Meta-Learning with a Geometry-Adaptive Preconditioner

Suhyun Kang[1]*, Duhun Hwang[1]*, Moonjung Eo[1], Taesup Kim[2], Wonjong Rhee[1,3]†
[1] Department of Intelligence and Information, Seoul National University
[2] Graduate School of Data Science, Seoul National University
[3] IPAI and AIIS, Seoul National University
{su_hyun, yelobean, eod87, taesup.kim, wrhee}@snu.ac.kr

Abstract

Model-agnostic meta-learning (MAML) is one of the most successful meta-learning algorithms. It has a bi-level optimization structure where the outer-loop process learns a shared initialization and the inner-loop process optimizes task-specific weights. Although MAML relies on the standard gradient descent in the inner-loop, recent studies have shown that controlling the inner-loop's gradient descent with a meta-learned preconditioner can be beneficial. Existing preconditioners, however, cannot simultaneously adapt in a task-specific and path-dependent way. Additionally, they do not satisfy the Riemannian metric condition, which can enable the steepest descent learning with preconditioned gradient. In this study, we propose Geometry-Adaptive Preconditioned gradient descent (GAP) that can overcome the limitations in MAML; GAP can efficiently meta-learn a preconditioner that is dependent on task-specific parameters, and its preconditioner can be shown to be a Riemannian metric. Thanks to the two properties, the geometry-adaptive preconditioner is effective for improving the inner-loop optimization. Experiment results show that GAP outperforms the state-of-the-art MAML family and preconditioned gradient descent-MAML (PGD-MAML) family in a variety of few-shot learning tasks. Code is available at: https://github.com/Suhyun777/CVPR23-GAP.

1. Introduction

Meta-learning, or *learning to learn*, enables algorithms to quickly learn new concepts with only a small number of samples by extracting prior-knowledge known as meta-knowledge from a variety of tasks and by improving the generalization capability over the new tasks. Among the meta-learning algorithms, the category of optimization-based meta-learning [8, 17, 20, 21, 48] has been gaining popularity due to its flexible applicability over diverse fields including robotics [55, 59], medical image analysis [40, 54], language modeling [37, 42], and object detection [46, 61]. In particular, Model-Agnostic Meta-Learning (MAML) [20] is one of the most prevalent gradient-based meta-learning algorithms.

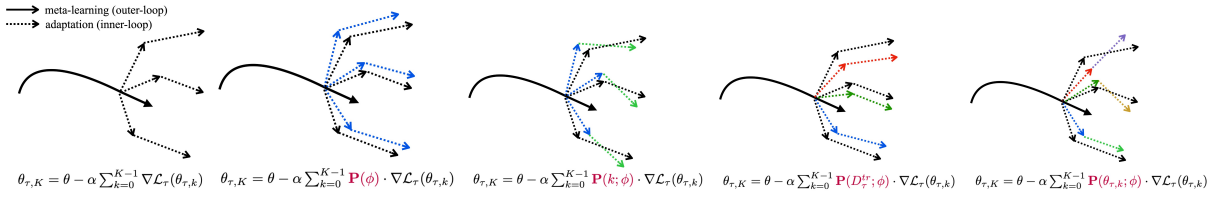
Many recent studies have improved MAML by adopting a Preconditioned Gradient Descent (PGD[1]) for inner-loop optimization [34, 36, 44, 49, 53, 57, 66]. In this paper, we collectively address PGD-based MAML algorithms as the PGD-MAML family. A PGD is different from the ordinary gradient descent because it performs a preconditioning on the gradient using a preconditioning matrix $\mathbf{P}$, also called a *preconditioner*. A PGD-MAML algorithm meta-learns not only the initialization parameter θ_0 of the network but also the meta-parameter ϕ of the preconditioner $\mathbf{P}$.

For the inner-loop optimization, $\mathbf{P}$ was kept static in most of the previous works (Figure 1(b)) [34, 36, 44, 57, 66]. Some of the previous works considered adapting the preconditioner $\mathbf{P}$ with the inner-step k (Figure 1(c)) [49] and some others with the individual task (Figure 1(d)) [53]. They achieved performance improvement by considering individual tasks and inner-step, respectively, and showed that both factors were valuable. However, both factors have not been considered simultaneously in the existing studies.

When a parameter space has a certain underlying structure, there exists a Riemannian metric corresponding the parameter space [3, 4]. If the preconditioning matrix is the Riemannian metric, the preconditioned gradient is known to become the steepest descent on the parameter space [2–4, 6, 27]. An illustration of a toy example is shown in Figure 2. The optimization path of an ordinary gradient descent is shown in Figure 2(a). Compared to the ordinary gradient descent, a preconditioned gradient descent with a preconditioner that does not satisfy the Riemannian metric condition can actually harm the optimization. For the example in Figure 2(b), the preconditioner affects the optimization into an undesirable direction and negatively affects the gradient descent. On the

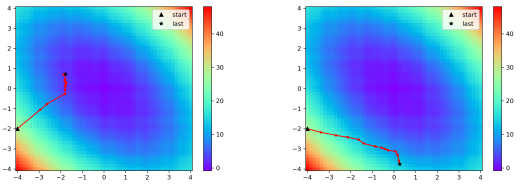
*Equal contribution.
†Corresponding author.

[1]PGD in our work should not be confused with Projected Gradient Descent [13].

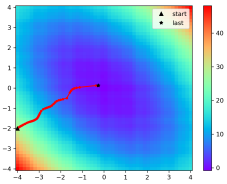


(a) MAML (b) Non-adaptive $\mathbf{P}(\phi)$ (c) Adaptive $\mathbf{P}(k;\phi)$ (d) Adaptive $\mathbf{P}(D_\tau^{tr};\phi)$ (e) Adaptive $\mathbf{P}(\theta_{\tau,k};\phi)$

Figure 1. Diagram of MAML and PGD-MAML family. For the inner-loop adaptation in each diagram, the dotted lines of the same color indicate that they use a common preconditioning matrix (preconditioner). (a) MAML adaptation: no preconditioner is used (*i.e.*, $\mathbf{P} = \mathbf{I}$). (b) $\mathbf{P}(\phi)$: a constant preconditioner is used in the inner-loop where the preconditioner's meta-parameter ϕ is meta-learned. (c) $\mathbf{P}(k;\phi)$: a constant preconditioner is used for each inner-step k. Preconditioner for each step is meta-learned, but $\mathbf{P}(k,\phi)$ is not task-specific. (d) $\mathbf{P}(D_\tau^{\text{tr}};\phi)$: a constant preconditioner is used for each task. Preconditioner for each task is meta-learned, but $\mathbf{P}(D_\tau^{\text{tr}};\phi)$ is not dependent on k. (e) GAP adapts $\mathbf{P}(\theta_{\tau,k};\phi)$: a fully adaptive preconditioner is used where it is *task-specific* and *path-dependent*. Instead of saying 'dependent on k', we specifically say it is *path-dependent* because the exact dependency is on the task-specific parameter set $\theta_{\tau,k}$ that is considerably more informative than k.



(a) Gradient Descent (GD) (b) Preconditioned GD (Non-Riemannian metric)



(c) Preconditioned GD (Riemannian metric)

Figure 2. A toy example for illustrating the effect of Riemannian metric (see Supplementary Section A). When the curvature of the parameter space is poorly conditioned, (a) gradient descent can suffer from the difficulty of finding the solution, (b) a preconditioned gradient descent with a preconditioner that does not satisfy the Riemannian metric condition can suffer further, and (c) a preconditioned gradient descent with a preconditioner that is a Riemannian metric can perform better.

contrary, if the preconditioner is the Riemannian metric corresponding the parameter space, the preconditioned gradient descent can become the steepest descent and can exhibit a better optimization behavior as shown in Figure 2(c). While the Riemannian metric condition (*i.e.*, positive definiteness) is a necessary condition for steepest descent learning, the existing studies on PGD-MAML family did not consider constraining preconditioners to satisfy the condition for Riemannian metric.

In this study, we propose a new PGD method named **G**eometry **A**aptive **P**reconditioned gradient descent (GAP). Specifically, GAP satisfies two desirable properties which have not been considered before. First, GAP's preconditioner $\mathbf{P}_{\text{GAP}}$ is a fully adaptive preconditioner that can adapt to the individual task (*task-specific*) and to the optimization-path (*path-dependent*). The full adaptation is made possible by having the preconditioner depend on the task-specific parameter $\theta_{\tau,k}$ (Figure 1(e)). Second, we prove that $\mathbf{P}_{\text{GAP}}$ is a Riemannian metric. To this end, we force the meta-parameters of $\mathbf{P}_{\text{GAP}}$ to be positive definite. Thus, GAP guarantees the steepest descent learning on the parameter space corresponding to $\mathbf{P}_{\text{GAP}}$. Owing to the two properties, GAP enables a geometry-adaptive learning in inner-loop optimization.

For the implementation of GAP, we utilize the Singular Value Decomposition (SVD) operation to come up with our preconditioner satisfying the desired properties. For the recently proposed large-scale architectures, computational overhead can be an important design factor and we provide a low-computational approximation, *Approximate GAP*, that can be proven to asymptotically approximate the operation of GAP.

To demonstrate the effectiveness of GAP, we empirically evaluate our algorithm on popular few-shot learning tasks; few-shot regression, few-shot classification, and few-shot cross-domain classification. The results show that GAP outperforms the state-of-the-art MAML family and PGD-MAML family.

The **main contributions** of our study can be summarized as follows:

- We propose a new preconditioned gradient descent method called GAP, where it learns a preconditioner that enables a geometry-adaptive learning in the inner-loop optimization.
- We prove that GAP's preconditioner has two desirable properties: (1) It is both task-specific and path-dependent (*i.e.*, dependent on task-specific parameter $\theta_{\tau,k}$). (2) It is a Riemannian metric.

- For large-scale architectures, we provide a low-computational approximation that can be theoretically shown to approximate the GAP method.

- For popular few-shot learning benchmark tasks, we empirically show that GAP outperforms the state-of-the-art MAML family and PGD-MAML family.

2. Background

2.1. Model-Agnostic Meta-Learning (MAML)

The goal of MAML [20] is to find the best initialization that the model can quickly adapt from, such that the model can perform well for a new task. MAML consists of two levels of main optimization processes: inner-loop and outer-loop optimizations. Consider the model $f_\theta(\cdot)$ with parameter θ. For a task $\tau = \{D_\tau^{\text{tr}}, D_\tau^{\text{val}}\}$ sampled from the task distribution $p(\mathcal{T})$, the inner-loop optimization is defined as:

$$\theta_{\tau,K} = \theta_{\tau,0} - \alpha \sum_{k=0}^{K-1} \nabla_{\theta_{\tau,k}} \mathcal{L}_\tau^{\text{in}}(\theta_{\tau,k}; D_\tau^{\text{tr}}) \;\; s.t \;\; \theta_{\tau,0} = \theta, \quad (1)$$

where $\theta_{\tau,k}$ is task-specific parameters for task τ, and α is the learning rate for inner-loop optimization, $\mathcal{L}_\tau^{\text{in}}$ is the inner-loop's loss function, and K is the number of gradient descent steps. With $D_{\tau_i}^{\text{val}}$ in each task, we can define outer-loop optimization as

$$\theta \leftarrow \theta - \beta \nabla_\theta \mathbb{E}_\tau \left[\mathcal{L}_\tau^{\text{out}}(\theta_{\tau,K}; D_\tau^{\text{val}}) \right], \quad (2)$$

where β is the learning rate for outer-loop optimization, and $\mathcal{L}_\tau^{\text{out}}$ is the outer-loop's loss function.

2.2. Preconditioned Gradient Descent (PGD)

PGD is a method that minimizes the empirical risk through a gradient update with a preconditioner that rescales the geometry of the parameter space. Given model parameters θ and task $\tau = \{D_\tau^{\text{tr}}, D_\tau^{\text{val}}\}$, we can define the preconditioned gradient update with a preconditioner $\mathbf{P}$ as follows:

$$\begin{gathered} \theta_{\tau,k+1} = \theta_{\tau,k} - \alpha \mathbf{P} \nabla_{\theta_{\tau,k}} \mathcal{L}_\tau(\theta_{\tau,k}; D_\tau^{\text{tr}}) \\ k = 0, 1, \cdots \text{ and } \theta_{\tau,0} = \theta \end{gathered}, \quad (3)$$

where $\mathcal{L}_\tau(\theta_{\tau,k}; D_\tau^{\text{tr}})$ is the empirical loss for task τ and parameters $\theta_{\tau,k}$. Setting $\mathbf{P} = \mathbf{I}$ recovers Eq. (3) to the basic gradient descent (GD). Choice of $\mathbf{P}$ for exploiting the second-order information includes the inverse Fisher information matrix $\mathbf{F}^{-1}$ which leads to the natural gradient descent (NGD) [4]; the inverse Hessian $\mathbf{H}^{-1}$ which corresponds to the Newton's method [31]; and the diagonal matrix estimation with the past gradients which results in the adaptive gradient methods [18, 28]. They often reduce the effect of pathological curvature and speed up the optimization [5].

2.3. Unfolding: reshaping a tensor into a matrix

In this study, the concept of *unfolding* is used to transform the gradient tensor of convolutional kernels into a matrix form. *Tensor unfolding*, also known as matricization or flattening, is the process of reshaping the elements of an N-dimensional tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ into a matrix [29]. The mode-n unfolding of an N-dimensional tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is defined as:

$$\mathbf{X} \xrightarrow[\text{mode-n unfolding}]{} \mathbf{X}_{[n]} \in \mathbb{R}^{I_n \times I_M}, \text{ where } I_M = \prod_{k \neq n} I_k \quad (4)$$

For example, the weight tensor of a convolutional layer is represented as a 4-D tensor ($\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k_h \times k_w}$), where it is composed of kernels and it can be unfolded into a matrix as one of the following four forms: (1) $\mathbf{W}_{[1]} \in \mathbb{R}^{C_{\text{out}} \times (C_{\text{in}} k_h k_w)}$, (2) $\mathbf{W}_{[2]} \in \mathbb{R}^{C_{\text{in}} \times (C_{\text{out}} k_h k_w)}$, (3) $\mathbf{W}_{[3]} \in \mathbb{R}^{k_h \times (C_{\text{out}} C_{\text{in}} k_w)}$, (4) $\mathbf{W}_{[4]} \in \mathbb{R}^{k_w \times (C_{\text{out}} C_{\text{in}} k_h)}$.

2.4. Riemannian manifold

An n-dimensional Riemannian manifold is defined by a manifold $\mathcal{M}$ and a Riemannian metric $g : \mathcal{M} \to \mathbb{R}^{n \times n}$, which is a smooth function from each point $\boldsymbol{x} \in \mathcal{M}$ to a positive definite matrix [32]. The metric $g(\boldsymbol{x})$ defines the inner product of two tangent vectors for each point of the manifold $\langle \cdot, \cdot \rangle : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \times \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathbb{R}$, where $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ is the tangent space of $\boldsymbol{x}$. For $\boldsymbol{v}, \boldsymbol{w} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, the inner product can be expressed $\langle \boldsymbol{v}, \boldsymbol{w} \rangle = \boldsymbol{v}^T g(\boldsymbol{x}) \boldsymbol{w}$. A Riemannian manifold can be characterized by the curvature of the curves defined by a metric. The curvature of a Riemannian manifold can be computed at each point of the curves, while some manifolds have curvatures of a constant value. For example, the unit sphere $\mathcal{S}$ has constant positive curvature of $+1$.

3. Methodology

In this section, we propose a new preconditioned gradient descent method called GAP in the MAML framework. In Section 3.1, we introduce GAP in the inner-loop optimization and describe how to meta-train GAP in the outer-loop optimization. In Section 3.2, we prove that GAP has two desirable properties. In Section 3.3, we provide a low-computational approximation that can be useful for large-scale architectures.

3.1. GAP: Geometry-Adaptive Preconditioner

3.1.1 Inner-loop optimization

We consider an L-layer neural network $f_\theta(\cdot)$ with parameters $\theta = \{\mathbf{W}^1, \cdots, \mathbf{W}^l, \cdots, \mathbf{W}^L\}$. In the standard MAML with task $\tau \sim p(\mathcal{T})$, each $\mathbf{W}^l$ is adapted with the gradient update as below:

$$\mathbf{W}_{\tau,K}^l \leftarrow \mathbf{W}_{\tau,0}^l - \alpha \cdot \sum_{k=0}^{K-1} \mathbf{G}_{\tau,k}^l \;\; s.t \;\; \mathbf{W}_{\tau,0}^l = \mathbf{W}^l, \quad (5)$$

Algorithm 1 Geometry-Adaptive Preconditioned gradient descent (GAP)

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β_1, β_2: learning rates
Randomly initialize $\theta = \{\mathbf{W}^1, \cdots, \mathbf{W}^L\}$.
Initialize $\phi = \{\mathbf{M}^1, \cdots, \mathbf{M}^L\}$ as identity matrix.
while not converged **do**
  Sample a batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\tau \in \mathcal{T}_i$ **do**
    for inner-loop step $k = 0$ **to** $K-1$ **do**
      for layer $l = 1$ **to** L **do**
        Compute $\mathbf{G}^l_{\tau,k} = \nabla_{\mathbf{W}^l_{\tau,k}} \mathcal{L}^{\text{in}}_\tau(\theta_{\tau,k}; D^{\text{tr}}_\tau)$ using D^{tr}_τ
        Reshape $\mathbf{G}^l_{\tau,k}$ to $\mathbf{G}^l_{\tau,k}$ via Eq. (6)
        Transform $\mathbf{G}^l_{\tau,k}$ to $\tilde{\mathbf{G}}^l_{\tau,k}$ using $\mathbf{M}^l$ via Eq. (8)
        Reshape $\tilde{\mathbf{G}}^l_{\tau,k}$ back to the original form of gradient tensor, $\tilde{\mathbf{G}}^l_{\tau,k}$
        Compute l-layer adapted weight: $\mathbf{W}^l_{\tau,k+1} = \mathbf{W}^l_{\tau,k} - \alpha \cdot \tilde{\mathbf{G}}^l_{\tau,k}$
      end for
    end for
    Compute $\mathcal{L}^{\text{out}}_\tau(\theta_{\tau,K}; D^{\text{val}}_\tau)$ by evaluating $\mathcal{L}^{\text{out}}_\tau$ w.r.t D^{val}_τ.
  end for
  Update the weights and meta parameters:
  $\theta \leftarrow \theta - \beta_1 \nabla_\theta \sum_{\tau \in \mathcal{T}} \mathcal{L}^{\text{out}}_\tau(\theta_{\tau,K}; D^{\text{val}}_\tau)$
  $\phi \leftarrow \phi - \beta_2 \nabla_\phi \sum_{\tau \in \mathcal{T}} \mathcal{L}^{\text{out}}_\tau(\theta_{\tau,K}; D^{\text{val}}_\tau)$
end while

where $\mathbf{G}^l_{\tau,k} = \nabla_{\mathbf{W}^l_{\tau,k}} \mathcal{L}^{\text{in}}_\tau(\theta_{\tau,k}; D^{\text{tr}}_\tau)$ is the gradient with respect to $\mathbf{W}^l_{\tau,k}$ and α is the learning rate for inner-loop optimization. In the GAP, we first use the mode-1 unfolding to reshape the gradient tensor into a matrix form (see Section 2.3). For a convolutional layer (*i.e.*, $\mathbf{G}^l_{\tau,k} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k \times k}$), we reshape the gradient tensor as below:

$$\mathbf{G}^l_{\tau,k} \xrightarrow[\text{unfolding}]{\text{mode-1}} \begin{cases} \mathbf{G}^l_{\tau,k} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} k^2} & \text{if } C_{\text{out}} \leq C_{\text{in}} k^2 \\ \mathbf{G}^l_{\tau,k} \in \mathbb{R}^{C_{\text{in}} k^2 \times C_{\text{out}}} & \text{if } C_{\text{in}} k^2 < C_{\text{out}}, \end{cases} \tag{6}$$

where $\mathbf{G}^l_{\tau,k}$ denotes $(\mathbf{G}^l_{\tau,k})_{[1]}$ for the notational brevity. Note that we chose mode-1 unfolding because it performs best among the four unfolding forms as shown in Table 1. For a linear layer in a matrix form, there is no need for an unfolding. Second, we transform the singular values of the gradient matrix using additional meta parameters $\phi = \{\mathbf{M}^l\}_{l=1}^L$. The meta parameter $\mathbf{M}^l$ are diagonal matrices with positive elements defined as:

$$\mathbf{M}^l = \begin{cases} \text{diag}(\text{Sp}(m^l_1), \cdots, \text{Sp}(m^l_{C_{\text{out}}})) & \text{if } C_{\text{out}} \leq C_{\text{in}} k^2 \\ \text{diag}(\text{Sp}(m^l_1), \cdots, \text{Sp}(m^l_{C_{\text{in}} k^2})) & \text{if } C_{\text{in}} k^2 < C_{\text{out}} \end{cases}, \tag{7}$$

where $m^l_i \in \mathbb{R}$ and $\text{Sp}(x) = \frac{1}{2} \cdot \log(1 + \exp(2 * x))$. They are applied to the gradient matrix as follows:

$$\tilde{\mathbf{G}}^l_{\tau,k} = \mathbf{U}^l_{\tau,k}(\mathbf{M}^l \cdot \mathbf{\Sigma}^l_{\tau,k}) \mathbf{V}^{l\,\text{T}}_{\tau,k}, \tag{8}$$

where $\mathbf{G}^l_{\tau,k} = \mathbf{U}^l_{\tau,k} \mathbf{\Sigma}^l_{\tau,k} \mathbf{V}^{l\,\text{T}}_{\tau,k}$ is the singular value decomposition (SVD) of $\mathbf{G}^l_{\tau,k}$. Finally, we reshape $\tilde{\mathbf{G}}^l_{\tau,k}$ back to its original gradient tensor form $\tilde{\mathbf{G}}^l_{\tau,k}$ (*i.e.*, inverse unfolding). The resulting preconditioned gradient descent of GAP becomes the following:

$$\mathbf{W}^l_{\tau,K} \leftarrow \mathbf{W}^l_{\tau,0} - \alpha \cdot \sum_{k=0}^{K-1} \tilde{\mathbf{G}}^l_{\tau,k} \;\; s.t \;\; \mathbf{W}^l_{\tau,0} = \mathbf{W}^l, \tag{9}$$

where $\tilde{\mathbf{G}}^l_{\tau,k}$ is the preconditioned gradient based on the meta parameters ϕ.

mode-1	mode-2	mode-3	mode-4
54.86 ± 0.85	53.02 ± 0.87	51.23 ± 0.76	51.45 ± 0.77

Table 1. Performance comparison of unfolding types. We performed the experiment with 5-way 1-shot on mini-ImageNet and used the standard Conv-4 backbone.

3.1.2 Outer-loop optimization

For outer-loop optimization, GAP follows the typical process of MAML. Unlike MAML, however, GAP meta-learns two meta parameter sets θ and ϕ as follows:

$$\theta \leftarrow \theta - \beta_1 \nabla_\theta \mathbb{E}_\tau \left[\mathcal{L}^{\text{out}}_\tau(\theta_{\tau,K}; D^{\text{val}}_\tau) \right], \tag{10}$$

$$\phi \leftarrow \phi - \beta_2 \nabla_\phi \mathbb{E}_\tau \left[\mathcal{L}^{\text{out}}_\tau(\theta_{\tau,K}; D^{\text{val}}_\tau) \right], \tag{11}$$

where β_1 and β_2 are the learning rates for the outer-loop optimization. We initialize $\mathbf{M}^l$ as an identity matrix for all l. The training procedure is provided in Algorithm 1.

3.2. Desirable properties of GAP

In this section, we prove that GAP's preconditioner $\mathbf{P}_{\text{GAP}}$ satisfies two desirable properties.

Theorem 1. *Let $\tilde{\mathbf{G}}^l_{\tau,k} \in \mathbb{R}^{m \times n}$ be the 'l-layer k-th inner-step' gradient matrix transformed by meta parameter $\mathbf{M}^l$ for task τ. Then preconditioner $\mathbf{P}_{GAP}$ induced by $\tilde{\mathbf{G}}^l_{\tau,k}$ is a Riemannian metric and depends on the task-specific parameters $\theta_{\tau,k}$.*

The proof and closed form of $\mathbf{P}_{\text{GAP}}$ are provided in Supplementary Section B. The following two properties emerge from the theorem.

Property 1. Dependency on task-specific parameters: Theorem 1 formally shows that $\mathbf{P}_{\text{GAP}}$ depends on the task-specific parameters $\theta_{\tau,k}$. While the previous studies considered a non-adaptive preconditioner $\mathbf{P}(\phi)$ [34, 36, 44, 49, 57, 66] and a partially adaptive preconditioner $\mathbf{P}(k; \phi)$ [49] or $\mathbf{P}(D^{\text{tr}}_\tau; \phi)$ [53], $\mathbf{P}_{\text{GAP}}$ can be considered to be the most advanced adaptive preconditioner because it is fully adaptive (*i.e.*, task-specific and path-dependent) by being dependent on $\theta_{\tau,k}$ as shown in Figure 1.

Property 2. Riemannian metric: If the parameter space has a certain underlying structure, the ordinary gradient of a function $\nabla\mathcal{L}$ does not represent its steepest direction [4].

To define the steepest direction on the parameter space, we need a Riemannian metric $g(\boldsymbol{w})$, which is a positive-definite matrix defined for each parameter $\boldsymbol{w}$. A Riemannian metric defines the steepest descent direction by $-g(\boldsymbol{w})^{-1}\nabla\mathcal{L}$ [4]. If a preconditioning matrix is a Riemannian metric, it defines the geometry of the underlying structure and enables steepest descent learning. Because we prove that $\mathbf{P}_{\text{GAP}}$ is a Riemannian metric for each parameter in Theorem 1, $\mathbf{P}_{\text{GAP}}$ is theoretically guaranteed to enable steepest descent learning on its corresponding parameter space. $\mathbf{P}_{\text{GAP}}$ consists of two factors, a unitary matrix of the inner-loop gradient $\mathbf{U}_{\tau,k}$ and a meta-parameter $\mathbf{M}$; $\mathbf{M}$ enables us to reflect the shared geometry information across the tasks. Task-specific and path-dependent geometry information can be reflected in the metric through $\mathbf{U}_{\tau,k}$. Two factors allow our Riemannian metric to have higher function complexity than a constant metric. For example, we can consider various structures other than a unit-sphere that corresponds to the constant metric of $+1$. Even though $\mathbf{P}_{\text{GAP}}$ is guaranteed to be a Riemannian metric, it is crucial that the meta-learned $\mathbf{P}_{\text{GAP}}$ corresponds to the true parameter space or at least $\mathbf{P}_{\text{GAP}}$ is close enough to be useful. We will discuss this issue in Section 5.

3.3. Approximate GAP: a low-computational approximation of GAP

As presented in Section 3.1, GAP uses an SVD operation. The SVD operation can be burdensome for large-scale networks because it implies that the computational cost can be significantly increased. In recent studies, the use of a large-scale architecture has been emphasized as the key factor for improving the performance of meta-learning [25]. To make use of GAP for large-scale architectures without causing a computational problem, we provide an efficient approximation, named *Approximate GAP*, under an assumption.

Assumption 1. *The elements of the gradient matrix follow an i.i.d. normal distribution with zero means.*

Following [39, 60], we adopt the assumption to have the gradient matrix become orthogonal as n increases. Although the utilization of the assumption is a limiting factor, we empirically confirmed that the row vectors of the gradient matrix are indeed asymptotically orthogonal as n increases (see Figure 3). For the assumption, our approximation can be established as the following.

Theorem 2. *Let $\mathbf{G} \in \mathbb{R}^{m\times n}$ be a gradient matrix and $\tilde{\mathbf{G}}$ be the gradient transformed by meta parameter $\mathbf{M}$. Under the Assumption 1, as n becomes large, $\tilde{\mathbf{G}}$ asymptotically becomes equivalent to $\mathbf{MG}$ as follows:*

$$\tilde{\mathbf{G}} \cong \mathbf{MG} \tag{12}$$

Note that we have chosen the larger dimension of the gradient matrix as n when reshaping with Eq. (6). We provide the proof in Supplementary Section B. This approximation has a clear trade-off between scalability and adaptiveness. Approximate GAP efficiently reduces the computational cost as shown in Table 2, whereas the preconditioner $\mathbf{P}_{\text{GAP}}$ becomes not adaptive but constant. However, Approximate GAP still guarantees the preconditioner to be a Riemannian metric. As shown in Table 4 & 5, Approximate GAP incurs a slight performance drop but it still achieves a high performance owing to the Riemannian metric property.

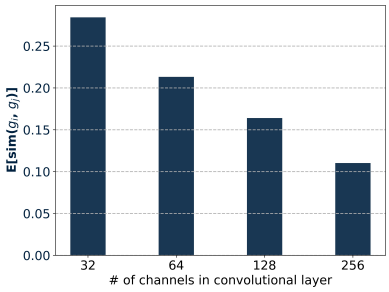


Figure 3. The average of cosine similarity between row vectors of gradient matrix as n increases.

Algorithm	train time (secs)	test time (secs)	GPU-memory (MB)
MAML	35796.61	52.66	10185
GAP	50916.63	120.44	10279
Approximate GAP	36684.72	53.62	10217

Table 2. Comparison of training time, GPU memory, and test time for MAML, GAP, and Approximate GAP. We performed the experiment with 5-way 1-shot on mini-ImageNet and used 600 tasks in the test. We used the standard Conv-4 backbone.

4. Experiments

In this section, we show the superiority of GAP by comparing it with the state-of-the-art PGD-MAML family and the MAML family. Hyper-parameter setups used in our experiments can be found in Supplementary Section C.1.

4.1. Few-shot regression

Datasets and experimental setup. The goal of few-shot regression is to fit an unknown target function for the given K sample points from the function. For the evaluation of few-shot regression, we use the sinusoid regression benchmark [20]. In this benchmark, sinusoid is used as the target function. Each task has a sinusoid $y(x) = A\sin(\omega x + b)$ as the target function, where the parameter values are within the following range: amplitude $A \in [0.1, 5.0]$, frequency $\omega \in [0.8, 1.2]$, and phase $b \in [0, \pi]$. For each task, input data point x is sampled from $[-5.0, 5.0]$. In the experiment, we use a simple Multi-Layer Perceptron (MLP), following the setting in [20]. The details of the architecture are provided in Supplementary Section C.2.1.

Results. We evaluate GAP and compare it with MAML family and PGD-MAML family on a regression task. As

Algorithm	5-shot	10-shot	20-shot
MAML [20]	1.13 ± 0.18	0.77 ± 0.11	0.48 ± 0.08
Meta-SGD† [36]	0.90 ± 0.16	0.53 ± 0.09	0.31 ± 0.05
MT-Net [34]	0.76 ± 0.09	0.49 ± 0.05	0.33 ± 0.04
ALFA [9]	0.92 ± 0.19	0.62 ± 0.16	0.34 ± 0.07
L2F [10]	$0.71 \pm$ N/A	$0.37 \pm$ N/A	$0.16 \pm$ N/A
PAMELA† [49]	0.54 ± 0.06	0.41 ± 0.04	0.17 ± 0.03
MeTAL [8]	0.74 ± 0.18	0.44 ± 0.11	0.21 ± 0.06
GAP†	$\mathbf{0.16 \pm 0.04}$	$\mathbf{0.04 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$

Table 3. Few-shot regression for the sinusoid regression benchmark with a *2-layer MLP* backbone. We report MSE $\pm$ 95% confidence intervals(ci) for 600 tasks following the setup in [20]. † denotes PGD-MAML family.

shown in Table 3, GAP consistently achieves the lowest mean squared error (MSE) scores, with the lowest confidence intervals in all three cases. The performance of GAP is improved by 89% on 10-shot and 94% on 20-shot compared to the performance of state-of-the-art algorithms.

4.2. Few-shot classification

Datasets and experimental setup. For the few-shot classification, we evaluate two benchmarks: (1) mini-ImageNet [56]; this dataset has 100 classes and it is a subset of ImageNet [16], and we use the same split as in [51], with 64, 16 and 20 classes for train, validation and test, respectively. (2) tiered-ImageNet [52]; this is also a subset of ImageNet with 608 classes grouped into 34 high-level categories, and divided into 20, 6 and 8 for train, validation, and test, respectively. For all the experiments, our model follows the standard *Conv-4* backbone used in [56]. The details of the architecture are provided in Supplementary Section C.2.2. Following the experimental protocol in [20], we use 15 samples per class in the query-set to compute the meta gradients. In meta training and meta testing, the inner-loop optimization is updated in five and ten steps, respectively.

Results. Table 4 & 5 present the performance of GAP, the state-of-the-art PGD-MAML family, and the state-of-the-art MAML-family on mini-ImageNet and tiered-ImageNet under two typical settings: 5-way 1-shot and 5-way 5-shot. The GAP outperforms all of the previous PGD-MAML family and MAML family. Compared to the state-of-the-art MAML family, GAP improves the performance with a quite significant margin for both mini-ImageNet and tiered-ImageNet datasets. Compared to the state-of-the-art PGD-MAML family, GAP shows that the 1- and 5-shot accuracy can be increased by 1.4 % and 1.5 % on mini-ImageNet dataset, and by 0.7 % and 0.68 % on tiered-ImageNet dataset, respectively. We also evaluated Approximate GAP that is introduced in Section 3.3. The results show that the approximated version can perform comparably to the original GAP. Although Approximate GAP shows slightly lower accuracies than the original, its performance is superior to most of the existing algorithms because of its Riemannian metric property.

Algorithm	5-way 1-shot	5-way 5-shot
MAML [20]	47.89 ± 1.20	64.59 ± 0.88
Meta-SGD† [36]	50.47 ± 1.87	64.00 ± 0.90
BMAML [64]	53.80 ± 1.46	64.23 ± 0.69
ANIL [48]	46.70 ± 0.40	61.50 ± 0.50
LLAMA [23].	49.40 ± 1.83	N/A
PLATIPUS [21]	50.13 ± 1.86	-
T-net [34]	50.86 ± 1.82	N/A
MT-net [34]	51.70 ± 1.84	N/A
MAML++ [7]	52.15 ± 0.26	68.32 ± 0.44
iMAML-HF [50]	49.30 ± 1.88	N/A
WarpGrad [22]	52.30 ± 0.90	68.40 ± 0.60
MC1† [44]	53.74 ± 0.84	68.01 ± 0.73
MC2† [44]	54.08 ± 0.88	67.99 ± 0.73
MH-C† [66]	48.64 ± 0.33	64.52 ± 0.51
MH† [66]	49.41 ± 0.96	67.16 ± 0.42
BOIL [43]	49.61 ± 0.16	66.46 ± 0.37
ARML [63]	50.42 ± 1.79	64.12 ± 0.90
ALFA [9]	50.58 ± 0.51	69.12 ± 0.47
L2F [10]	52.10 ± 0.50	69.38 ± 0.46
ModGrad† [53]	53.20 ± 0.86	69.17 ± 0.69
PAMELA† [49]	53.50 ± 0.89	70.51 ± 0.67
SignMAML [19]	42.90 ± 1.50	60.70 ± 0.70
CTML [45]	50.47 ± 1.83	64.15 ± 0.90
MeTAL [8]	52.63 ± 0.37	70.52 ± 0.29
ECML [24]	48.94 ± 0.80	65.26 ± 0.67
Sharp-MAML_up [1]	$49.56 \pm$ N/A	$63.06 \pm$ N/A
Sharp-MAML_low [1]	$49.72 \pm$ N/A	$63.18 \pm$ N/A
Sharp-MAML_both [1]	$50.28 \pm$ N/A	$65.04 \pm$ N/A
FBM [62]	50.62 ± 1.79	64.78 ± 0.35
CxGrad [33]	51.80 ± 0.46	69.82 ± 0.42
HyperMAML [47]	51.84 ± 0.57	66.29 ± 0.43
EEML [35]	52.42 ± 1.75	68.40 ± 0.95
MH-O† [66]	52.50 ± 0.61	67.76 ± 0.34
Sparse-MAML† [57]	50.35 ± 0.39	67.03 ± 0.74
Sparse-ReLU-MAML† [57]	50.39 ± 0.89	64.84 ± 0.46
Sparse-MAML+† [57]	51.04 ± 0.59	67.03 ± 0.74
Approximate GAP†	53.52 ± 0.88	70.75 ± 0.67
GAP†	$\mathbf{54.86 \pm 0.85}$	$\mathbf{71.55 \pm 0.61}$

Table 4. 5-way few-shot classification accuracy (%) on mini-ImageNet with a *Conv-4* backbone. We report mean $\pm$ 95% confidence intervals(ci) for 600 tasks according to [20]. † denotes PGD-MAML family.

Algorithm	5-way 1-shot	5-way 5-shot
Meta-SGD† [36]	50.92 ± 0.93	69.28 ± 0.80
MAML [20]	51.70 ± 1.80	70.30 ± 1.80
MT-net [34]	51.95 ± 1.83	N/A
WarpGrad [22]	57.20 ± 0.90	74.10 ± 0.70
BOIL [43].	48.58 ± 0.27	69.37 ± 0.12
ALFA [9]	53.16 ± 0.49	70.54 ± 0.46
L2F [10]	54.40 ± 0.50	73.34 ± 0.44
ARML [63]	52.91 ± 1.83	N/A
PAMELA† [49]	54.81 ± 0.88	74.39 ± 0.71
Sparse-ReLU-MAML† [57]	53.18 ± 0.52	69.06 ± 0.28
Sparse-MAML† [57]	53.47 ± 0.53	68.83 ± 0.65
Sparse-MAML+† [57]	53.91 ± 0.67	69.92 ± 0.21
MeTAL [8]	54.34 ± 0.31	70.40 ± 0.21
CxGrad [33]	55.55 ± 0.46	73.55 ± 0.41
ECML [24]	47.34 ± 0.88	64.77 ± 0.75
Approximate GAP†	56.86 ± 0.91	74.41 ± 0.72
GAP†	$\mathbf{57.60 \pm 0.93}$	$\mathbf{74.90 \pm 0.68}$

Table 5. 5-way few-shot classification accuracy (%) on tiered-ImageNet dataset with a *Conv-4* backbone. We report mean $\pm$ 95% confidence intervals(ci) for 600 tasks according to [20]. † denotes PGD-MAML family.

Algorithm	tiered-ImageNet		CUB		Cars	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML [20]	51.61 ± 0.20	65.76 ± 0.27	40.51 ± 0.08	53.09 ± 0.16	33.57 ± 0.14	44.56 ± 0.21
ANIL [48]	52.82 ± 0.29	66.52 ± 0.28	41.12 ± 0.15	55.82 ± 0.21	34.77 ± 0.31	46.55 ± 0.29
BOIL [43]	53.23 ± 0.41	69.37 ± 0.23	44.20 ± 0.15	60.92 ± 0.11	36.12 ± 0.29	50.64 ± 0.22
BMAML [64]	N/A	N/A	33.52 ± 0.36	51.35 ± 0.16	N/A	N/A
ALFA [9]	N/A	N/A	N/A	58.35 ± 0.25	N/A	N/A
L2F [10]	N/A	N/A	N/A	60.89 ± 0.22	N/A	N/A
MeTAL [8]	N/A	N/A	N/A	58.20 ± 0.24	N/A	N/A
HyperMAML [47]	N/A	N/A	36.52 ± 0.61	49.43 ± 0.14	N/A	N/A
CxGrad [33]	N/A	N/A	N/A	63.92 ± 0.44	N/A	N/A
Sparse-MAML† [57]	53.47 ± 0.53	68.83 ± 0.65	41.37 ± 0.73	60.58 ± 1.10	35.90 ± 0.50	52.63 ± 0.56
Sparse-ReLU-MAML† [57]	53.77 ± 0.94	68.12 ± 0.69	42.89 ± 0.45	57.53 ± 0.94	36.04 ± 0.55	49.95 ± 0.42
Sparse-MAML+† [57]	53.91 ± 0.67	69.92 ± 0.21	43.43 ± 1.04	62.02 ± 0.78	37.14 ± 0.77	53.18 ± 0.44
Approximate GAP†	57.47 ± 0.99	71.66 ± 0.76	43.77 ± 0.79	62.92 ± 0.73	37.00 ± 0.75	53.28 ± 0.76
GAP†	$\mathbf{58.56 \pm 0.93}$	$\mathbf{72.82 \pm 0.77}$	$\mathbf{44.74 \pm 0.75}$	$\mathbf{64.88 \pm 0.72}$	$\mathbf{38.44 \pm 0.77}$	$\mathbf{55.04 \pm 0.77}$

Table 6. 5-way few-shot cross domain classification accuracy (%) with a *Conv-4* backbone, meta training on mini-ImageNet dataset, and meta-testing on tiered-ImageNet, CUB, or Cars datasets. We report mean $\pm$ 95% confidence intervals(ci) for 600 tasks according to [20]. † denotes PGD-MAML family.

4.3. Cross-domain few-shot classification

The cross-domain few-hot classification introduced by [14] addresses a more challenging and practical few-shot classification scenario in which meta-train and meta-test tasks are sampled from different task distributions. These scenarios are designed to evaluate meta-level overfitting of meta-learning algorithms by creating a large domain gap between meta-trains and meta-tests. In particular, an algorithm can be said to be meta-overfitting if it relies too much on the prior knowledge of previously seen meta-train tasks instead of focusing on a few given examples to learn a new task. This meta-level overfitting makes the learning system more likely to fail to adapt to new tasks sampled from substantially different task distributions.

Datasets and experimental setup. To evaluate the level of meta-overfitting for GAP, we evaluate a cross-domain few-shot classification experiment. The mini-ImageNet is used for the meta-train task, and the tiered-ImageNet [52], CUB-200-2011 [58], Cars [11] datasets are used for the meta-test task. The CUB dataset has 200 fine-grained classes and consists of a total of 11,788 images; it is further divided into 100 meta-train classes, 50 meta-validation classes, and 50 meta-test classes. The Cars [30] dataset consists of 16,185 images of 196 classes of cars; it is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in 50-50. The classes are typically at the level of Make, Model, Year, *e.g.*, 2012 Tesla Model S or 2012 BMW M3 coupe. As with the few-shot classification experiment, we use the standard *Conv-4* backbone and follow the same experimental protocol.

Results. Table 6 presents the cross-domain few-shot performance for GAP, MAML family, and PGD-MAML family. GAP significantly outperforms the state-of-the-art algorithms on 5-way 1-shot and 5-way 5-shot cross-domain classification tasks. In particular, for the tiered-ImageNet dataset, the performance was improved by 8.6% and 4.1% on 1-shot and 5-shot classification tasks, respectively. Because GAP can simultaneously consider a task's individuality and optimization trajectory in the inner-loop optimization, it can overcome meta-overfitting better than the existing methods. However, Approximate GAP shows more performance degradation in cross-domain few-shot classification than in few-shot classification. In particular, when the domain difference with the meta-train is more significant (*i.e.*, the tiered-ImageNet dataset) than when the domain difference with the meta-train is marginal (*i.e.*, CARS and CUB datasets), it shows a more considerable performance drop. We can see that full adaptation plays an important role in cross-domain few-shot classification.

5. Discussion

Number of meta parameters: Recent MAML family and PGD-MAML family require a large increase in the number of meta-learning parameters as shown in Table 7. One advantage of GAP is that it requires only a very small increase in the number of meta parameters, when compared to the baseline of MAML. This is possible because we transform a gradient tensor into a gradient matrix, perform the SVD of the matrix, and assign only a small number of meta parameters that correspond to the diagonal matrix of the gradient matrix. For the *Conv-4* network, GAP requires only 0.2% increase of the meta parameters. Although the increase in the number of meta parameters is negligible, SVD of the gradient matrix can incur a large computational burden for large networks. This is addressed by Approximate GAP.

Approximate GAP vs. simple constant preconditioners: Approximate GAP is a low-complexity method where SVD

Algorithm	# of params	% increase
MAML [20]	1.2109×10^5	
Meta-SGD [36]	2.4218×10^5	100.0%
MC [44]	2.7106×10^6	2140.4%
PAMELA [49]	1.6239×10^5	34.1%
MH [66]	7.2196×10^7	59586.7%
Sparse-MAML [57]	2.4218×10^5	100.0%
GAP	$\mathbf{1.2131 \times 10^5}$	**0.2%**

Table 7. Comparison of the number of parameters for MAML, existing methods, and GAP.

Algorithm	Structure	Riemannian metric (*i.e.*, positive definite)	Acc. (%)
Meta-SGD	$\text{diag}(a_1, \cdot, a_n)$	X	50.47%
Meta-SGD with positive definiteness	$\text{diag}(a_1, \cdot, a_n)$	O	52.39%
Approximate GAP	$\text{blkdiag}(M, \cdots, M)$	O	**53.52%**

Table 8. Performance comparison of three constant (*i.e.*, non-adaptive) preconditioners for 5-way 1-shot on mini-ImageNet: Meta-SGD, Meta-SGD modified to satisfy positive definiteness, and our Approximate GAP.

Algorithm	1-shot	5-shot
GAP w/o $\mathbf{P}_{\text{GAP}}$	48.23 ± 0.80	65.80 ± 0.75
GAP w/ $\mathbf{P}_{\text{GAP}}$	54.86 ± 0.85	71.55 ± 0.61

Table 9. Ablation study of $\mathbf{P}_{\text{GAP}}$ on mini-ImageNet. Performance of the GAP-trained model is significantly affected by not applying $\mathbf{P}_{\text{GAP}}$.

operation is avoided by approximating GAP with a constant diagonal preconditioner. A natural question to ask is how does Approximate GAP compare with other constant diagonal preconditioners. To answer this question, we have compared Approximate GAP with Meta-SGD and a modified Meta-SGD. Meta-SGD [36] is a well-known constant diagonal preconditioner (*i.e.*, $\text{diag}(a_1, \cdots, a_n)$) that does not need to be positive definite and we also investigate its modification with a constraint on positive definiteness. The results are shown in Table 8. It can be observed that enforcing positive definiteness can improve Meta-SGD. Furthermore, an additional improvement can be achieved by Approximate GAP. While both modified Meta-SGD and Approximate GAP are positive definite, Approximate GAP is different because it inherits an additional constraint from GAP – a block diagonal structure where a constant diagonal matrix M is repeated (*i.e.*, $\text{blkdiag}(M, \cdots, M)$). The inherited constraint provides a gain over the modified Meta-SGD.

Does GAP learn a useful preconditioner: While a Riemannian metric can be helpful, it does not mean any Riemannian metric will result in an improvement. For the true parameter space with a specific underlying structure, the corresponding Riemannian metric needs to be applied to enable steepest descent [3, 4]. For the special case of a two-layer neural network with a mean squared error (MSE) loss, it was proven that Fisher information matrix is the corresponding Riemannian metric [4]. For a general neural network, however, a proper Riemannian metric is unknown and it needs to be learned. In our work, we have devised a method to guarantee a Riemannian metric and have used the outer-loop optimization to learn the Riemannian metric. In general, the learned Riemannian metric is unlikely to correspond perfectly to the true parameter space. Then, an important question is if the Riemannian metric learned by GAP is close enough to the desired one and if it is useful. To investigate this issue, we performed an ablation study by not applying the preconditioner $\mathbf{P}_{\text{GAP}}$. After training a GAP model, we have evaluated the performance with and without applying $\mathbf{P}_{\text{GAP}}$. The results are shown in Table 9 and clearly the preconditioner learned with the outer-loop optimization plays an essential role for improving the performance.

Why is preconditioner helpful for meta-learning: When the batch size is small, the resulting empirical gradient can be noisy [53, 65]. A typical few-shot learning has only a small number of samples for the inner-loop optimization, and its gradient can be noisy. On the other hand, it was shown in [5] that preconditioned gradient descent with a positive definite preconditioner can achieve a lower risk than gradient descent when the labels are noisy, the model is mis-specified, or the signal is misaligned with the features. Under a misalignment, a properly chosen positive definite preconditioner can generalize better than gradient descent [5]. Considering the noisy gradient of inner loop optimization, it can be surmised that a positive definite preconditioner that is adaptive (*i.e.*, a Riemannian metric) can be helpful for improving MAML. Note that the noisy case is in contrast to the case of supervised learning with a large amount of data [5].

6. Conclusion

In this work, we have proposed GAP that is a PGD-MAML algorithm utilizing SVD operation. Thanks to the fully adaptive property, GAP can handle individual tasks with a wide diversity within the MAML framework. Additionally, GAP can enable steepest descent on the parameter space owing to the Riemannian metric property.

Acknowledgements

This work was supported by ETRI [23ZR1100, A Study of Hyper-Connected Thinking Internet Technology by autonomous connecting, controlling and evolving ways], NRF (NRF-2020R1A2C2007139), IITP [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)], and the New Faculty Startup Fund from Seoul National University.

References

[1] Momin Abbas, Quan Xiao, Lisha Chen, Pin-Yu Chen, and Tianyi Chen. Sharp-maml: Sharpness-aware model-agnostic meta learning. *arXiv preprint arXiv:2206.03996*, 2022. 6

[2] Shunichi Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, (3):299–307, 1967. 1

[3] Shun-ichi Amari. Neural learning in structured parameter spaces-natural riemannian gradient. *Advances in neural information processing systems*, 9, 1996. 1, 8

[4] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998. 1, 3, 4, 5, 8

[5] Shun-ichi Amari, Jimmy Ba, Roger Grosse, Xuechen Li, Atsushi Nitanda, Taiji Suzuki, Denny Wu, and Ji Xu. When does preconditioning help or hurt generalization? *arXiv preprint arXiv:2006.10732*, 2020. 3, 8

[6] Shun-Ichi Amari and Scott C Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 2, pages 1213–1216. IEEE, 1998. 1

[7] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018. 6

[8] Sungyong Baik, Janghoon Choi, Heewon Kim, Dohee Cho, Jaesik Min, and Kyoung Mu Lee. Meta-learning with task-adaptive loss function for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9465–9474, 2021. 1, 6, 7

[9] Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. *Advances in Neural Information Processing Systems*, 33:20755–20765, 2020. 6, 7

[10] Sungyong Baik, Seokil Hong, and Kyoung Mu Lee. Learning to forget for meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2379–2387, 2020. 6, 7

[11] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018. 7

[12] Irénée-Jules Bienaymé. *Considérations à l'appui de la découverte de Laplace sur la loi de probabilité dans la méthode des moindres carrés*. Imprimerie de Mallet-Bachelier, 1853. 13

[13] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 1

[14] 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. 7

[15] Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: A meta-learning library for pytorch. *arXiv preprint arXiv:1909.06576*, 2019. 14

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[17] Lin Ding, Peng Liu, Wenfeng Shen, Weijia Lu, and Shengbo Chen. Gradient-based meta-learning using uncertainty to weigh loss for few-shot learning. *arXiv preprint arXiv:2208.08135*, 2022. 1

[18] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. 3

[19] Chen Fan, Parikshit Ram, and Sijia Liu. Sign-maml: Efficient model-agnostic meta-learning by signsgd. *arXiv preprint arXiv:2109.07497*, 2021. 6

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 1, 3, 5, 6, 7, 8, 14

[21] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018. 1, 6

[22] Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2019. 6

[23] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018. 6

[24] Markus Hiller, Mehrtash Harandi, and Tom Drummond. On enforcing better conditioned meta-learning for rapid few-shot adaptation. *arXiv preprint arXiv:2206.07260*, 2022. 6

[25] Shell Xu Hu, Da Li, Jan Stühmer, Minyoung Kim, and Timothy M Hospedales. Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9068–9077, 2022. 5

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 14

[27] Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001. 1

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3, 14

[29] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. 3

[30] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 7

[31] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012. 3

[32] John M Lee. *Smooth manifolds*. Springer, 2012. 3, 12

[33] Sanghyuk Lee, Seunghyun Lee, and Byung Cheol Song. Contextual gradient scaling for few-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 834–843, 2022. 6, 7

[34] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2927–2936. PMLR, 2018. 1, 4, 6

[35] Geng Li, Boyuan Ren, and Hongzhi Wang. Eeml: Ensemble embedded meta-learning. *arXiv preprint arXiv:2206.09195*, 2022. 6

[36] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017. 1, 4, 6, 8

[37] Zequn Liu, Ruiyi Zhang, Yiping Song, and Ming Zhang. When does maml work the best? an empirical study on model-agnostic meta-learning in nlp applications. *arXiv preprint arXiv:2005.11700*, 2020. 1

[38] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 14

[39] Ahmed M Abdelmoniem, Ahmed Elzanaty, Mohamed-Slim Alouini, and Marco Canini. An efficient statistical-based gradient compression technique for distributed training systems. *Proceedings of Machine Learning and Systems*, 3:297–322, 2021. 5

[40] Gabriel Maicas, Andrew P Bradley, Jacinto C Nascimento, Ian Reid, and Gustavo Carneiro. Training medical image analysis systems like radiologists. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 546–554. Springer, 2018. 1

[41] George Marsaglia. Choosing a point from the surface of a sphere. *The Annals of Mathematical Statistics*, 43(2):645–646, 1972. 13

[42] Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *arXiv preprint arXiv:1905.05644*, 2019. 1

[43] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. Boil: Towards representation change for few-shot learning. *arXiv preprint arXiv:2008.08882*, 2020. 6, 7

[44] Eunbyung Park and Junier B Oliva. Meta-curvature. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 4, 6, 8

[45] Danni Peng and Sinno Pan. Clustered task-aware meta-learning by learning from learning paths. 2021. 6

[46] Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M Hospedales, and Tao Xiang. Incremental few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13846–13855, 2020. 1

[47] M Przewięźlikowski, P Przybysz, J Tabor, M Zięba, and P Spurek. Hypermaml: Few-shot adaptation of deep models with hypernetworks. *arXiv preprint arXiv:2205.15745*, 2022. 6, 7

[48] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019. 1, 6, 7

[49] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. Meta-learning the learning trends shared across tasks. *arXiv preprint arXiv:2010.09291*, 2020. 1, 4, 6, 8

[50] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019. 6

[51] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016. 6

[52] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018. 6, 7

[53] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. On modulating the gradient for meta-learning. In *European Conference on Computer Vision*, pages 556–572. Springer, 2020. 1, 4, 6, 8

[54] Rishav Singh, Vandana Bharti, Vishal Purohit, Abhinav Kumar, Amit Kumar Singh, and Sanjay Kumar Singh. Metamed: Few-shot medical image classification using gradient-based meta-learning. *Pattern Recognition*, 120:108111, 2021. 1

[55] Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, and Jie Tan. Rapidly adaptable legged robots via evolutionary meta-learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776. IEEE, 2020. 1

[56] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016. 6, 14

[57] Johannes Von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. *Advances in Neural Information Processing Systems*, 34:5250–5263, 2021. 1, 4, 6, 7, 8

[58] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 7

[59] Shuhuan Wen, Zeteng Wen, Di Zhang, Hong Zhang, and Tao Wang. A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning. *Applied Soft Computing*, 110:107605, 2021. 1

[60] Simon Wiedemann, Temesgen Mehari, Kevin Kepp, and Wojciech Samek. Dithered backprop: A sparse and quantized backpropagation algorithm for more efficient deep neural network training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 720–721, 2020. 5

[61] Xiongwei Wu, Doyen Sahoo, and Steven Hoi. Meta-rcnn: Meta learning for few-shot object detection. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1679–1687, 2020. 1

[62] Peng Yang, Shaogang Ren, Yang Zhao, and Ping Li. Calibrating cnns for few-shot meta learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2090–2099, 2022. 6

[63] Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. Automated relational meta-learning. *arXiv preprint arXiv:2001.00745*, 2020. 6

[64] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018. 6, 7

[65] Yikai Zhang, Hui Qu, Chao Chen, and Dimitris Metaxas. Taming the noisy gradient: train deep neural networks with small batch sizes. In *The Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. 8

[66] Dominic Zhao, Johannes von Oswald, Seijin Kobayashi, João Sacramento, and Benjamin F Grewe. Meta-learning via hypernetworks. 2020. 1, 4, 6, 8