GEMS: Generating Efficient Meta-Subnets

Varad Pimpalkhute
TCS Research,
Mumbai, India
varad.p@tcs.com

Shruti Kunde
TCS Research,
Mumbai, India
shruti.kunde@tcs.com

Rekha Singhal
TCS Research,
Mumbai, India
rekha.singhal@tcs.com

Abstract

Gradient-based meta learners (GBML) such as MAML [6] aim to learn a model initialization across similar tasks, such that the model generalizes well on unseen tasks sampled from the same distribution with few gradient updates. A limitation of GBML is its inability to adapt to real-world applications where input tasks are sampled from multiple distributions. An existing effort [23] learns $\mathcal{N}$ initializations for tasks sampled from $\mathcal{N}$ distributions; roughly increasing training time by a factor of $\mathcal{N}$. Instead, we use a single model initialization to learn distribution-specific parameters for every input task. This reduces negative knowledge transfer across distributions and overall computational cost. Specifically, we explore two ways of efficiently learning on multi-distribution tasks: 1) Binary Mask Perceptron (BMP) which learns distribution-specific layers, 2) Multi-modal Supermask (MMSUP) which learns distribution-specific parameters. We evaluate the performance of the proposed framework (GEMS) on few-shot vision classification tasks. The experimental results demonstrate an improvement in accuracy and a speed-up of $\sim 2\times$ to $4\times$ in the training time, over existing state of the art algorithms on quasi-benchmark datasets in the field of meta-learning.

1. Introduction

Human beings quickly learn to identify objects in their surroundings by observing just a few samples and utilizing previous knowledge. Meta-learning, or learning-to-learn, aims to emulate the human brain by training a model on few tasks from a distribution (also known as few-shot learning [21, 20]) and generalizing on unseen tasks from the same distribution. We study gradient-based meta-learning (GBML) algorithms [3, 6, 9, 13, 15, 17, 23, 14, 10] such as MAML [6] that aim to learn an optimal model-prior, such that the model converges rapidly with few gradient updates when exposed to unseen tasks sampled from the same distribution.

The basic premise behind GBML is to learn the underlying structure of the input task. Model will generalize well if the structure of the unseen tasks is similar to that of the training tasks. Most state-of-the-art GBML algorithms in the literature assume that tasks are sampled from either the same or similar distributions. As similarity among the distributions decreases, there is an increase in negative knowledge transfer, resulting in deterioration of model accuracy. It becomes imperative to use multiple model initializations for tasks sampled from different distributions. For example, a human-being may apply the knowledge gained to drive a four-wheeler (car) of a particular model, to different types of four-wheeled vehicles. The same knowledge may not be beneficial for flying an aircraft or riding a bike. However, training on multiple model initializations (e.g., [23], Multi-MAML [1]) results in a linear increase in training time, albeit resulting in a better generalization as compared to training on a single model initialization. Consequently, we observe a trade-off between the computation cost and model performance.

In this paper, we address this trade-off, by coming up with an efficient strategy for multi-distributional training of GBML algorithms. In previous work [15], authors have demonstrated that a model is capable of generalizing well on unseen tasks from similar distributions even if all layers in the network, except the head-layer are frozen. The efficacy of this approach decreases in a multi-distribution scenario as the layers are frozen agnostic to the structure of the input task. This led us to propose an approach which determines the specific layers to freeze, based on the structure of the input distribution. Deeper exploration within parameters of the layer, led us to train on *task-specific* parameters and share the knowledge gained across *task-agnostic* parameters, resulting in an improved generalization in a multi-distributional setting. We propose two GEMS (approaches) (1) Binary mask approach which identifies relevant layers in a model. We then go one step further and propose a (2) Multimodal Meta Supermask approach which identifies

[1] Multi-MAML learns a separate model initialization for each input distribution $\{p_i(\mathcal{T}) \rightarrow \theta_i | i = 1, 2, ..., \mathcal{N}\}$.

relevant subnetworks of parameters in a model. Both of our approaches are implemented using a single model initialization on GBML algorithms. As a part of the empirical analysis, both our proposed approaches are tested on unseen tasks from a similar distribution as well as tasks from a distribution unknown to the model(cross domain) during training. Our aim is to maximize the accuracy in a multi-distributional setting, while minimizing the compute/training time.

$$\text{MAML}_{(Acc.)} \leq \text{GEMS}_{(Acc.)} \leq \text{MULTI-MAML}_{(Acc.)}$$
$$\text{MAML}_{(Cmp.)} \leq \text{GEMS}_{(Cmp.)} \leq \text{MULTI-MAML}_{(Cmp.)}$$

Our contributions are as follows:

1. Binary Mask Perceptron (BMP) approach which identifies task-specific network architecture layers for training.
2. Multimodal Meta Supermark (MMSUP) approach identifies task-specifc subnets of neurons in the network architecture for training.
3. Empirical analysis on quasi-benchmark datasets in the meta-learning field.

2. Related Work

The idea of learning-to-learn with few shots has been prevalent for some time now [21, 18, 19]. Most model-agnostic meta-learners aim to learn a model initialization for a certain task distribution leading to fast adaption using gradient descent. The results reported based on these meta-learners are encouraging [6, 9]. MAML is an algorithm that tries to find the optimal model initialization for an input distribution. It is model-agnostic and is widely used across various domains for few-shot learning. Certain variants of MAML [15, 13, 14, 10] focused on improving task-specific learning in the inner loop, whereas, others [3, 4] focused on improving task-agnostic learning in the outer loop. [17, 9, 2] focused on addressing challenges in MAML such as overfitting, unstable training, compute-efficient, etc. However, the performance of such approaches is limited, especially when the taskset is sampled from multi-modal task distributions.

Multi-initialization MAML. Multi-MAML and [23] (MMAML) address the challenge of training multiple distributions on a single initialization. Training multiple distributions on a single initialization leads to deterioration of accuracy, resulting in a need to train multiple initializations. Multi-MAML (training $\mathcal{N}$ distributions on $\mathcal{N}$ corresponding initializations) reduces negative knowledge transfer, based on the assumption that the input task distribution is known. MMAML addresses this issue by introducing a modulation network that automates the process of identifying the mode (initialization) of the input task. However, both Multi-MAML and MMAML are computation expensive as they train on different $\mathcal{N}$ initializations, leading to an increase in the overall training time. Furthermore, no knowledge is shared between tasks from different distributions.

Gradient Sparsity. Instead of fixing the layers or parameters to be frozen during training, [15, 13] meta-learns a binary mask corresponding to the parameters of the backbone. These set of parameters are masked on the weights of the backbone, akin to switching trainable and non-trainable parameters. However, meta-learning additional set of parameters leads to a computational overhead. We instead propose a simple MLP having far less parameters that finds trainable layers instead of trainable parameters. We also highlight that [14] was inclined more towards single distributional training as opposed to our approach that focuses on multi-distributional training.

Network Pruning. [1] focus on pruning weights of the underlying backbone to reduce the computational expense during training and inference in GBML algorithms. The authors make use of Lottery Ticket Hypothesis[2] to determine a sub-network that has a sufficiently good accuracy, thus pruning rest of the weights in the backbone. While this does result in reduced computation, it comes at the cost of deterioration in accuracy. Instead, updating weights of the sub-network while freezing rest of the other weights ensures that only the relevant weights are updated while retaining the knowledge gained from previous tasks. In another effort in network pruning [22] authors present an approach for learning a sparse meta-initialization network from training tasks such that in a new task the learned sub-network can quickly converge to the optimal solution via gradient descent.

Our aim is train a single compute-optimal, model-initialization that can generalize well on tasks from multiple-distribution. We propose the GEMS framework that employs two novel approaches, namely BMP, which generates a binary mask using an adapter and MMSUP, which determines relevant subnetworks of model parameters to be trained on for tasks from different distributions.

3. Methodology

In this section, we describe our proposed approach - **G**enerating **E**fficient **M**eta **S**ubnets (GEMS). A drawback of the gradient-based meta-learning algorithms is that the input task distribution is assumed to be unimodal i.e., the tasks are assumed to be sampled from a single distribution. Naively training tasks sampled from multiple distributions on MAML leads to a deterioration in accuracy, when number of distributions is increased by one [23]. A brute force

[2]Lottery Ticket Hypothesis [7] articulate that in a neural network, there exists a set of subnetworks that when trained on achieve competitive accuracy.

approach (Multi-MAML) is able to train all $\mathcal{N}$ distributions on $\mathbb{N}$ separate MAML architectures to get N different model initializations. MMAML [23] automates the process of identifying the distribution of input task, thus training on $\mathcal{N}$ different model intializations. Both approaches are computationally more expensive than MAML. Can we do better? Rather than training on multiple initializations, does the answer lie in identifying the effectiveness of a given model parameter for training a task? Does transferring positive knowledge between two tasks from different distributions, along with identifying distribution-specific parameters, improve performance in a multi-distribution setup? We propose two algorithms to address this issue: 1) Binary Mask Perceptron (BMP), 2) Multi-modal Meta Supermask (MMSUP). Binary Mask Perceptron meta-learns distribution-specific layers and distribution-specific learning rate by introducing a binary mask adapter. Multi-modal Meta Supermask is inspired from Lottery Ticket Hypothesis [7, 16, 25], thus, identifying distribution-specific subnetworks in the underlying architecture, while sharing knowledge among distribution agnostic parameters. Masking of parameters thus plays an important role in our approach.

In Section 3, we formulate the meta-learning problem statement. In Section 3.1, and Section 3.2, we explain the proposed methods in detail.

Algorithm 1 Binary Mask Perceptron (BMP)

Require: Learning rates η, β, Multi-Task Distributions $p_0(\mathcal{T}), p_1(\mathcal{T}), ..., p_{\mathcal{N}-1}(\mathcal{T})$
Ensure: Randomly initialize θ, ϕ
while not done **do**
  Sample a batch of tasks $\mathcal{T}_i \in \{p_0(\mathcal{T}), ..., p_{\mathcal{N}-1}(\mathcal{T})\}$
  for each $\mathcal{T}_i$ **do**
    Sample datapoints $\{\mathcal{D}_{train} = \{x^{(j)}, y^{(j)}\}\}$ from $\mathcal{T}_i$
    Evaluate $\mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_m})$ at each gradient step m by evaluating $\mathcal{L}_{\mathcal{T}_i}$ w.r.t $\mathcal{D}_{train}$
    Compute Binary Mask BM^m and task-specific LR $\alpha_{\mathcal{T}_i}^m$ using $g_\phi\big(\mathcal{T}_i, \theta_{\mathcal{T}_i}^m, \nabla_{\theta_{\mathcal{T}_i}} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_m})\big)$ Equation 4
    Compute updates on task-specific weights using gradient descent: $\theta_{\mathcal{T}_i}^{m+1,l} = \theta_{\mathcal{T}_i}^{m,l} - \alpha_{\mathcal{T}_i}^m \big(\mathrm{BM}_l^m \circ \nabla_{\theta_{\mathcal{T}_i}} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_m})\big)$
    Sample datapoints $\{\mathcal{D}_{test} = \{x^{(j)}, y^{(j)}\}\}$ from $\mathcal{T}_i$ for meta-update
  end for
  Compute $\mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{test}}(f_{\mathcal{T}_i}^{\theta_m})$ by evaluating loss-criterion w.r.t $\mathcal{D}_{test}$.
  Update weights: $\phi \leftarrow \phi - \beta \nabla_\phi \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{test}}(f_{\mathcal{T}_i}^{\theta_m})$ and $\theta \leftarrow \theta - \eta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{test}}(f_{\mathcal{T}_i}^{\theta_m})$
end while

Preliminaries

In few-shot learning (FSL), we are given $\mathcal{K}$ samples for each of the $\mathcal{N}$ classes, and the goal is to train the model (f_θ) to converge well on the input dataset $\mathcal{D}_{\mathcal{K}}^{\mathcal{N}}$. Meta-learning, especially the gradient-based meta-learning algorithm, Model Agnostic Meta Learning (MAML), is often used on few-shot tasks. MAML identifies a good model initialization during training such that the model f_θ is able to rapidly converge on unseen tasks with a few adaptation steps. Given a model f, randomly initialized with parameters θ_0, we assume that tasks are sampled from a single distribution $p(\mathcal{T})$, such that $\mathcal{T}_i \sim p(\mathcal{T})$. In a k-shot setting, each task consists of $\mathcal{K}$ data points sampled from each of the N classes present in the task. MAML trains f_θ to learn an optimal set of parameters θ' over tasks $\mathcal{T} \in \mathcal{D}_{\text{TRAIN}}$ such that $f_{\theta'}$ converges well on unseen tasks $\mathcal{T} \in \mathcal{D}_{test}$, where $\{\mathcal{D}_{train}, \mathcal{D}_{test}\} \in p(\mathcal{T})$ and $\mathcal{D}_{train} \cap \mathcal{D}_{test} = \phi$

MAML learns an initialization via two optimization loops: 1) Outer loop (learning from all tasks is incorporated to update the model initialization), 2) Inner loop (task-specific adaptation over few gradient update steps is performed). For a given task $\mathcal{T}_i$ sampled from distribution $\mathcal{D}_{train}$, with corresponding loss function $\mathcal{L}_{\mathcal{T}_i}$, the task performs fast adaptation using m gradient steps from initial weights θ_0 as shown:

$$\theta_{\mathcal{T}_i}^m = \theta_{\mathcal{T}_i}^{m-1} - \alpha \nabla_{\theta_{\mathcal{T}_i}} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_{m-1}}) \tag{1}$$

where, $\theta_{\mathcal{T}_i}^0 = \theta_0$. Next, the learning from all the tasks are consolidated to give a generalized performance on tasks sampled from $\mathcal{D}_{test}$. Thus, in the outer loop, one meta-initialization is learned that generalizes across all tasks:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{test}}(f_{\mathcal{T}_i}^{\theta_m}) \tag{2}$$

The disadvantage of MAML is highlighted in Equation 2 where knowledge from all tasks gets consolidated. For tasks from multiple distributions, it is unlikely that all tasks can be adapted from a single meta-initialization, which adversely affects the accuracy. Thus, for multi-distribution training, it becomes necessary to identify distribution-specific and distribution-agnostic parameters. Our work builds upon MAML to address the challenge of deteriorating performance in a multi-distribution setup.

3.1. Binary Mask Perceptron

MAML-based approaches in the literature such as ANIL [15], and BOIL [13] show significant effectiveness in performance over MAML. ANIL freezes all layers except the head in the inner loop optimization step. Our approach dynamically freezing layers depending on task characteristics, instead of freezing a fixed set of parameters. Given a model f_θ, we identify a set of learnable parameters $\theta_{p_i(\mathcal{T})}$ and a learning rate $\alpha_{p_i(\mathcal{T})}$ for input tasks sampled from distribution $p_i(\mathcal{T})$. The proposed method is described in Figure 1 and Algorithm 1.

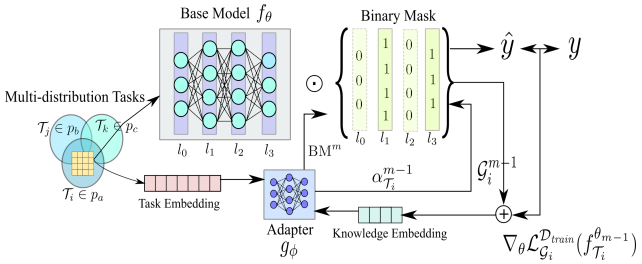


Figure 1. Binary Mask Perceptron

Algorithm 2 Multi-modal Meta Supermasks (MMSUP)

Require: Learning rates β,
Multi-Task Distributions $p_0(\mathcal{T}), p_1(\mathcal{T}), ..., p_{\mathcal{N}-1}(\mathcal{T})$
Ensure: Randomly initialize θ, ϕ
while not done **do**
  Sample a batch of tasks $\mathcal{T}_i \in \{p_0(\mathcal{T}), p_1(\mathcal{T}), ..., p_{\mathcal{N}-1}(\mathcal{T})\}$
  for each $\mathcal{T}_i$ **do**
    Sample datapoints $\{\mathcal{D}_{train} = \{x^{(j)}, y^{(j)}\}$ from $\mathcal{T}_i$
    Compute sparsity % in each layer $k \leftarrow g_\phi(\theta, \nabla_\theta)$
    Compute subnetwork $\mathcal{G}_i$: choose top k% weights in θ_l for $l \in \{0, 1, ..., L\}$
    Evaluate $\mathcal{L}_{\mathcal{G}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_m})$ at each gradient step m by evaluating $\mathcal{L}_{\mathcal{T}_i}$ w.r.t $\mathcal{D}_{train}$
    Compute updates only on subnetwork $\mathcal{G}_i$ using gradient descent: $\theta_{\mathcal{G}_i}^{m+1} = \theta_{\mathcal{G}_i}^{m} - \alpha \nabla_\theta \mathcal{L}_{\mathcal{G}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_m})$
    Weights not present in the subnetwork: $\theta_{\mathcal{T}_i - \mathcal{G}_i}^{m+1} = \theta_{\mathcal{T}_i - \mathcal{G}_i}^{m}$
    Sample datapoints $\{\mathcal{D}_{test} = \{x^{(j)}, y^{(j)}\}$ from $\mathcal{T}_i$ for meta-update.
  end for
  Compute $\mathcal{L}_{\mathcal{G}_i}^{\mathcal{D}_{test}}(f_{\mathcal{T}_i}^{\theta_m})$ by evaluating loss-criterion w.r.t $\mathcal{D}_{test}$.
  Update weights: $\theta_{\mathcal{G}} \leftarrow \theta_{\mathcal{G}} - \alpha \sum_{i=0}^{\mathcal{T}} \big(\nabla_\theta \mathcal{L}_{\mathcal{G}_i - \mathcal{G}'}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_{m-1}}) - \nabla_\theta \mathcal{L}_{\mathcal{G}'}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_{m-1}})\big)/\mathcal{T}$ where, $\mathcal{G} = \mathcal{G}_i \cap \mathcal{G}_j$.
  Update MLP weights: $g_\phi \leftarrow g_\phi - \beta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{G}_i}^{\mathcal{D}_{test}}(f_{\mathcal{G}_i}^{\theta_m})$.
end while

We initialize model f_θ with parameters θ_0, and introduce an adapter network g_ϕ with randomly initialized parameters ϕ_0. For a given input task, g_ϕ takes as input, (1) features from the current task $\mathcal{T}_i$ and (2) prior knowledge stored in the form of weights and gradients in f_θ, to generate two things as output: (1) task-specific learning rate and (2) a binary mask to adaptively mask updates for non-trainable layers in f_θ. Thus, Equation 1 is modified as follows:

$$\theta_{\mathcal{T}_i}^{m,l} = \theta_{\mathcal{T}_i}^{m-1,l} - \alpha_{\mathcal{T}_i}^{m-1}\big(\text{BM}_l^{m-1} \circ \nabla_{\theta_{\mathcal{T}_i}} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_{m-1}})\big) \tag{3}$$

where, $l = 1, 2, ..., L$ is the l^{th} layer of the model f_θ. BM_l^{m-1} is the binary mask $\in \{0, 1\}$ for the l^{th} layer of model at the $(m-1)^{th}$ gradient update step and $\alpha_{\mathcal{T}_i}^{m-1}$ is the learning rate at the $(m-1)^{th}$ for the input task $\mathcal{T}_i$. The binary mask and learning rate are generated at each gradient update step m in the inner loop using adapter network g_ϕ that is a function of $\mathcal{T}_i, \theta$, and $\nabla_\theta \mathcal{L}(f_\theta)$:

$$\text{BM}^m, \alpha_{\mathcal{T}_i}^m = g_\phi\big(\mathcal{T}_i, \theta_{\mathcal{T}_i}^m, \nabla_{\theta_{\mathcal{T}_i}} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_m})\big) \tag{4}$$

We are thus able to generate a binary mask for every $\mathcal{T}_i$ in inner loop. The purpose is to learn the weight of a given layer for learning on $\mathcal{T}_i$ and control the magnitude of the update step. The intuition is that similar tasks will have a larger intersection of shared parameters as compared to dissimilar tasks. Thus, the binary mask is used to efficiently modulate distribution-specific and distribution-agnostic parameters of f_θ. Lastly, the parameters ϕ of g_ϕ are trained in the outer-loop optimization step as follows:

$$\phi \leftarrow \phi - \beta \nabla_\phi \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}^{\mathcal{D}_{test}}(f_{\mathcal{T}_i}^{\theta_m}) \tag{5}$$

Outer-loop optimization for f_θ remains the same as Equation 2. For masking the binary mask with the gradients of f_θ, we make use of Straight Through Estimator (STE) [5] widely used for masking operations [25] and [14]. STE ignores the gradients of the binary mask and backpropagates the gradients unchanged. The implementation details of STE are discussed in the supplementary material. We also provide additional details on the architecture and the input arguments of g_ϕ in the supplementary material.

3.2. Multi-modal Meta Supermasks

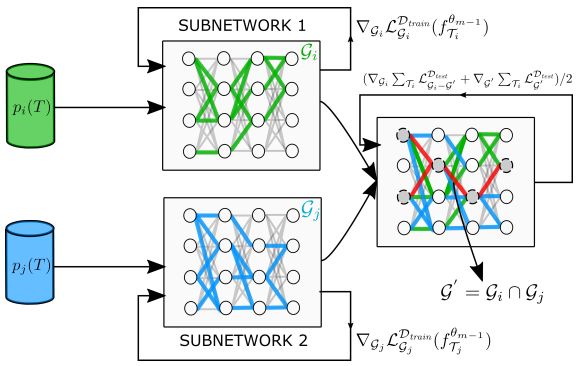


Figure 2. Multi-modal Meta Supermasks

Multi-modal Meta Supermasks (MMSUP) extends BMP to parameter level freezing. We aim to identify a subnetwork that is able to learn efficiently on a given task $\mathcal{T}_i$. The intuition is that if we reduce overlap between the learnable parameters of $\mathcal{T}_i \in p_0(\mathcal{T})$ and $\mathcal{T}_j \in p_1(\mathcal{T})$,

Table 1. 5ways, 1 shot - Same Distribution

Training Distrib.	Testing Distrib.	Meta-learning Architectures							
		MAML		Multi-MAML		BMP		MMSUP	
		Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
CUB200 + VGG102	CUB200	0.506±0.008	6.58	0.533±0.011	11.57	0.563±0.006	6.63	0.524±0.028	7.17
	VGG102	0.703±0.009		0.719±0.011		0.751±0.005		0.723±0.022	
CUB200 + Fungi	CUB200	0.487±0.016	6.33	0.533±0.011	13.58	0.522±0.028	7.46	0.525±0.022	7.79
	Fungi	0.392±0.014		0.424±0.006		0.421±0.034		0.425±0.005	
CUB200 + VGG102 + Fungi	CUB200	0.483±0.015	7.02	0.533±0.011	19.06	0.543±0.002	6.52	0.517±0.027	7.35
	VGG102	0.691±0.011		0.719±0.011		0.733±0.015		0.691±0.017	
	Fungi	0.405±0.006		0.424±0.006		0.441±0.004		0.403±0.003	
CUB200 + VGG102 + Fungi + Aircraft	CUB200	0.479±0.010	5.59	0.533±0.011	28.49	0.517±0.011	7.72	0.487±0.018	8.23
	VGG102	0.677±0.011		0.719±0.011		0.649±0.005		0.689±0.019	
	Fungi	0.398±0.004		0.424±0.006		0.423±0.013		0.380±0.006	
	Aircraft	0.283±0.004		0.400±0.010		0.402±0.025		0.349±0.012	

Table 2. 5ways, 1 shot - Cross Domain Distribution

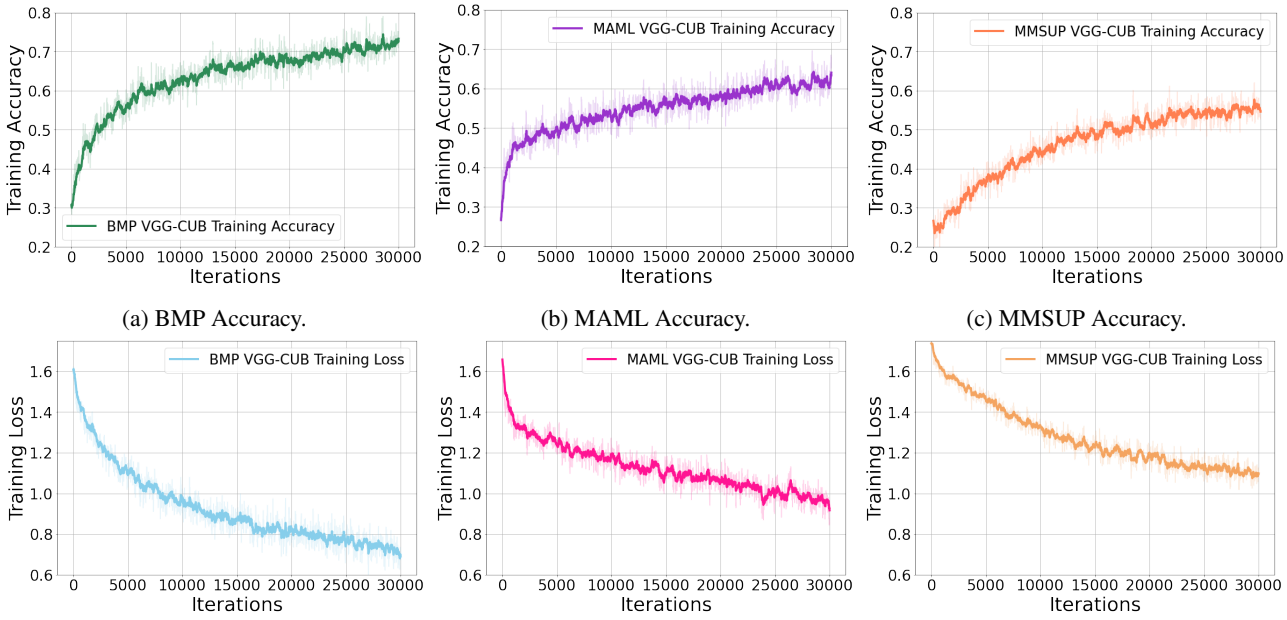
Training Distributions	Testing Distributions	Meta-learning Architectures							
		MAML		Multi-MAML		BMP		MMSUP	
		Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
CUB200 + VGG102	Fungi	0.410	6.58	0.361	11.57	0.411	6.62	0.388	7.17
	Aircraft	0.269		0.277		0.307		0.291	
CUB200 + Fungi	VGG102	0.603	6.33	0.638	13.58	0.684	7.46	0.611	7.8
	Aircraft	0.268		0.277		0.294		0.306	

it might reduce the deterioration in accuracy. Given $\mathcal{N}$ distributions, MMSUP identifies $\mathcal{N}$ subnets $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_\mathcal{N}$ for each distribution while enabling knowledge sharing between distribution-agnostic parameters of the network. For each input distribution type, a subnetwork of parameters is identified. These are parameters that are relevant to a particular distribution. Frankle and Carbin [7] demonstrate the existence of subnetworks that can be trained to achieve accuracy comparable to that of the original network in their Lottery Ticket Hypothesis. [16] builds on this by proposing an edge-pop algorithm to find a subnetwork within a randomly initialized overparameterized network. We improve on the edge-pop algorithm to generate $\mathcal{N}$ subnetworks corresponding to the distributions in the training. Identifying distribution-specific parameters, avoids training of parameters that may not be relevant for a particular distribution and hence avoid training additional irrelevant parameters in the network. Our approach is illustrated in Figure 2 and Algorithm 2.

Given a model f_θ, and training distributions $p_1(\mathcal{T}), p_2(\mathcal{T}), ..., p_\mathcal{N}(\mathcal{T})$, we aim to identify a subset $\mathcal{G}_i$ of parameters of f_θ for task $\mathcal{T}_i$ such that $\mathcal{L}_\mathcal{T}(\mathcal{G}_i) \leq \mathcal{L}_\mathcal{T}(f_\theta)$. Note that our approach is different from [25] as our aim is not to identify a subnetwork, rather it's to identify a subset of parameters that result in minimum deterioration of accuracy. Thus, rather than maintaining a separate score to learn the ideal subnetwork for an input distribution, we learn the relevant weights of the underlying network f_θ. We meta-learn the sparsity % (k%) present in each of the layers of the underlying architecture using a MLP (g_ϕ). We can also keep the sparsity parameter constant, however, we later observe in Section 4 that varying sparsity results in better performance. Inner loop update is similar to Equation 1 with some minor changes:

$$\begin{aligned} \theta_{\mathcal{G}_i}^{m+1} &= \theta_{\mathcal{G}_i}^{m} - \alpha \nabla_\theta \mathcal{L}_{\mathcal{G}_i}^{\mathcal{D}_{train}}(f_{\mathcal{T}_i}^{\theta_m}) \\ \theta_{\mathcal{T}_i - \mathcal{G}_i}^{m+1} &= \theta_{\mathcal{T}_i - \mathcal{G}_i}^{m} \end{aligned} \tag{6}$$

where, $\theta_{\mathcal{G}_i}$ is set of top-k% parameters from each of the layers in f_θ and $\mathcal{G}_i^{'}$ are the task agnostic parameters (parameters common across all the distributions). We don't update the parameters not belonging to the subset. Lastly, the outer-loop optimization step is carried out as follows:



(a) BMP Accuracy. (b) MAML Accuracy. (c) MMSUP Accuracy.

(d) BMP Error. (e) MAML Error. (f) MMSUP Error.

Figure 3. Training accuracy and Loss

$$\theta_{\mathcal{G}} = \theta_{\mathcal{G}} - \alpha\left(\sum_{i=0}^{\mathcal{T}} \frac{\nabla_{\theta}\mathcal{L}^{\mathcal{D}_{train}}_{\mathcal{G}_i-\mathcal{G}'}(f^{\theta_{m-1}}_{\mathcal{T}_i})}{\mathcal{T}}\right) - \alpha\frac{\nabla_{\theta}\mathcal{L}^{\mathcal{D}_{train}}_{\mathcal{G}'}(f^{\theta_{m-1}}_{\mathcal{T}_i})}{\mathcal{T}} \tag{7}$$

$$g_{\phi} \leftarrow g_{\phi} - \beta\nabla_{\theta}\sum_{\mathcal{T}_i}\mathcal{L}^{\mathcal{D}_{test}}_{\mathcal{G}_i}(f^{\theta_m}_{\mathcal{G}_i}) \tag{8}$$

where, $\{\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup ... \cup \mathcal{G}_{\mathcal{N}}\}$.

4. Experiments and Ablation Studies

We evaluate both our approaches in the image-classification domain, using quasi-benchmark datasets from the field of meta-learning. Our approaches are compared with the baselines as MAML and MultiMAML. MAML represents the family of model agnostic metalearners and is known to generalize well on tasks from known distributions and hence forms the baseline for our experiments for training accuracy of the model. Multi-MAML comprises of M (number of different modalities) MAML models and we compare our approach against the large training times incurred in a Multi-MAML scenario. In Multi-MAML a separate model is trained for each input distribution. In case of M input distributions, multi-MAML will train M models. In our experiments we use 5 datasets, and hence in the multi-MAML approach, we will train 5 models (one for each dataset). The binary mask perceptron and the meta supermask approach are both training a single base model, unlike Multi-MAML and are thus compute-optimal. Unlike MAML, both approaches have been designed to work on diverse distributions. We illustrate the efficacy of our approaches on 5 image datasets, namely, CUBirds [24], Aircraft [11], VGG Flowers [12] and Fungi [8]. Details of the datasets and hyper-parameters are outlined in the Supplementary section. All experiments are conducted on a dedicated MIG A100 GPU setup, with 30 GB RAM, 8vCPUS and 10GB GPU memory. Each experiment has been repeated 3 times with different seeds to ensure sufficient randomness.

4.1. Multi-distribution performance of BMP and MMSUP

Table 1 depicts training a single model in a multi-distribution scenario, i.e, the model is trained on 2,3 or 4 datasets. The trained model was tested on unseen tasks from known distributions. BMP and MMSUP outperform base-MAML in terms of accuracy, or achieve a comparable accuracy as both the approaches focus on identifying layers or parameters which are specific to the distribution when training. The training time of both BMP and MMSUP is higher than Multi-MAML. However in most cases, the training time for MAML is lower in both cases. Reason being, BMP requires an adapter to determine the binary mask and MMSUP needs to identify the relevant sub-network for training a particular input distribution. Both these operations add to the overall training time. However, BMP and MMSUP achieve a lower training time compared to Multi-

Table 3. Varying Sparsity

Datasets	Sparsity Value								
	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
VGG102	0.669	0.676	0.676	0.681	0.671	0.681	0.665	0.652	0.653
CUB200	0.509	0.502	0.52	0.515	0.509	0.512	0.501	0.499	0.475

Table 4. Varying Sparsity with depth of network

Datasets	Sparsity Depth							
	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
VGG102	0.671	0.672	0.678	0.679	0.706	0.666	0.664	0.671
CUB200	0.491	0.496	0.488	0.494	0.515	0.503	0.503	0.504

MAML as Multi-MAML trains a separate model for each modality. Figure 3 depicts the accuracy and the loss landscapr for BMP, MMPSUP and MAML. The overall training time for BMP is lower than MMSUP, as MMSUP identifies all network parameters that are relevant for a give input task. BMP, on the other hand, considers entire layers of the network at a time. The accuracy of both approaches is comparable. As BMP is less granular than MMSUP, it may end performing better than MMSUP (accuracy) in cross domain, as all parameters within a layer are selected. BMP may end up selecting a layer which may not have all relevant parameters for a given distribution, but could be relevant to the unseen distribution during inference. We have observed that varying the sparsity in MMSUP gives a relatively better results as compared to keeping sparsity fixed.

Table 2 depicts the cross-domain results, where model is trained on multiple distributions and tested on tasks from unseen distributions. Both BMP and MMSUP achieve a better accuracy than MAML as MAML is known to generalize well on tasks from a similar distribution. The training time is as depicted in Table 1

Table 5. Adaptation Steps for BMP and MMSUP

Algorithm	# Adaptation Steps				
	1	**2**	**3**	**4**	**5**
BMP	0.774	0.764	0.755	0.746	0.739
MMSUP	0.734	0.722	0.703	0.689	0.723

4.2. Ablation Studies

In this section, we perform ablation studies to better understand and analyze the performance of GEMS. We note the training time, and the accuracy on 5-way 1-shot training of two distributions (VGG102 and CUB200 datasets) instances on 4-CONV backbone. All the results in the subsequent subsections have been tested on VGG102 dataset.

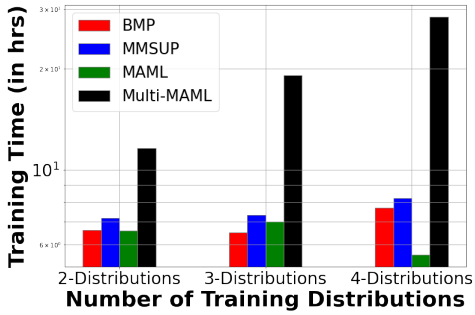


Figure 4. Performance of BMP and MMSUP

4.3. Ablation study on adaptation steps

We first analyze the effectiveness of BMP and MMSUP during rapid learning by varying the number of adaptation steps in the inner loop. As the number of adaptation (gradient) steps increases, the model learns more task-specific parameters. We measure the accuracy and training time for both BMP and MMSUP when trained on different number of adaptation steps as shown Table 5 and Figure 5. As observed in Figure 5, regardless of the number of adaptation steps, both BMP and MMSUP outperform the accuracy of MAML algorithm trained on 5 adaptation steps. Furthermore, performance of BMP improves as adaptation steps decrease. As MMSUP doesn't show a significant change in accuracy, we conclude that increasing the number of adaptation steps doesn't affect the accuracy of the algorithms significantly. This leads us to hypothesize that BMP and MMSUP already learned a good prior as described in [15]. During test time, it results in rapid convergence in just one adaptation step on the input task eventually making additional adaptation steps redundant.

4.4. Ablation study on varying sparsity

To get an understanding of how sparsity affects the performance of MMSUP, we keep the sparsity value constant across all the layers (including head layer) of the 4-CONV backbone. As an example, we plot our observations on

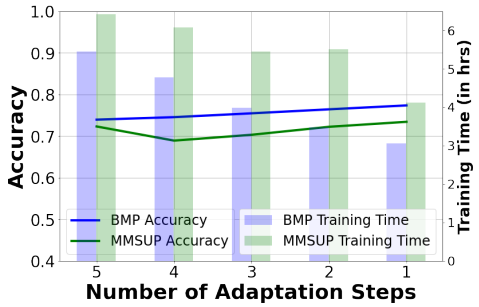


Figure 5. Adaptation Steps

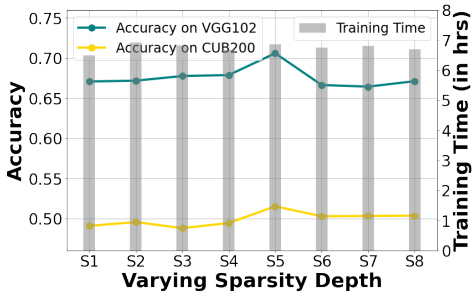


Figure 7. Sparsity Depth

VGG102 and CUB200 datasets in Table 3 and Figure 6. A similar pattern is observed for rest of the distributions. It is observed that as sparsity increases, the accuracy and training time both decrease. However, the accuracy performance for both VGG102 and CUB200 is sub-optimal as compared to MAML trained on zero sparsity. This leads us to conclude that keeping a fixed sparsity value doesn't result in an optimal performance, and recognizing a pattern across layers becomes important.

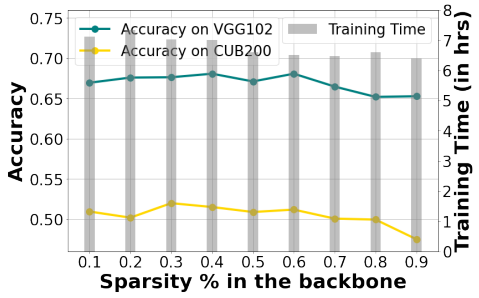


Figure 6. Varying Sparsity

4.5. Ablation study on varying sparsity with depth

Building up on the previous ablation study, we vary the sparsity percentage in each layer and record our observations in Table 4 and Figure 7. We observe that as the sparsity decreases with depth, the accuracy increases. This phenomenon is recorded on VGG102 and CUB200 datasets but the same trend is observed on the other two distributions too. We conclude from this study that MMSUP learns on the initial layers early in the training. This conclusion is consistent with the findings of [15, 13] that manually freeze layers of the backbone in the inner loop update. Thus, automating the process of identifying sparsity in each layer helps MMSUP outperform hand designed algorithms such as ANIL [15] and BOIL [13].

5. Discussion

GBML family of algorithms assume that tasks are sampled from a single distribution. Naively training on tasks from multiple distributions leads to a deterioration in accuracy. Training multiple models, one model per distribution adds to the training time owing to increased compute. Hence we have proposed BMP and MMSUP, both of which train a single model in a multi-distribution scenario. The BMP approach has an adapter which determines a binary mask, thus training only those layers which are relevant for the given input distribution. This also leads to improved training accuracy in a cross-domain scenario. However BMP has a higher training time than MAML, because of additional compute introduced by the adapter to determine the binary mask. MMSUP, goes a step further and determines relevant subnets for each input distribution, thus generalizing well as compared to MAML. However MMSUP has a higher training time as compared to BMP and MAML, as determining a subnet can be computationally expensive, increasing the training time. However, both BMP and MMSUP, beat Multi-MAML in terms of training time as they train a single model on multiple distributions as opposed to Multi-MAML which trains multiple models.

6. Conclusions

In this paper, we outlined two approaches, namely BMP and MMSUP which are capable of training a single model in a multi-distribution scenario. Both the approaches, while having a higher training time as compared to the MAML baseline, outperform the best-case scenario (Multi-MAML) in terms of training time and often in accuracy. We also illustrate the performance in a cross-domain scenario. Both our approaches generalize well on tasks from known as well an unknown distributions. We also present extensive ablation studies which facilitate deeper understanding of BMP and MMSUP approaches, thus validating the efficacy of the approaches. In this paper we have tested our proposed approaches for image classification tasks using quasi-benchmark datasets in the field of metalearning. We plan to further expand this to image segmentation, object detection and reinforcement learning. We also intend to extend this work in a multi-modal scenario, where the model should be capable of generalizing well on input tasks from different modalities, while optimizing the overall training time.

References

[1] Milad Alizadeh, Shyam A. Tailor, Luisa M Zintgraf, Joost van Amersfoort, Sebastian Farquhar, Nicholas Donald Lane, and Yarin Gal. Prospect pruning: Finding trainable weights at initialization using meta-gradients. In *International Conference on Learning Representations*, 2022.

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *International Conference on Learning Representations*, 2019.

[3] Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. *Neural Information Processing Systems*, 34, 2020.

[4] Harkirat Behl, Atılım Güneş Baydin, and Philip H.S. Torr. Alpha maml: Adaptive model-agnostic meta-learning. In *6th ICML Workshop on Automated Machine Learning, Thirty-sixth International Conference on Machine Learning (ICML 2019), Long Beach, CA, US*, 2019.

[5] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017.

[7] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*. OpenReview.net, 2019.

[8] Tobias Guldberg Frøslev, Jacob Heilmann-Clausen, Christian Lange, Thomas Læssøe, Jens Henrik Petersen, Ulrik Søchting, Thomas Stjernegaard Jeppesen, and Jan Vesterholt. Danish mycological society, fungal records database, 2022.

[9] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

[10] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, pages 2933–2942, 2018.

[11] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, University of Massachusetts Amherst, 2013.

[12] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, pages 722–729, 2008.

[13] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. {BOIL}: Towards representation change for few-shot learning. In *International Conference on Learning Representations*, 2021.

[14] Johannes Von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and Joao Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[15] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

[16] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari. What's hidden in a randomly weighted neural network? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11890–11899, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[17] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019.

[18] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook*. Diplomarbeit, Technische Universität München, München, 1987.

[19] J. Schmidhuber. A neural network that embeds its own meta-levels. In *IEEE International Conference on Neural Networks*, pages 407–412 vol.1, 1993.

[20] Jürgen Schmidhuber. On learning how to learn learning strategies. Technical report, Technische Universität München, 1995.

[21] Sebastian Thrun and Lorien Pratt. *Learning to Learn: Introduction and Overview*, page 3–17. Kluwer Academic Publishers, USA, 1998.

[22] Hongduan Tian, Bo Liu, Xiao-Tong Yuan, and Qingshan Liu. Meta-learning with network pruning. In *European Conference on Computer Vision*, pages 675–700. Springer, 2020.

[23] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J. Lim. *Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[24] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge J. Belongie. The caltech-ucsd birds-200-2011 dataset. In *Caltech-UCSD*, 2011.

[25] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.