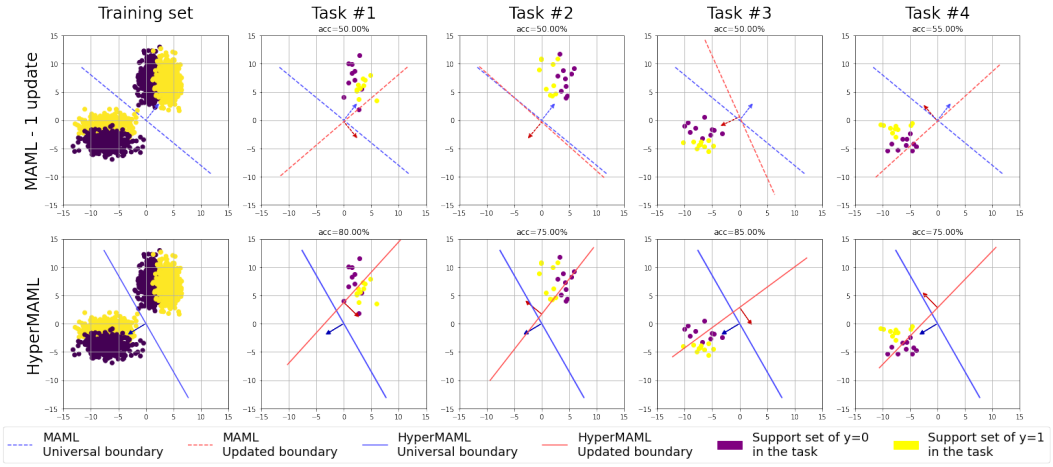

Figure 1: Let us consider a two-dimensional dataset consisting of four Gaussian data clusters (the first column). In the Meta-Learning scenario, we produce a task that consists of samples from two horizontal or vertical ellipses with permuted labels (columns 2-5). The MAML model with one gradient step in the inner loop cannot update all four tasks (see the first row). In practice, in the case of high-dimensional data, using many gradient updates in MAML introduces significant computational overhead (see Section 4.4). Our HyperMAML (pictured in the second row) can solve the task by using the Hypernetwork paradigm. We use only a single update in our method, which can dramatically change weights. We include the details of this experiment in section A of the Appendix.

few gradient steps. Moreover, hypernetworks have previously been used as models for biological information processing, which suggests that such models are more biologically plausible than the gradient-based techniques such as MAML (Segovia-Juarez & Conrad, 1999). In practice, MAML works when there exist universal weights that are close enough to the optimal solution for each task. To visualize such a situation, we present a simple 2D example, where a single gradient update fails to sufficiently adapt the model to a given task – see Fig. 3. We cannot effectively switch weight in one gradient step. On the other hand, when we use many gradient steps, we obtain a complex optimization procedure that uses an inner and outer loop. Such a procedure can be seen as second-order optimization, which is complex to train (Finn et al., 2017). Contrary to MAML we do not need an inner loop in the optimization procedure, and consequently, we do not have second-order optimization. We also reduce the number of hyperparameters, which are used in the inner loop of MAML approach, which would need to be tuned in a grid search. As a result, our algorithm obtains better results than the classical MAML algorithm and produces results comparable to other state-of-the-art algorithms.

The contributions of our work can be summarized as follows:

- We introduce HyperMAML, a novel approach to the Few-Shot learning problem by aggregating information from the support set and directly producing weights updates.
- In HyperMAML, we do not use loss calculation or gradient backpropagation for the update to the new task, thus making the model more biologically feasible and computationally efficient.
- We significantly increase the update ability compared to the classical MAML algorithm, as evidenced by the increased accuracy in numerous benchmarks we perform.

2 RELATED WORK

The problem of Meta-Learning and Few-Shot learning (Hospedales et al., 2020; Schmidhuber, 1992; Bengio et al., 1992) has received a growing amount of attention from the scientific community over the recent years, with the abundance of methods emerging as a result. The idea of HyperMAML is influenced by two kinds of such methods:

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., 2021; Zhmoginov et al., 2022). Other approaches combine weight generators with gradient-based optimizers by choosing target weights from a set of templates (Zhao et al., 2020) or optimizing low-dimensional embeddings which condition the target weight generator (Rusu et al., 2019). 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.

Similarly, HyperMAML utilizes a Hypernetwork to generate weights updates for performing specific tasks. The key difference is that in HyperMAML, the Hypernetwork is not the sole source of model weights. Instead, following (Finn et al., 2017), HyperMAML maintains a set of universal weights and uses the hypernetwork to generate the updates to those weights for novel tasks.

Optimization-based methods, such as MetaOptNet (Lee et al., 2019) are 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), which inspired a multitude of research and numerous extensions to the original algorithm. This includes various techniques for stabilizing its training and improving performance, such as Multi-Step Loss Optimization, and scheduling the learning rate of the meta-optimizer (Antoniou et al., 2018) , using the Bayesian variant of MAML (Yoon et al., 2018), or making MAML permutation-invariant Ye & Chao (2021).

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 not only the base initialization of the model but also the optimizer itself – namely, training a neural network that transforms gradients calculated w.r.t. loss of the support set predictions into weight updates (Munkhdalai & Yu, 2017; Munkhdalai et al., 2018; Li et al., 2017; Rajasegaran et al., 2020).

HyperMAML shares a key characteristic with the optimization-based methods – namely, it also utilizes a base set of weights, which are updated to obtain a model fit for a given task. The key difference between HyperMAML and MAML is that while MAML adapts to novel tasks through multiple steps of gradient-based optimization, HyperMAML generates the updates in a single step using a Hypernetwork. This makes HyperMAML more similar to methods like (Li et al., 2017; Munkhdalai & Yu, 2017), which generate weight updates through trained meta-optimizers. However, contrary to those approaches, in HyperMAML the Hypernetwork predicts the weight updates based on (i) latent representation of the support set, (ii) predictions of the base model for the support set, (iii) ground-truth labels of the support examples (see Fig. 2). Thus HyperMAML does not require calculating either the loss function or its gradients during generating of the task-specific weight updates, making it more computationally efficient.

3 HYPERMAML: HYPERNETWORK FOR FEW-SHOT LEARNING

In this section, we present our HyperMAML model for Few-Shot learning. First, we start by presenting background and notations for Few-Shot learning. Then we describe how the MAML algorithm works. Finally, we present HyperMAML, which can be understood as an extension of the classical MAML.

Algorithm 1 MAML - Model-Agnostic Meta-Learning (Finn et al., 2017)

Require: $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^{N}$: set of training tasks
Require: α, β: step size hyper parameters

```
randomly initialize θ
while not done do
    Sample batch of tasks B from D
    for each task T_i = {S_i, Q_i} from batch B do
        Evaluate ∇_θ L_{S_i}(f_θ) with respect to examples from S_i given loss by equation 2
        Compute adapted parameters θ'_i with gradient descent using formula given by eq. equation 1
    end for
    Update the global parameters of the model θ with formula given by eq. equation 4.
```

3.1 BACKGROUND

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) 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 function f_θ with parameters θ. In the Few-Shot problems f_θ models discriminative probabilities for the classes, $f_\theta(\mathbf{x}) = p(\mathbf{y}|\mathbf{x}, \theta)$. The standard MAML model is trained with the procedure given by Algorithm 1. In each of the training iterations the batch of tasks $\mathcal{B}$ is sampled from $\mathcal{D}$. Further, for each task $\mathcal{T}_i = \{\mathcal{S}_i, \mathcal{Q}_i\}$ from $\mathcal{B}$, MAML adapts the model's parameters θ'_i that are specific for a given task. The actual values of θ'_i are calculated using one or more gradient descent updates. In the simplest case of one gradient iteration, the parameters are updated as follows:

$$\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{S}_i}(f_\theta), \tag{1}$$

where α is the step size that may be fixed as a hyperparameter or meta-learned, 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:

$$\mathcal{L}_\mathcal{Z}(f_\theta) = \sum_{(\mathbf{x}_{i,l}, \mathbf{y}_{i,l}) \in \mathcal{Z}} \sum_{k=1}^{K} -y_{i,l}^{k} \log f_{\theta,k}(\mathbf{x}_{i,j}), \tag{2}$$

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. After calculating the tasks-specific updates θ'_i the general model parameters are trained by optimizing for the performance

of $f_{\theta_i'}$ with respect to θ across tasks from batch $\mathcal{B}$. More concretely, the meta-objective used to train the general parameters of the models is as follows:

$$\mathcal{L}_{MAML}(f_\theta) = \sum_{\mathcal{T}_i \in \mathcal{B}} \mathcal{L}_{\mathcal{Q}_i}(f_{\theta'}) = \sum_{\mathcal{T}_i \in \mathcal{B}} \mathcal{L}_{\mathcal{Q}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{S}_i}(f_\theta)}), \tag{3}$$

Note that the meta-optimization is performed over the model parameters θ, whereas the objective is computed using the updated model parameters θ'. In effect, our proposed method aims to optimize the model parameters such that one or a small number of gradient steps on a new task will produce maximally effective behavior on that task.

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

$$\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{MAML}(f_\theta), \tag{4}$$

where β is the meta step size.

During the inference stage, in order to perform predictions for newly observed task $\mathcal{T}_* = \{\mathcal{S}_*, \mathcal{X}_*\}$ the loss function $\mathcal{L}_{\mathcal{S}_*}(f_\theta)$ is calculated first using eq. equation 2. Next, the parameters θ'_* for task $\mathcal{T}_*$ are calculated from eq. equation 1. The final predictions for query examples $\mathcal{X}_*$ are performed by the model $f_{\theta'_*}$, where for selected query example $\mathbf{x}_q \in \mathcal{X}_*$ we have $p(\mathbf{y}|\mathbf{x}_q, \theta'_*) = f_{\theta'_*}(\mathbf{x}_q, \theta'_*)$.

The main limitation of the approach is that it produces the general weights for all possible tasks, and the adjustment is performed via a gradient-based approach performed on the support set. For some non-trivial challenging tasks, the dedicated parameters θ'_* may be located far from the base weights, θ. Consequently, the adaptation procedure would require significantly more gradient steps, the training may be unstable, and the model will tend to overfit to support set. To overcome this limitation, we propose to replace the gradient-based adaption with the Hypernetwork approach, where the update step is returned by an additional deep model that extracts the information from the support set (see toy example in Fig. 3).

3.2 HYPERMAML - OVERVIEW

We introduce our HyperMAML – a model that utilizes Hypernetworks for modeling weights updates in the classical MAML algorithm. The main idea of the proposed updating scenario is to use information extracted from support examples and predictions given by universal weights to find optimal updates dedicated to a given task. Thanks to this approach, we can switch the classifier's parameters between completely different tasks based on the support set and existing prediction of the universal weights.

The architecture of the HyperMAML is provided in Fig. 2. In this section we present the model for one-shot scenario, and further discuss how to extend it to Few-Shot problems. 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 S. Following the idea from MAML we consider the parameterized function f_θ, that models the discriminative distribution for the classes. In addition, in our architecture we distinguish the trainable encoding network $E(\cdot)$, that transforms data to low-dimensional representation. We postulate to calculate $p(\mathbf{y}|\mathbf{x}_q, \theta') = f_{\theta'}(\mathbf{e}_q)$, where $\mathbf{e}_q$ is the query example $\mathbf{x}_q$ transformed using encoder $E(\cdot)$, and θ' represents the updated parameters for a considered task, $\theta' = \theta + \Delta_\theta$. Compared to gradient-based adaptation step described by equation 1 used to calculate Δ_θ we propose to predict the values using hypernetwork, directly from support set.

Each of the inputs from support set $\mathcal{X}_S$ is transformed by Encoder $E(\cdot)$ in order to obtain low-dimensional matrix of embeddings $\mathbf{E}_\mathcal{S} = [\mathbf{e}_{\mathcal{S},1}, \ldots, \mathbf{e}_{\mathcal{S},K}]^{\mathrm{T}}$. Next, the corresponding class labels for support examples, $\mathbf{Y}_\mathcal{S} = [\mathbf{y}_{\mathcal{S},1}, \ldots, \mathbf{y}_{\mathcal{S},K}]^{\mathrm{T}}$ are concatenated to the corresponding embeddings stored in the rows of matrix $\mathbf{E}_\mathcal{S}$. In addition, we also calculate the predicted values for the examples from the support set using the general model $f_\theta(\mathbf{E}_\mathcal{S}) = \hat{\mathbf{Y}}_\mathcal{S}$, and also concatenate them to $\mathbf{E}_\mathcal{S}$. The matrix transformed support inputs $\mathbf{E}_\mathcal{S}$, together with true support labels $\mathbf{Y}_\mathcal{S}$, and corresponding predictions $\hat{\mathbf{Y}}_\mathcal{S}$ returned by general model are delivered as an input to the hypernetwork $H(\cdot)$ that returns the update Δ_θ. The hypernetwork consists of fully-connected layers with ReLU activations – see section E.1 in the Appendix for details. The parameters for final target model are calculated with the following formula:

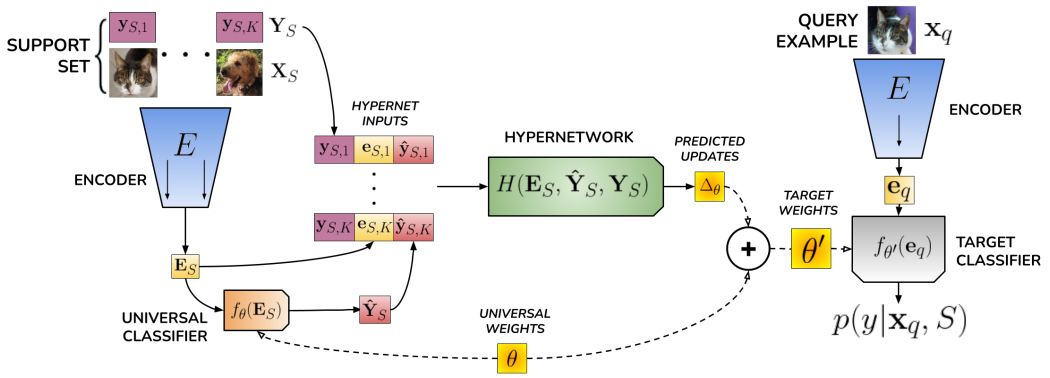

Figure 2: The overview of HyperMAML architecture. The input support examples are processed by convolutional encoding network $E(\cdot)$ and delivered to the fully-connected hypernetwork $H(\cdot)$ together with the true support labels and predictions from general model $f_\theta(\cdot)$. The hypernetwork transforms them, and returns the update of weigths $\Delta\theta$ for target classifier $f_{\theta'}$. The query example is transformed by Encoder $E(\cdot)$, and the final class distribution is returned by the target model $f_{\theta'}$ dedicated to the considered task.

$$\theta' = \theta + \Delta_\theta = \theta + H(\mathbf{E}_\mathcal{S}, \hat{\mathbf{Y}}_\mathcal{S}, \mathbf{Y}_\mathcal{S}). \tag{5}$$

Practically, the Hypernetwork observes the support examples with the corresponding true values and decides how the global parameters θ should be adjusted to the considered task. In addition, the predictions from global model f_θ are also delivered to the model in order to identify the misclassifications and try to correct them during the update state.

3.3 HyperMAML - training

For training the model we assume that encoder $E(\cdot)$ is parametrized by γ, $E := E_\gamma$, and the hypernetwork $H(\cdot)$ by η, $H := H_\eta$. The training procedure is described in Algorithm 2. First, we sample the batch of tasks $\mathcal{B}$ from the given dataset $\mathcal{D}$. Next, for each task $\mathcal{T}_i$ in batch $\mathcal{B}$ we calculate the update $\Delta_{\theta i}$ using the support set $\mathcal{S}_i$, and provide the updated parameters θ' according to the rule given by eq. equation 5. Finally, the objective to train the parameters of the system is calculated using the query sets from the batch tasks $\mathcal{T}_i$:

$$\mathcal{L}_{HyperMAML}(f_\theta) = \sum_{\mathcal{T}_i \in \mathcal{B}} \mathcal{L}_{\mathcal{Q}_i}(f_{\theta'}) = \sum_{\mathcal{T}_i \in \mathcal{B}} \mathcal{L}_{\mathcal{Q}_i}(f_{\theta + \Delta_\theta}), \tag{6}$$

where $\mathcal{L}_{\mathcal{Q}_i}(f_{\theta'})$ is given by eq. equation 2. The parameters of the encoder, hypernetwork, and global parameters θ represent the meta parameters of the system, and they are updated with stochastic gradient descent (SGD) by optimizing $\mathcal{L}_{HyperMAML}(f_\theta)$.

Adaptation to the Few-Shot scenario. The proposed method can be easily extended for Few-Shot scenarios following the aggregation technique from (Sendera et al., 2022). For our approach, we aggregate the embedding values of the support examples from the same class using *mean* operation. In addition, the corresponding predictions within the class are also averaged and concatenated to the averaged per class embedding together with the true class label, and further processed via the hypernetwork.

Warming-up universal weights In practice, it is not trivial to initialize the universal weights of HyperMAML. Classical initialization does not allow to update the universal weights and only the Hypernetwork's parameters are changing. To solve this problem we use a smooth transition from gradient to Hypernetwork update:

Algorithm 2 HyperMAML

Require: $\mathcal{D} = \{\mathcal{T}_n\}_{n=1}^{N}$: set of training tasks
Require: β: step size hyper parameter

randomly initialize θ, γ, η
while not done **do**
  Sample batch of tasks $\mathcal{B}$ from $\mathcal{D}$
  for each task $\mathcal{T}_i = \{\mathcal{S}_i, \mathcal{Q}_i\}$ from batch $\mathcal{B}$ **do**
    Compute adapted parameters θ_i' from $\mathcal{S}_i$ using formula given by eq. equation 5
  end for
  Calculate the loss $\mathcal{L}_{HyperMAML}(f_\theta)$ given by eq. equation 6.
  $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{HyperMAML}(f_\theta)$ ▷ Update the global target parameters θ
  $\eta \leftarrow \eta - \beta \nabla_\eta \mathcal{L}_{HyperMAML}(f_\theta)$ ▷ Update parameters of the hypernetwork H_η
  $\gamma \leftarrow \gamma - \beta \nabla_\gamma \mathcal{L}_{HyperMAML}(f_\theta)$ ▷ Update the parameters of the Encoder E_γ

$$\theta' = \theta + \lambda \cdot H(\mathbf{E}_\mathcal{S}, \hat{\mathbf{Y}}_\mathcal{S}, \mathbf{Y}_\mathcal{S}) - (1 - \lambda) \cdot \alpha \nabla_\theta \mathcal{L}_{\mathcal{S}_i}(f_\theta) \tag{7}$$

where λ is changing from zero to one in a few initial training epochs.

4 Experiments

In the typical Few-Shot learning setting, making a valuable and fair comparison between proposed models is often complicated because of the existence of the significant differences in architectures and implementations of known methods. In order 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 of 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 (Chen et al., 2019). The models are trained from scratch, except for the models trained on **mini-ImageNet** where they are initialized with a pretrained backbone, following (Qiao et al., 2017; Rusu et al., 2019; Ye et al., 2021). In all experiments, the query set of each task consists of 16 samples for each class (80 in total). We split the datasets 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 provide the additional training details in Section E of the Appendix.

4.1 Classification

First, we consider the classical Few-Shot learning scenario. We benchmark the performance of the HyperMAML and other methods on two challenging and widely considered datasets: Caltech-USCD Birds (**CUB**) (Wah et al., 2011) and **mini-ImageNet** (Ravi & Larochelle, 2017). In case of the **mini-ImageNet** dataset we initialize the backbone with pretrained weights, following (Qiao et al., 2017; Ye et al., 2021). We compare HyperMAML to a number of MAML-related and Hypernetwork-based methods, as well as the current state-of-the-art algorithms in the tasks of 1-shot and 5-shot classification, and report the results in Table 1. We report a comparison to a wider pool of few-shot learning methods, as well as the results of models utilizing a larger backbone in the Tables 6 and 7 in the Appendix.

In the 1-shot scenario, HyperMAML yields top performing results (66.11%) on the **CUB** dataset, inferior only to FEAT (Ye et al., 2021) (68.87%). On the **mini-ImageNet** dataset, HyperMAML is among the five best methods, achieving the accuracy of 53.41%. In the 5-shot setting, HyperMAML is among the top-3 best performing models achieving both on the **CUB** and **mini-ImageNet** datasets,

[1] An anonymized version of our code is available at `https://anonymous.4open.science/r/few-shot-hypernets-public-DB4F`. We shall release the code with our experiments after the end of the review period.

achieving 78.89% and 68.76% accuracy, with FEAT (Ye et al., 2021) and HyperShot (Sendera et al., 2022) outperforming it by a small margin.

The obtained results show that HyperMAML achieves performance better or comparable to a variety of Few-Shot learning methods, in particular MAML (Finn et al., 2017), as well as techniques which derive from it (Antoniou et al., 2018; Rajeswaran et al., 2019; Fan et al., 2021; Yoon et al., 2018; Ye & Chao, 2021).

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

	CUB		**mini-ImageNet**	
Method	1-shot	5-shot	1-shot	5-shot
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)	–	–	53.80 ± 1.46	64.23 ± 0.69
Unicorn-MAML (Ye & Chao, 2021)	–	–	*54.89*	–
HyperShot (Sendera et al., 2022)	65.27 ± 0.24	*79.80 ± 0.16*	52.42 ± 0.46	*68.78 ± 0.29*
MLH (Zhao et al., 2020)	–	–	49.41 ± 0.96	67.16 ± 0.42
PPA (Qiao et al., 2017)	–	–	54.53 ± 0.40	–
FEAT (Ye et al., 2021)	**68.87 ± 0.22**	**82.90 ± 0.15**	**55.15 ± 0.20**	**71.61 ± 0.16**
HyperTransformer (Zhmoginov et al., 2022)	–	–	54.10	68.50
HyperMAML	*66.11 ± 0.28*	78.89 ± 0.19	53.41 ± 0.21	*68.76 ± 0.17*

4.2 CROSS-DOMAIN ADAPTATION

In the cross-domain adaptation setting, the model is evaluated on tasks coming from a different distribution than the one it had been trained on. Therefore, such a task is more challenging than standard classification and is a plausible indicator of a model's ability to generalize. In order to benchmark the performance of HyperMAML in cross-domain adaptation, we combine two datasets so that the training fold is drawn from the first dataset and validation and the testing fold – from another one. We report the results in Table 2. In the task of 1-shot **Omniglot→EMNIST** classification, HyperMAML achieves the second-best result (79.84%), with HyperShot+finetuning (80.65%) being the top one. Compared to the other methods, we observe relatively smaller performance growth as more data becomes available. In the 5-shot **Omniglot→EMNIST** classification task HyperMAML yields comparable results (89.22%) to HyperShot (Sendera et al., 2022) (90.81%) and DKT (Patacchiola et al., 2020) (90.30%), which are the state-of-the-art in this setting. In the most challenging task of **mini-ImageNet→CUB** classification, our method performs comparably to baseline methods such as MAML, ProtoNet and Matching Net (Finn et al., 2017; Vinyals et al., 2016; Snell et al., 2017), particularly in the 1-shot setting.

Table 2: The classification accuracy results for the inference tasks on cross-domain tasks (**Omniglot→EMNIST** and **mini-ImageNet→CUB**) datasets 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
MatchingNet (Vinyals et al., 2016)	75.01 ± 2.09	87.41 ± 1.79	36.98 ± 0.06	50.72 ± 0.36
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	*90.30 ± 0.49*	**40.14 ± 0.18**	56.40 ± 1.34
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
HyperShot (Sendera et al., 2022)	78.06 ± 0.24	89.04 ± 0.18	39.09 ± 0.28	*57.77 ± 0.33*
HyperShot + finetuning	**80.65 ± 0.30**	**90.81 ± 0.16**	*40.03 ± 0.41*	**58.86 ± 0.38**
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	*79.07 ± 1.09*	89.22 ± 0.78	36.32 ± 0.61	49.43 ± 0.14

4.3 PARAMETER UPDATE MAGNITUDE

Next, we consider the ability of HyperMAML to produce the correct weight updates. One of the drawbacks of MAML is that, in practice, gradient updates change weights very slowly, especially when meta tasks require completely different weights (see Fig. 3). On the other hand, Hypernetworks can produce significant updates. To verify such behaviour on a real, non-trivial dataset, we calculate the norm of classifier weight updates in MAML and HyperMAML trained for **Omniglot→EMNIST** classification and report the results in Table 3.

As we can see, HyperMAML produces larger updates, which, combined with higher accuracy than MAML (see Table 2), suggests faster and more accurate convergence.

Table 3: Magnitude of the update of weight and bias by MAML (with one update) and HyperMAML.

	MAML	**HyperMAML**
weight	2.53 ± 0.29	113.28 ± 0.63
bias	1.47 ± 0.14	27.72 ± 0.68

In the case of classical MAML, there exist few modifications in updating procedures (inner loop) like MAML++ (Antoniou et al., 2018) or Meta-SGD (Li et al., 2017). However, such modifications are required in classical MAML since few gradient updates do not always guarantee convergence. On the contrary, HyperMAML provides a possible solution to the problem by a novel update that is not generated directly by gradient descent, but rather by a forward pass of a Hypernetwork.

4.4 COMPUTATIONAL EFFICIENCY

Finally, we verify the hypothesis that HyperMAML offers an increased computational efficiency compared to MAML. To this end, we measure the times of processing the entire **Omniglot → EMNIST** test dataset (600 tasks in total) by MAML with different numbers of gradient steps and HyperMAML and report the results in Table 4, and for other datasets in Table 8 in the Appendix.

We find that processing the test data with HyperMAML takes approximately the same time as using MAML with just 2 gradient updates. We also note that even given the budget of 100 gradient updates, MAML never matches the accuracy achieved by a single update generated by HyperMAML.

Table 4: Time (in seconds) spent on processing the entire **Omniglot → EMNIST** test dataset (600 tasks) by MAML with different numbers of gradient steps and HyperMAML.

Model	**Steps**	**Time**	**Accuracy**
MAML	0	2.17 ± 0.28	19.978 ± 0.348
	1	5.16 ± 0.13	74.67 ± 0.25
	2	7.30 ± 0.11	74.69 ± 0.24
	3	8.87 ± 0.08	74.71 ± 0.25
	4	10.79 ± 0.19	74.71 ± 0.24
	5	12.66 ± 0.14	74.71 ± 0.23
	10	22.60 ± 0.28	74.74 ± 0.25
	25	51.75 ± 0.24	74.78 ± 0.25
	50	103.88 ± 2.24	74.82 ± 0.25
	100	212.62 ± 2.24	74.86 ± 0.27
HyperMAML	–	8.21 ± 0.41	80.78 ± 0.59

5 CONCLUSIONS

In this work, we introduced HyperMAML – a novel Meta-Learning algorithm strongly motivated by MAML (Finn et al., 2017). The crucial difference between the two methods lies in the fact that in HyperMAML the update of the model is given not by the gradient optimization, but by the trainable Hypernetwork. Consequently, HyperMAML is more computationally efficient than MAML, as during adaptation it performs only a single parameter update. Moreover, HyperMAML is not only more biologically feasible as it does not use backpropagation, but it can adapt easier to different settings and has a smaller number of hyperparameters. Our experiments show that HyperMAML outperforms the classical MAML in a number of standard Few-Shot learning benchmarks and achieves results better or comparable to various other state-of-the-art methods in most cases.

Our results indicate that Few-Shot weight adaptation can be performed directly, without calculating the actual loss or gradients – a more biologically plausible learning method (Lillicrap et al., 2020; Song et al., 2020; Whittington & Bogacz, 2019). Moreover, HyperMAML adapts to new tasks quicker than MAML, which needs to be tuned with many gradient updates. Thus, HyperMAML is a step toward more efficient and environment-friendly Meta-Learning techniques.

REFERENCES

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.

Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2018.

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.

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.

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. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 9537–9548, 2018.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375, 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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

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

Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification, 2019.

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.

Dahyun Kang, Heeseung Kwon, Juhong Min, and Minsu Cho. Relational embedding for few-shot classification, 2021.

Minyoung Kim and Timothy Hospedales. Gaussian process meta few-shot classifier learning via linear discriminant laplace approximation. *arXiv preprint arXiv:2111.05392*, 2021.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 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.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification, 2020.

Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning, 2019.

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.

Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *International Conference on Machine Learning*, pp. 3664–3673. PMLR, 2018.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

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.

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.

Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training, 2020.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization, 2019.

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. doi: 10.1162/neco.1992.4.1.131. URL https://doi.org/10.1162/neco.1992.4.1.131.

J.L. Segovia-Juarez and M. Conrad. Hypernetwork model of biological information processing. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 1, pp. 511–515 Vol. 1, 1999. doi: 10.1109/CEC.1999.781976.

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.

Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *Advances in neural information processing systems*, 33:22566–22579, 2020.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning, 2019.

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.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need?, 2020.

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.

Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

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.

James CR Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250, 2019.

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. Few-shot learning via embedding adaptation with set-to-set functions, 2021.

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.

Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Differentiable earth mover's distance for few-shot learning, 2020.

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 2D EXAMPLE

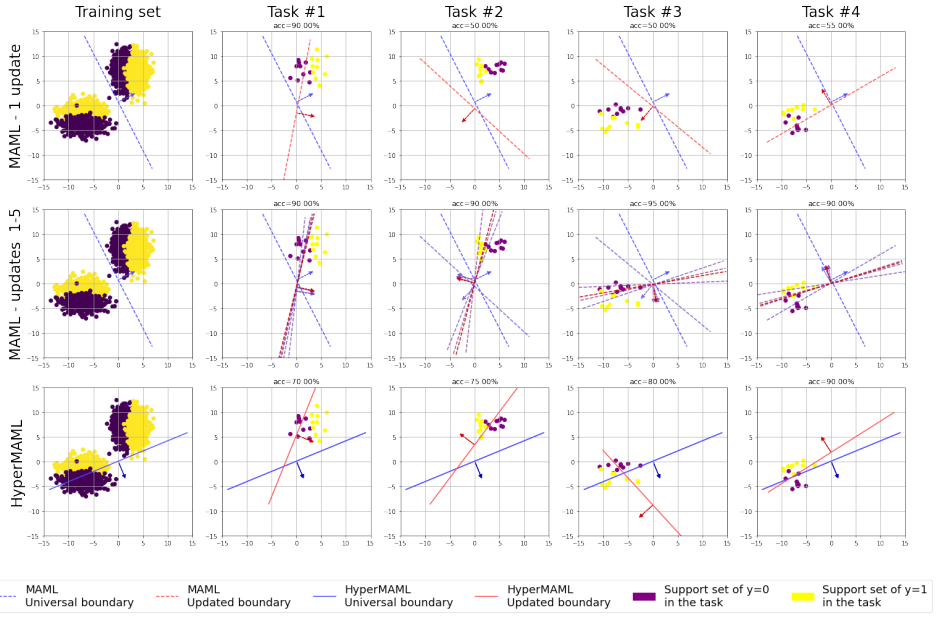

Figure 3: Let us consider a data set consisting of four Gaussian data (the first column). In the Meta-Learning scenario, we produce a task that consists of samples from two horizontal or vertical ellipses with permuted labels (second to the fifth column). The MAML model with one gradient step in the inner loop (see the first row) cannot update all four tasks. On the other hand, when we use five gradient updates, the model gives a reasonable solution (see the second row). In practice, for MAML, we cannot effectively use many steps in inner lop in the case of really high dimensional data. Our HyperMAML can solve the task by using the hypernetwork paradigm. We use only one update in our method, which can dramatically change weights. Details of the experiment we add to supplementary material.

To visualize the fact that the MAML algorithm is significantly limited by the weight update procedure based on the gradient method, we will analyze a simple 2D example. Let us consider a data set consisting of 4 Gaussian data; see the first column in Fig. 3. In the meta-learning scenario, we produce a task that consists of samples from two horizontal or vertical ellipses with permuted labels. In practice, we would like to force a method to adapt to four tasks presented in the second to the fifth column in Fig. 3. In the training procedure, we use a neural network with one layer – in practice, a linear model. We can visualize weights and decision boundaries thanks to using such simple architecture. We marked the universal weights with the blue color and the updated parameters – with red.

The MAML model with one gradient step in the inner loop (see the first row in Fig. 3) cannot update all four tasks. On the other hand, when we use five gradient updates, the model gives a reasonable solution (see the second row in Fig. 3). In practice, for MAML, we cannot effectively use many steps in inner lop in the case of really high dimensional data.

Our HyperMAML can solve the task by using the hypernetwork paradigm. We use only one update in our method, which can dramatically change weights. Notice that even for five steps, MAML cannot significantly update weight. In practice, it uses only rotations.

B Ablation study of mechanisms used in HyperMAML

In this section, we present two mechanisms we utilize when training HyperMAML.

Switching mechanism is a smooth transition between training the convolutional encoder through MAML and HyperMAML objective. We consider the MAML warm-up as a starting point of the training loop and then smoothly move towards HyperMAML training. During training, we define two "milestone" epochs (see Section E.3, between which the transition occurs. During the transition, we continuously decrease the participation of the MAML objective in the training process. It is done by multiplying MAML loss by p, ranging from 1.0 to 0.0 for a given number of epochs, and multiplying the HyperMAML loss by $1 - p$. Our motivation for this mechanism is to train a better universal optimizing the MAML objective during the warm-up part of the training and then Switch to the HyperMAML objective gradually.

Enhancement of the embeddings is another mechanism in our framework. When preparing the input to the Hypernetwork from the support set, we first obtain support embeddings from the encoder. Then, we forward those embeddings through the universal classifier and obtain its predictions for support examples. We concatenate those predictions to the support embeddings, as well as their respective ground-truth labels. The whole process is visualized in Figure 4. Our motivation to perform such an operation is to give the Hypernetwork the information about the current decision of the classifier, to better estimate the updates to its weights. We note that even though the Hypernetwork generates the weights of the classifier based on the enhanced support embeddings, the generated downstream classifier does not use enhancements when processing the query set. Instead, it only processes the raw embeddings obtained from the encoder (see Figure 2 from the main paper).

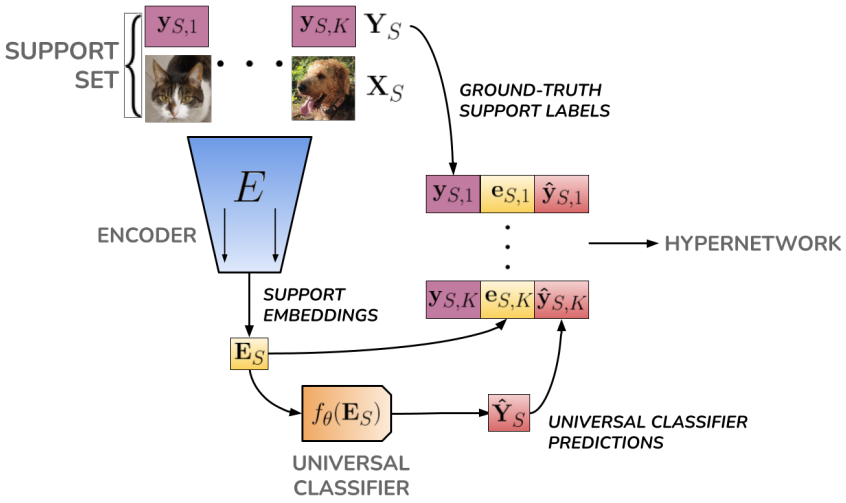

Figure 4: Illustration of the embedding enhancement mechanism. The support embeddings (which serve as the input to the HyperMAML Hypernetwork) are enhanced with the predictions of the base classifier and their respective ground-truth labels.

We perform an ablation study of both mechanisms on the task 5-shot 1-way **Omniglot→EMNIST** classification. The results, reported in Table 5, indicate that both mechanisms utilized individually improve the performance of HyperMAML, and the combination of the two yields the best results.

Table 5: The classification accuracy of HyperMAML on **Omniglot→EMNIST** task in the 5-way 1-shot scenario. We consider different strategies by adding switch or embedding enhancement strategy to our model.

Model	Switch	Enhancement	Accuracy	Learning rate	Milestones
HyperMAML	no	no	74.39 ± 0.88	0.0001	-
	no	yes	75.28 ± 1.29	0.0001	-
	yes	no	76.89 ± 0.71	0.01	$51,550$
	yes	yes	79.10 ± 1.43	0.01	$51,550$

C FULL CLASSIFICATION RESULTS

We provide an expanded version of Table 1 from the main paper, with numerous additional baseline methods – see Table 6. We also report the performance of HyperMAML, as well as several baseline methods trained with ResNet-12 He et al. (2015) as backbone on the **mini-ImageNet** dataset in Table 7.

Table 6: The classification accuracy results for the inference tasks on **CUB** and **mini-ImageNet** datasets in the 5-shot setting. The highest results are bold and 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
SNAIL (Mishra et al., 2018)	–	–	45.10	55.20
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
SimpleShot (Wang et al., 2019)	–	–	49.69 ± 0.19	66.92 ± 0.17
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
MatchingNet (Vinyals et al., 2016)	60.19 ± 1.02	75.11 ± 0.35	48.25 ± 0.65	62.71 ± 0.44
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 + CosSim (Patacchiola et al., 2020)	63.37 ± 0.19	77.73 ± 0.26	48.64 ± 0.45	62.85 ± 0.37
DKT + BNCosSim (Patacchiola et al., 2020)	62.96 ± 0.62	77.76 ± 0.62	49.73 ± 0.07	64.00 ± 0.09
GPLDLA (Kim & Hospedales, 2021)	63.40 ± 0.14	78.86 ± 0.35	52.58 ± 0.19	–
VAMPIRE (Nguyen et al., 2020)	–	–	51.54 ± 0.74	64.31 ± 0.74
PLATIPUS (Finn et al., 2018)	–	–	50.13 ± 1.86	–
ABML (Ravi & Beatson, 2018)	49.57 ± 0.42	68.94 ± 0.16	45.00 ± 0.60	–
OVE PG GP + Cosine (ML) (Snell & Zemel, 2020)	63.98 ± 0.43	77.44 ± 0.18	50.02 ± 0.35	64.58 ± 0.31
OVE PG GP + Cosine (PL) (Snell & Zemel, 2020)	60.11 ± 0.26	79.07 ± 0.05	48.00 ± 0.24	67.14 ± 0.23
Reptile (Nichol et al., 2018)	–	–	49.97 ± 0.32	65.99 ± 0.58
R2-D2 (Bertinetto et al., 2018)	–	–	48.70 ± 0.60	65.50 ± 0.60
VSM (Zhen et al., 2020)	–	–	54.73 ± 1.60	68.01 ± 0.90
PPA (Qiao et al., 2017)	–	–	54.53 ± 0.40	–
DFSVLwF (Gidaris & Komodakis, 2018)	–	–	$\mathbf{56.20 \pm 0.86}$	–
FEAT (Ye et al., 2021)	$\mathbf{68.87 \pm 0.22}$	$\mathbf{82.90 \pm 0.15}$	$\textit{55.15} \pm \textit{0.20}$	$\mathbf{71.61 \pm 0.16}$
HyperShot (Sendera et al., 2022)	65.27 ± 0.24	79.80 ± 0.16	52.42 ± 0.46	68.78 ± 0.29
HyperShot+ finetuning (Sendera et al., 2022)	$\textit{66.13} \pm \textit{0.26}$	80.07 ± 0.22	53.18 ± 0.45	69.62 ± 0.28
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)	–	–	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
MetaNet (Munkhdalai & Yu, 2017)	–	–	49.21 ± 0.96	–
PAMELA (Rajasegaran et al., 2020)	–	–	53.50 ± 0.89	$\textit{70.51} \pm \textit{0.67}$
HyperMAML	$\textit{66.11} \pm \textit{0.28}$	78.89 ± 0.19	53.41 ± 0.21	68.76 ± 0.17

Table 7: The classification accuracy results for the inference tasks in the **mini-ImageNet** dataset in the 5-way (1-shot and 5-shot) scenarios. We consider models using the ResNet-12 backbone. The highest results are bold and second-highest in italic (the larger, the better).

Method	1-shot	5-shot
cosine classifier (Chen et al., 2019)	55.43 ± 0.81	77.18 ± 0.61
TADAM (Oreshkin et al., 2018)	58.50 ± 0.30	76.70 ± 0.30
Shot-Free (Ravichandran et al., 2020)	59.04	77.64
TPN (Liu et al., 2019)	59.46	75.65
HyperShot (Sendera et al., 2022)	59.12 ± 0.26	76.53 ± 0.22
HyperShot + adaptation (Sendera et al., 2022)	60.30 ± 0.31	76.21 ± 0.20
MTL (Sun et al., 2019)	61.20 ± 1.80	75.50 ± 0.80
RFS-simple (Tian et al., 2020)	62.02 ± 0.63	79.64 ± 0.44
ProtoNet (Snell et al., 2017)	62.39 ± 0.21	80.53 ± 0.14
MetaOptNet (Lee et al., 2019)	62.64 ± 0.82	78.63 ± 0.46
MatchingNet (Vinyals et al., 2016)	63.08 ± 0.80	75.99 ± 0.60
MAML (Finn et al., 2017)	63.11 ± 0.92	80.81 ± 0.14
PPA (Qiao et al., 2017)	59.60	73.34
CAN (Hou et al., 2019)	63.85 ± 0.48	79.44 ± 0.34
NegMargin (Liu et al., 2020)	63.85 ± 0.81	81.57 ± 0.56
DeepEMD (Zhang et al., 2020)	65.91 ± 0.82	*82.41 ± 0.56*
Unicorn-MAML (Ye et al., 2021)	65.17 ± 0.20	84.30 ± 0.14
FEAT (Ye et al., 2021)	*66.78 ± 0.20*	82.05 ± 0.14
RENet (Kang et al., 2021)	**67.60 ± 0.44**	**82.58 ± 0.30**
LEO (Rusu et al., 2019)	61.76 ± 0.08	81.44 ± 0.09
HyperMAML	60.93 ± 0.22	75.74 ± 0.17

D COMPUTATIONAL EFFICIENCY – RESULTS ON NATURAL IMAGE DATASETS

In this section, we perform similar experiments to those described in Section 4.3 of the main paper and measure the inference time of HyperMAML and MAML with different numbers of gradient steps on **CUB** and **mini-ImageNet** datasets. We summarize the results in Table 4. Similarly to the benchmark performed on smaller images from the **Omniglot** $\rightarrow$ **EMNIST** dataset, the inference time of HyperMAML is comparable to MAML with two gradient steps. Likewise, MAML never achieves accuracy higher than HyperMAML.

As opposed to Section 4.3 of the main paper, we report the accuracies of MAML only up to seven gradient steps. This is due to an insufficient amount of GPU memory available for making more steps in the MAML implementation we used (Chen et al., 2019). We also note that the accuracies of MAML reported here are significantly higher than the ones reported in Table 1 of the main paper. The MAML accuracies from that table were previously reported in the literature (Patacchiola et al., 2020), whereas the results in Table 4 have been obtained by the MAML implementation in our codebase (Chen et al., 2019).

E TRAINING DETAILS

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

E.1 ARCHITECTURE OVERVIEW

The architecture of HyperMAML consists of the following parts (as outlined in the Figure 2 in the main body of the work):

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

Table 8: Time (in seconds) spent on processing the entire **CUB** and **mini-ImageNet** test datasets (600 tasks each) by MAML with different numbers of gradient steps and HyperMAML.

		CUB		**mini-ImageNet**	
Model	**Steps**	**Time**	**Accuracy**	**Time**	**Accuracy**
MAML	0	4.15 ± 0.03	20.01 ± 0.02	4.24 ± 0.14	20.10 ± 0.31
	1	6.89 ± 0.10	47.99 ± 0.22	8.48 ± 0.14	43.46 ± 0.42
	2	9.44 ± 0.15	58.53 ± 0.55	10.50 ± 0.80	48.71 ± 0.19
	3	11.60 ± 0.02	61.54 ± 0.41	13.17 ± 0.26	48.32 ± 0.38
	4	12.97 ± 0.09	62.18 ± 0.38	15.17 ± 0.18	48.82 ± 0.25
	5	15.00 ± 0.17	62.66 ± 0.40	17.29 ± 0.32	48.89 ± 0.31
	6	16.90 ± 0.12	62.77 ± 0.36	19.01 ± 0.26	48.80 ± 0.24
	7	19.07 ± 0.17	62.83 ± 0.34	20.55 ± 0.30	48.87 ± 0.29
HyperMAML	–	8.46 ± 0.37	66.11 ± 0.28	11.11 ± 0.25	53.41 ± 0.21

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.

For the **mini-ImageNet** dataset we additionally test the performance of HyperMAML with a larger backbone – namely ResNet-12 He et al. (2015).

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 two or 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 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.

E.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 (Kingma & Ba, 2014) and the MultiStepLR learning rate scheduler with the decay of 0.3 and learning rate starting from 0.01 or 0.001. We train HyperMAML for 4000 epochs on all the datasets, save for the simpler **Omniglot** $\rightarrow$ **EMNIST** classification task, where we train for 2048 epochs instead.

For the **mini-ImageNet** experiments we follow a strategy of using a pre-trained backbone, suggested by (Qiao et al., 2017; Rusu et al., 2019; Ye et al., 2021; Ye & Chao, 2021). More specifically, at the beginning of the training we initialize the Encoder of HyperMAML with weights of backbone pretrained for classification of **all** 64 classes from the **mini-ImageNet** training set. In practice, for consistency, we use the identical set of pretrained weights as Ye et al. (2021).

E.3 Hyperparameters

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

Table 9: Hyperparameters for each of conducted 1-shot experiments.

hyperparameter	**CUB**	**mini-ImageNet**	**mini-ImageNet → CUB**	**Omniglot → 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
warmup milestones	$51, 550$	$101, 1100$	$101, 1100$	$51, 550$

Table 10: Hyperparameters for each of the conducted 5-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	$101, 1100$	$101, 1000$	$101, 1000$	$51, 550$

F IMPLEMENTATION DETAILS

We implement HyperMAML using the PyTorch framework (Paszke et al., 2019). We shall release the code publicly after the end of the review period. Each experiment described in this work was run on a single NVIDIA RTX 2080 GPU.