Task-Specific Preconditioner for Cross-Domain Few-Shot Learning

Suhyun Kang[1], **Jungwon Park**[2], **Wonseok Lee**[3], **Wonjong Rhee**[2,3*]

[1] Samsung Research, Seoul, South Korea
[2] Department of Intelligence and Information, Seoul National University, Seoul, South Korea
[3] IPAI, Seoul National University, Seoul, South Korea
su1019.kang@samsung.com; {quoded97, dnjstjr1017, wrhee}@snu.ac.kr

Abstract

Cross-Domain Few-Shot Learning (CDFSL) methods typically parameterize models with task-agnostic and task-specific parameters. To adapt task-specific parameters, recent approaches have utilized fixed optimization strategies, despite their potential sub-optimality across varying domains or target tasks. To address this issue, we propose a novel adaptation mechanism called Task-Specific Preconditioned gradient descent (TSP). Our method first meta-learns Domain-Specific Preconditioners (DSPs) that capture the characteristics of each meta-training domain, which are then linearly combined using task-coefficients to form the Task-Specific Preconditioner. The preconditioner is applied to gradient descent, making the optimization adaptive to the target task. We constrain our preconditioners to be positive definite, guiding the preconditioned gradient toward the direction of steepest descent. Empirical evaluations on the Meta-Dataset show that TSP achieves state-of-the-art performance across diverse experimental scenarios.

1 Introduction

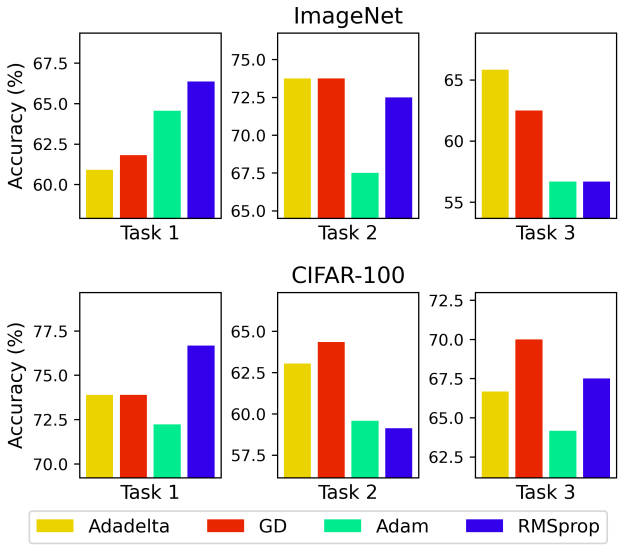
Few-Shot Learning (FSL) aims to learn a model that can generalize to novel classes using a few labeled examples. Recent advancements in FSL have been significantly propelled by meta-learning methods (Snell, Swersky, and Zemel 2017; Finn, Abbeel, and Levine 2017; Sung et al. 2018; Oreshkin, Rodríguez López, and Lacoste 2018; Garnelo et al. 2018; Rajeswaran et al. 2019). These approaches have achieved outstanding results in single domain FSL benchmarks such as Omniglot (Lake et al. 2011) and *mini*Imagenet (Ravi and Larochelle 2016). However, recent studies (Chen et al. 2019; Tian et al. 2020) have revealed that many existing FSL methods struggle to generalize in cross-domain setting, where the test data originates from domains that are either unknown or previously unseen. To study the challenge of generalization in cross-domain few-shot tasks, Triantafillou et al. (2019) introduced the *Meta-Dataset*, a more realistic, large-scale, and diverse benchmark. It includes multiple datasets from a variety of domains for both meta-training and meta-testing phases.

*Corresponding author

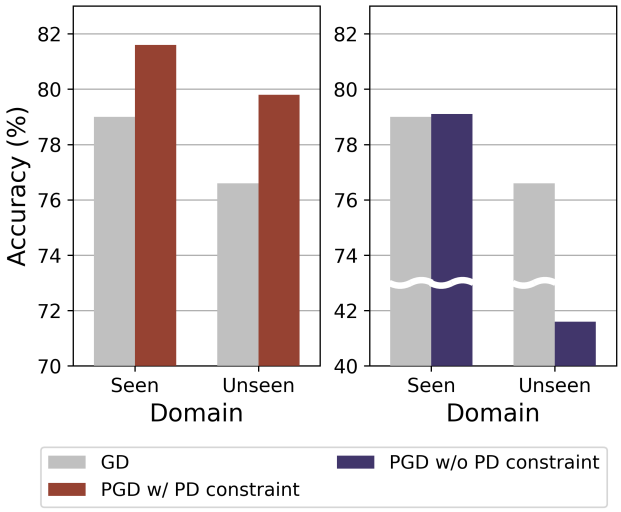
Leveraging the Meta-Dataset, various Cross-Domain Few-Shot Learning (CDFSL) methods have been developed (Requeima et al. 2019; Bateni et al. 2020, 2022; Liu et al. 2021; Triantafillou et al. 2021; Li, Liu, and Bilen 2021, 2022; Dvornik, Schmid, and Mairal 2020; Liu et al. 2020; Guo et al. 2023; Tian et al. 2024), demonstrating significant advancements in this field. These approaches typically parameterize deep neural networks with a large set of task-agnostic parameters alongside a smaller set of task-specific parameters. Task-specific parameters are optimized to the target task through an adaptation mechanism, generally following one of two primary methodologies. The first approach utilizes an auxiliary network functioning as a parameter generator, which, upon receiving a few labeled examples from the target task, outputs optimized task-specific parameters (Requeima et al. 2019; Bateni et al. 2020, 2022; Liu et al. 2020, 2021). The second approach directly finetunes the task-specific parameters through gradient descent using a few labeled examples from the target task (Dvornik, Schmid, and Mairal 2020; Li, Liu, and Bilen 2021; Triantafillou et al. 2021; Li, Liu, and Bilen 2022; Tian et al. 2024).

While both approaches have improved CDFSL performance through adaptation mechanism, a common limitation persists in the optimization strategies employed by these methods. Specifically, both approaches employ a fixed optimization strategy across different target tasks. However, Figure 1a shows that the optimal choice of optimizer may vary significantly depending on the given domain or target task. This implies that the performance can be significantly improved by adapting an optimization strategy to align well with the target domain and task. However, devising an effective and reliable scheme for its implementation has been challenging.

One promising approach for establishing a robust adaptive optimization scheme is to leverage Preconditioned Gradient Descent (PGD) (Himmelblau et al. 2018). PGD operates by specifying a preconditioning matrix, often referred to as a *preconditioner*, which re-scales the geometry of the parameter space. In the field of machine learning, previous research has shown that if the preconditioner is positive definite (PD), it establishes a valid Riemannian metric, which represents the geometric characteristics (e.g., curvature) of the parameter space and steers preconditioned gradients in



(a)



(b)

Figure 1: All experiments are conducted baed on TSA. (a) The optimal optimization strategy can vary significantly depending on the nature of the target task, leading to notable differences in performance on the Meta-Dataset. (b) The accuracy of seen and unseen for the Meta-Dataset. Compared to the baseline of using gradient descent, adopting a preconditioner without a PD constraint can be unreliable. With a PD constraint, it becomes reliable to adapt the preconditioner to the target task. Further details on these preconditioners are provided in Appendix A.

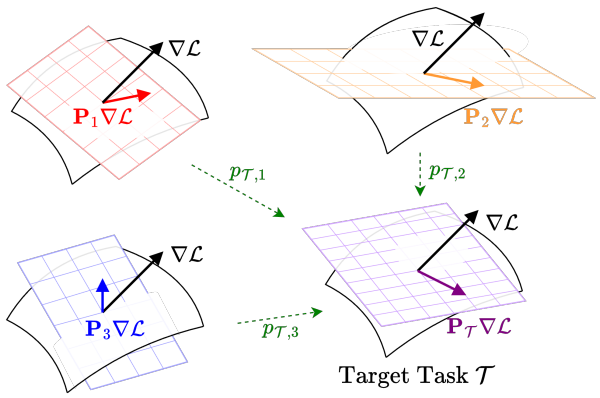


Figure 2: Illustration of forming a Task-Specific Preconditioner based on three DSPs that have been meta-trained for three meta-training domains.

the direction of steepest descent (Amari 1967, 1996, 1998; Amari and Douglas 1998). While the effectiveness of positive definiteness in PGD is supported by existing theoretical findings, its efficacy as an adaptive optimization scheme in CDFSL can be examined through a simple comparison. In Figure 1b, we compare PGD with and without a PD constraint for the preconditioner on the Meta-Dataset. Without a PD constraint, PGD shows markedly inferior performance, especially in unseen domains. Conversely, with a PD constraint, PGD consistently exhibits performance improvements across seen and unseen domains compared to the baseline using GD. This supports the pivotal role of positive definiteness in PGD for CDFSL.

Inspired by these findings, we introduce a novel adaptation mechanism named Task-Specific Preconditioned gradient descent (TSP). In our approach, we establish a Task-Specific Preconditioner that is constrained to be positive definite and adapt it to the specific nature of the target task. This preconditioner consists of two components. The first component is the Domain-Specific Preconditioners (DSPs), which are uniquely defined for each meta-training domain and meta-trained on tasks sampled from these domains through bi-level optimization during the meta-training phase. The second component is task-coefficient, which approximates the compatibility between the target task and each meta-training domain. Figure 2 illustrates the construction of the Task-Specific Preconditioner. For a given target task $\mathcal{T}$, the Task-Specific Preconditioner $\mathbf{P}_\mathcal{T}$ is constructed by linearly combining the DSPs $\mathbf{P}_k$ from multiple seen domains, with each weighted by the corresponding task-coefficient $p_{\mathcal{T},k}$. This process produces a preconditioner specifically adapted to the geometric characteristics of the target task's parameter space. By integrating knowledge from multiple seen domains, TSP distinguishes itself from traditional PGD techniques, such as GAP (Kang et al. 2023), which are discussed further in Section 6. Applying our approach to state-of-the-art CDFSL methods, such as TSA or TA^2-Net, significantly enhances performance on Meta-Dataset. For example, in multi-domain settings, applying TSP to TA^2-Net (Guo et al.

2023) achieves the best performance across all datasets.

2 Related Works

Meta-Learning for Few-Shot Learning Until recently, numerous approaches in the field of few-shot learning have adopted the meta-learning framework. These approaches can be mainly divided into three types: metric-based, model-based, and optimization-based methods. Metric-based methods (Sung et al. 2018; Snell, Swersky, and Zemel 2017; Oreshkin, Rodríguez López, and Lacoste 2018) train a feature encoder to extract features from support and query samples. They employ a nearest neighbor classifier with various distance functions to calculate similarity scores for predicting the labels of query samples. Model-based methods (Santoro et al. 2016; Munkhdalai and Yu 2017; Mishra et al. 2017; Garnelo et al. 2018) train an encoder to generate task-specific models from a few support samples. Optimization-based methods (Ravi and Larochelle 2016; Finn, Abbeel, and Levine 2017; Rajeswaran et al. 2019) train a model that can quickly adapt to new tasks with a few support samples, employing a bi-level optimization. In our method, we employ the bi-level optimization used in the optimization-based methods.

Cross-Domain Few-Shot Learning (CDFSL) Recent CDFSL methods define the universal model as a deep neural network and partition it into task-agnostic and task-specific parameters. The task-agnostic parameters represent generic characteristics that are valid for a range of tasks from various domains. On the other hand, the task-specific parameters represent adaptable attributes that are optimized to the target tasks through an adaptation mechanism. Task-agnostic parameters can be designed as a single network or multiple networks. The single network is trained on a large dataset from single domain (Requeima et al. 2019; Bateni et al. 2020, 2022; Liu et al. 2021) or multiple domains (Triantafillou et al. 2021; Li, Liu, and Bilen 2021, 2022; Guo et al. 2023), whereas the multiple networks are trained individually on each domain (Dvornik, Schmid, and Mairal 2020; Liu et al. 2020). Task-specific parameters can be designed as selection parameters (Dvornik, Schmid, and Mairal 2020; Liu et al. 2020), pre-classifier transformation (Li, Liu, and Bilen 2021, 2022; Guo et al. 2023), Feature-wise Linear Modulate (FiLM) layer (Requeima et al. 2019; Bateni et al. 2020, 2022; Liu et al. 2021; Triantafillou et al. 2021), or Residual Adapter (RA) (Li, Liu, and Bilen 2022; Guo et al. 2023). As the adaptation mechanism for the task-specific parameters, several studies (Requeima et al. 2019; Bateni et al. 2020, 2022; Liu et al. 2020, 2021) meta-learn an auxiliary network, which generates task-specific parameters adapted to the target task. On the other hand, other studies (Dvornik, Schmid, and Mairal 2020; Li, Liu, and Bilen 2021; Triantafillou et al. 2021; Li, Liu, and Bilen 2022) employ gradient descent to adapt task-specific parameters to the target task. In our work, we propose a novel adaptation mechanism in the form of a task-specific optimizer, which adapts task-specific parameters to the target task.

Preconditioned Gradient Descent in Meta-Learning In meta-learning, several optimization-based approaches (Li et al. 2017; Lee and Choi 2018; Park and Oliva 2019; Rajasegaran et al. 2020; Simon et al. 2020; Zhao et al. 2020; Von Oswald et al. 2021; Kang et al. 2023) have incorporated Preconditioned Gradient Descent (PGD) to adapt network's parameters to the target task (i.e., inner-level optimization). They meta-learn a preconditioning matrix, called a preconditioner, which is utilized to precondition the gradient. The preconditioner was kept static in most of the previous works (Li et al. 2017; Lee and Choi 2018; Park and Oliva 2019; Zhao et al. 2020; Von Oswald et al. 2021). Several prior studies have devised preconditioners tailored to adapt either per inner step (Rajasegaran et al. 2020), per task (Simon et al. 2020), or both simultaneously (Kang et al. 2023). Motivated by previous works (Amari 1967, 1996, 1998; Kakade 2001; Amari and Douglas 1998), (Kang et al. 2023) recently investigated the constraint of the preconditioner to satisfy the condition for a Riemannian metric (i.e., positive definiteness). They demonstrated that enforcing this constraint on the preconditioner was essential for improving the performance in few-shot learning. In our study, we propose a novel preconditioned gradient descent method with meta-learned task-specific preconditioner that guarantees positive definiteness for improving performance in CDFSL.

3 Backgrounds

Task Formulation for Meta-Learning in CDFSL In CDFSL, task $\mathcal{T}$ is formulated differently compared to traditional few-shot learning. In traditional few-shot learning, tasks are sampled from a single domain, resulting in the same form in both meta-training and meta-testing:

$$\text{meta-training and meta-testing: } \mathcal{T} = \{\mathcal{S}_\mathcal{T}, \mathcal{Q}_\mathcal{T}\} \quad (1)$$

where $\mathcal{S}_\mathcal{T}$ is a support set and $\mathcal{Q}_\mathcal{T}$ is a query set. On the other hand, in CDFSL, tasks are sampled from multiple domains, leading to different forms in meta-training and meta-testing:

$$\begin{aligned} &\text{meta-training: } \mathcal{T} = \{\mathcal{S}_\mathcal{T}\mathcal{Q}_\mathcal{T}, d_\mathcal{T}\}, \\ &\text{meta-testing: } \mathcal{T} = \{\mathcal{S}_\mathcal{T}\mathcal{Q}_\mathcal{T}\}, \end{aligned} \quad (2)$$

where $d_\mathcal{T}$ is a domain label indicating the domain from which the task was sampled. For instance, the domain label is an integer between 1 and K for K domains (i.e., $1 \leq d_\mathcal{T} \leq K$).

Bi-level Optimization in Meta-Learning Bi-level optimization (Rajeswaran et al. 2019) consists of two levels of main optimization processes: inner-level and outer-level optimizations. Let $f_{\theta(\phi)}$ be a model, where the parameter $\theta(\phi)$ is parameterized by the meta-parameter ϕ. For a task $\mathcal{T} = \{\mathcal{S}_\mathcal{T}, \mathcal{Q}_\mathcal{T}\}$, the inner-level optimization is defined as:

$$\theta_{\mathcal{T},T}(\phi) = \theta_{\mathcal{T},0}(\phi) - \alpha_{\text{in}} \cdot \sum_{t=0}^{T-1} \nabla_\theta \mathcal{L}_{\text{in}}(\theta_{\mathcal{T},t}(\phi); \mathcal{S}_\mathcal{T}) \quad (3)$$

where $\theta_{\mathcal{T},0}(\phi) = \theta(\phi)$, α_{in} is the learning rate for the inner-level optimization, $\mathcal{L}_{\text{in}}$ is the inner-level's loss function, and T is the total number of gradient descent steps. With $\mathcal{Q}_\mathcal{T}$ in each task, we can define outer-level optimization as:

$$\phi \leftarrow \phi - \alpha_{\text{out}} \cdot \nabla_\phi \mathbb{E}_\mathcal{T}\Big[\mathcal{L}_{\text{out}}(\theta_{\mathcal{T},T}(\phi); \mathcal{Q}_\mathcal{T})\Big] \quad (4)$$

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

Preconditioned Gradient Descent (PGD) PGD is a technique that minimizes empirical risk by using a gradient update with a preconditioner that re-scales the geometry of the parameter space. Given model parameters θ and task $\mathcal{T} = \{\mathcal{S}_\mathcal{T}, \mathcal{Q}_\mathcal{T}\}$, we can formally define the preconditioned gradient descent with a preconditioner $\mathbf{P}$ as follows:

$$\theta_{\mathcal{T},t} = \theta_{\mathcal{T},t-1} - \alpha \cdot \mathbf{P} \nabla_\theta \mathcal{L}(\theta_{\mathcal{T},t-1}; \mathcal{S}_\mathcal{T}), \quad t = 1, \cdots \quad (5)$$

where $\theta_{\mathcal{T},0} = \theta$, $\mathcal{L}(\theta_{\mathcal{T},t}; \mathcal{S}_\mathcal{T})$ is the empirical loss associated with the task $\mathcal{T}$, and $\theta_{\mathcal{T},t}$ is the parameters. When the preconditioner $\mathbf{P}$ is chosen to be the identity matrix $\mathbf{I}$, Eq. (5) becomes the standard Gradient Descent (GD). The choice of $\mathbf{P}$ to leverage second-order information offers several options, including the inverse Fisher information matrix $\mathbf{F}^{-1}$, leading to the Natural Gradient Descent (NGD) (Amari 1998), the inverse Hessian matrix $\mathbf{H}^{-1}$, corresponding to Newton's method (LeCun et al. 2002), and the diagonal matrix estimation with the past gradients, which results in adaptive gradient methods (Duchi, Hazan, and Singer 2011; Kingma and Ba 2014). They often reduce the effect of pathological curvature and speed up the optimization (Amari et al. 2020).

Dataset Classifier In CDFSL, Dataset Classifier (Triantafillou et al. 2021) reads a support set in a few-shot task and predicts from which of the training datasets it was sampled. Formally, let $\mathcal{T} = \{\mathcal{S}_\mathcal{T}, \mathcal{Q}_\mathcal{T}, d_\mathcal{T}\}$ be a train task sampled from K domains. Let g be a dataset classifier that takes the support set $\mathcal{S}_\mathcal{T}$ as input and generates logits as follows:

$$g(\mathcal{S}_\mathcal{T}) = z_\mathcal{T} = (z_{\mathcal{T},1}, \cdots, z_{\mathcal{T},K}) \in \mathbb{R}^K \quad (6)$$

In (Triantafillou et al. 2021), the dataset classifier g is trained to minimize the cross-entropy loss for the dataset classification problem (i.e., classification problem with K classes).

4 Method

In this section, we propose a novel adaptation mechanism named Task-Specific Preconditioned gradient descent (TSP). We first introduce Domain-Specific Preconditioner (DSP) and task-coefficients. Then, we describe the construction of Task-Specific Preconditioner using DSP and task-coefficients. Lastly, we show the positive definiteness of Task-Specific Preconditioner, which establishes it as a valid Riemannian metric. The algorithm for the training and testing procedures is provided in Appendix B.

4.1 Domain-Specific Preconditioner (DSP)

Consider L task-specific parameters $\theta = \{\theta^l \in \mathbb{R}^{m_l \times m_l}\}_{l=1}^L$. For K domains, we first define meta-parameters $\mathcal{M}_1, \cdots, \mathcal{M}_K$ as follows:

$$\mathcal{M}_k = \{\mathbf{M}_k^l \in \mathbb{R}^{m_l \times m_l}\}_{l=1}^L, \quad k = 1, \cdots, K \quad (7)$$

Then, for all l, we define Domain-Specific Preconditioners (DSPs) $\mathbf{P}_k^l$ using the meta-parameters as follows:

$$\mathbf{P}_k^l = \mathbf{M}_k^{l\mathbf{T}} \mathbf{M}_k^l + \mathbf{I}, \quad k = 1, \cdots, K \quad (8)$$

We compare various DSP designs (See Table 3) in Section 5.3 and choose the form of Eq. (8). Through bi-level optimization, DSPs can be meta-learned as follows.

Inner-level Optimization For each train task $\mathcal{T} = \{\mathcal{S}_\mathcal{T}, \mathcal{Q}_\mathcal{T}, d_\mathcal{T}\}$, in the inner-level optimization, we optimize the task-specific parameters θ through preconditioned gradient descent using $\mathbf{P}_{d_\mathcal{T}}^l$, updating θ as follows:

$$\theta_{\mathcal{T},T}^l = \theta_{\mathcal{T},0}^l - \alpha_{\text{in}} \cdot \sum_{t=0}^{T-1} \mathbf{P}_{d_\mathcal{T}}^l \nabla_{\theta_{\mathcal{T},t}^l} \mathcal{L}_{\text{in}}(\theta_{\mathcal{T},t}; \mathcal{S}_\mathcal{T}), \quad (9)$$

where $\theta_{\mathcal{T},0}^l = \theta^l$, α_{in} is the learning rate for the inner-level optimization, T is the total number of gradient descent steps, and $\mathcal{L}_{\text{in}}$ is the inner-level's loss function.

Outer-level Optimization In the outer-level optimization, we meta-learn meta-parameters $\mathcal{M}_1, \cdots \mathcal{M}_K$ as follows:

$$\mathcal{M}_k \leftarrow \mathcal{M}_k - \alpha_{\text{out}} \cdot \nabla_{\mathcal{M}_k} \mathbb{E}_\mathcal{T}\Big[\mathcal{L}_{\text{out}}(\theta_{\mathcal{T},T}; \mathcal{Q}_\mathcal{T})\Big], \quad k = 1, \cdots, K \quad (10)$$

where α_{out} is the learning rate for outer-level optimization and $\mathcal{L}_{\text{out}}$ is the outer-level's loss function.

4.2 Task-coefficients

Consider the dataset classifier g. Given a train task $\mathcal{T} = \{\mathcal{S}_\mathcal{T}, \mathcal{Q}_\mathcal{T}, d_\mathcal{T}\}$, we define task-coefficients $p_{\mathcal{T},1} \cdots, p_{\mathcal{T},K}$ as follows:

$$(p_{\mathcal{T},1}, \cdots, p_{\mathcal{T},K}) = \text{Softmax}(z_{\mathcal{T},1}, \cdots, z_{\mathcal{T},K}) \quad (11)$$

where $g(\mathcal{S}_\mathcal{T}) = (z_{\mathcal{T},1}, \cdots, z_{\mathcal{T},K})$. Note that we use the sigmoid function instead of softmax in the single-domain setting because the output dimension of the dataset classifier is one. While Triantafillou et al. (2021) updates the parameters of g to minimize only the cross-entropy loss $\mathcal{L}_{\text{CE}}$ with respect to the dataset label $d_\mathcal{T}$, we train the dataset classifier g to minimize the following augmented loss:

$$\mathcal{L}_{\text{CE}} + \lambda \cdot \mathcal{L}_{\text{Aux}} \quad (12)$$

where λ is a regularization parameter and $\mathcal{L}_{\text{Aux}}$ is the auxiliary loss, defined as follows:

$$\mathcal{L}_{\text{Aux}} = \mathbb{E}_\mathcal{T}\Big[\mathcal{L}_{\text{out}}(\theta_{\mathcal{T},T}; \mathcal{Q}_\mathcal{T})\Big] \quad (13)$$

Here, task-specific parameters $\theta_{\mathcal{T},T}^l$ can be obtained as follows:

$$\theta_{\mathcal{T},T}^l = \theta_{\mathcal{T},0}^l - \alpha_{\text{in}} \cdot \sum_{t=0}^{T-1} \sum_{k=1}^{K} p_{\mathcal{T},k} \cdot \mathbf{P}_k^l \nabla_{\theta_{\mathcal{T},t}^l} \mathcal{L}_{\text{in}}(\theta_{\mathcal{T},t}; \mathcal{S}_\mathcal{T}) \quad (14)$$

where $\mathbf{P}_k^l$ is the l-th DSP of domain k. In Eq. (12), the cross-entropy loss guides the dataset classifier to prioritize the ground-truth domain of the support set. Concurrently, the auxiliary loss guides toward DSPs that minimize any adverse effects on the performance of the query set during the inner-level optimization.

4.3 Task-Specific Preconditioner

Given a test task $\mathcal{T} = \{\mathcal{S}_\mathcal{T}, \mathcal{Q}_\mathcal{T}\}$, we define Task-Specific Preconditioner $\mathbf{P}_\mathcal{T}^l$ as follows:

$$\mathbf{P}_\mathcal{T}^l = \sum_{k=1}^{K} p_{\mathcal{T},k} \cdot \mathbf{P}_k^l, \quad l = 1, \cdots, L \quad (15)$$

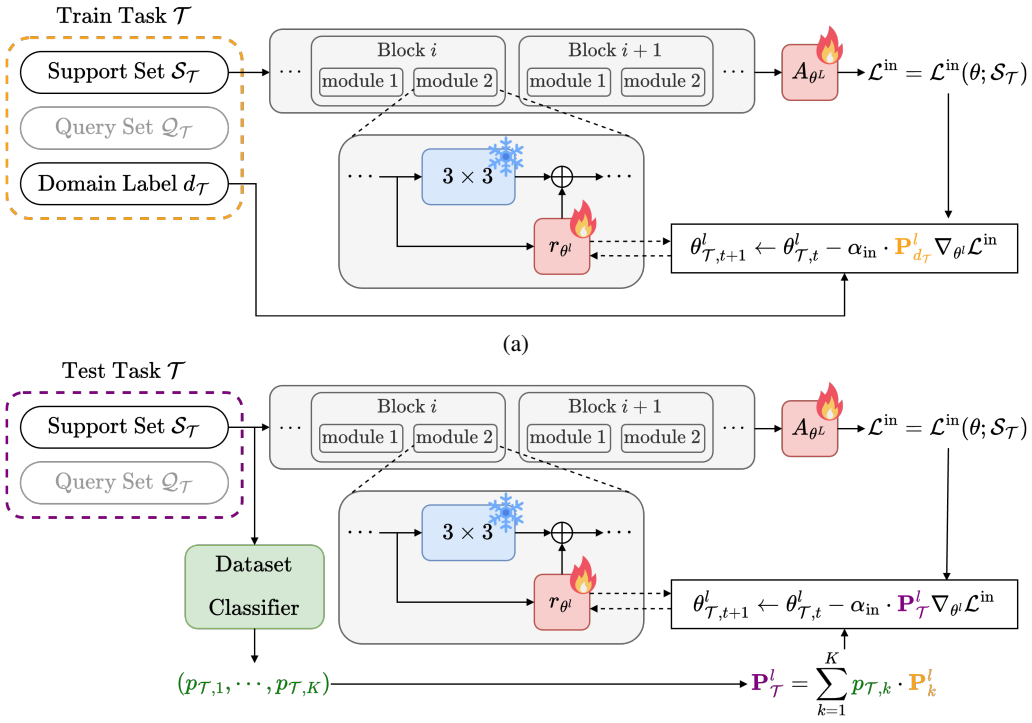


Figure 3: (a) PGD with Domain-Specific Preconditioner (DSP) in the inner-level optimization. During meta-training, for a train task $\mathcal{T}$, DSP is chosen based on the domain label $d_\mathcal{T}$, and each task-specific parameter θ^l are optimized using PGD with the selected DSP $\mathbf{P}^l_{d_\mathcal{T}}$. (b) PGD with Task-Specific Preconditioner. During meta-testing, for a test task, each Task-Specific Preconditioner $\mathbf{P}^l_\mathcal{T}$ is contructed using DSPs and task-coefficients generated by Dataset Classifier. Each task-specific parameter θ^l is then then optimized using PGD with $\mathbf{P}^l_\mathcal{T}$.

where $\mathbf{P}^l_k$ is the l-th DSP of domain k, and $p_{\mathcal{T},k}$ is the task-coefficient for the given task $\mathcal{T}$ and domain k. By employing $\mathbf{P}^l_\mathcal{T}$ as the preconditioning matrix, we can define Task-Specific Preconditioned gradient descent (TSP), as follows:

$$\theta^l_{\mathcal{T},T} = \theta^l_{\mathcal{T},0} - \beta \cdot \sum_{t=0}^{T-1} \mathbf{P}^l_\mathcal{T} \nabla_{\theta^l_{\mathcal{T},t}} \mathcal{L}_{\text{in}}(\theta_{\mathcal{T},t}; \mathcal{S}_\mathcal{T}), \quad (16)$$

where β is the learning rate used to adapt the task-specific parameters.

4.4 Positive Definiteness of TSP's Preconditioner

A preconditioner satisfying positive definiteness ensures a valid Riemannian metric, which represents the geometric characteristics of the parameter space (Amari 1967, 1996, 1998; Kakade 2001; Amari and Douglas 1998). Task-Specific Preconditioner $\mathbf{P}^l_\mathcal{T}$ is designed to be a positive definite matrix, which is verified in Theorem 1.

Theorem 1. *Let $p_k \in [0,1], k = 1, \cdots, K$, be the task-coefficients satisfying $\sum_{k=1}^{K} p_k = 1$. For the Domain-Specific Preconditioners $\mathbf{P}_k \in \mathbb{R}^{m \times m}, k = 1, \cdots, K$, Task-Specific Preconditioner $\mathbf{P}$ defined as $\mathbf{P} = \sum_{k=1}^{K} p_k \cdot \mathbf{P}_k$ is positive definite.*

The proof is provided in Appendix C. Drawing from prior research (Amari 1967, 1996, 1998; Kakade 2001; Amari and Douglas 1998), a preconditioner satisfying positive definiteness promotes gradients to point toward the steepest descent direction while avoiding undesirable paths in the parameter space. As shown in Figure 1b, positive definiteness improves CDFSL performance, especially in unseen domains. In Section 6, we will discuss why this property helps in CDFSL.

5 Experiments

5.1 Experimental Setup

Implementation Details In the experiments, we use Meta-Dataset (Triantafillou et al. 2019) that is the standard benchmark for evaluating the performance of CDFSL. To demonstrate the effectiveness of TSP as an adaptation mechanism, we apply it to the state-of-the-art CDFSL methods, TSA (Li, Liu, and Bilen 2022) and TA^2-Net (Guo et al. 2023), which are publicly available as open-source. Following previous studies (Bateni et al. 2022; Triantafillou et al. 2021; Li, Liu, and Bilen 2021, 2022; Guo et al. 2023), we adopted ResNet-18 as the backbone for the feature extractor. In all experiments, we follow the standard protocol described in (Triantafillou et al. 2019). For the Dataset Classifier Loss, weighting factor λ is set to 0.1, as it performs best compared to other values, as shown in Appendix D.1. Details of the Meta-Dataset, hyper-parameters, and additional implementation are available in Appendix E.

Test Dataset	SUR	URT	FLUTE	tri-M	URL	TSA	TA^2-Net	MOKD	TSP†	TSP††
ImageNet	56.2±1.0	56.8±1.1	58.6±1.0	51.8±1.1	58.8±1.1	59.5±1.0	59.6±1.0	57.3±1.1	60.5±1.0	**60.7±1.0**
Omniglot	94.1±0.4	94.2±0.4	92.0±0.6	93.2±0.5	94.5±0.4	94.9±0.4	95.5±0.4	94.2±0.5	95.6±0.4	**96.0±0.4**
Aircraft	85.5±0.5	85.8±0.5	82.8±0.7	87.2±0.5	89.4±0.4	89.9±0.4	90.5±0.4	88.4±0.5	90.5±0.4	**91.2±0.4**
Birds	71.0±1.0	76.2±0.8	75.3±0.8	79.2±0.8	80.7±0.8	81.1±0.8	81.4±0.8	80.4±0.8	82.3±0.7	**82.5±0.7**
Textures	71.0±0.8	71.6±0.7	71.2±0.8	68.8±0.8	77.2±0.7	77.5±0.7	77.4±0.7	76.5±0.7	78.6±0.6	**79.1±0.6**
Quick Draw	81.8±0.6	82.4±0.6	77.3±0.7	79.5±0.7	82.5±0.6	81.7±0.6	82.5±0.6	82.2±0.6	83.0±0.7	**83.2±0.6**
Fungi	64.3±0.9	64.0±1.0	48.5±1.0	58.1±1.1	68.1±0.9	66.3±0.8	66.3±0.9	68.6±1.0	68.6±0.9	**69.7±0.8**
VGG Flower	82.9±0.8	87.9±0.6	90.5±0.5	91.6±0.6	92.0±0.5	92.2±0.5	92.6±0.4	92.5±0.5	93.3±0.4	**93.4±0.4**
Traffic Sign	51.0±1.1	48.2±1.1	63.0±1.0	58.4±1.1	63.3±1.1	82.8±1.0	87.4±0.8	64.5±1.1	88.5±0.7	**89.4±0.8**
MSCOCO	52.0±1.1	51.5±1.1	52.8±1.1	50.0±1.0	57.3±1.0	57.6±1.0	57.9±0.9	55.5±1.0	58.5±0.9	**59.8±0.9**
MNIST	94.3±0.4	90.6±0.5	96.2±0.3	95.6±0.5	94.7±0.4	96.7±0.4	97.0±0.4	95.1±0.4	**97.1±0.3**	**97.1±0.4**
CIFAR-10	66.5±0.9	67.0±0.8	75.4±0.8	78.6±0.7	74.2±0.8	82.9±0.7	82.1±0.8	72.8±0.8	83.5±0.7	**83.7±0.8**
CIFAR-100	56.9±1.1	57.3±1.0	62.0±1.0	67.1±1.0	63.5±1.0	70.4±0.9	70.9±0.9	63.9±1.0	71.3±1.0	**72.2±0.9**
Avg Seen	75.9	77.4	74.5	76.2	80.4	80.4	80.7	80.0	81.6	**82.0**
Avg Unseen	64.1	62.9	69.9	69.9	70.6	78.1	79.1	70.3	79.8	**80.4**
Avg All	71.3	71.8	72.7	73.8	76.6	79.5	80.1	76.3	80.9	**81.4**
Avg Rank	8.8	8.2	8.0	7.8	5.5	4.3	3.2	5.8	1.9	**1.0**

Table 1: Performance comparison to state-of-the-art methods in a multi-domain setting. Mean accuracy and 95% confidence interval are reported. The best results are highlighted in **bold**. TSP† denotes TSP applied on TSA. TSP†† denotes TSP applied on TA^2-Net.

Baselines For the baselines, we compare our methods to the state-of-the-art CDFSL methods, including BOHB (Saikia, Brox, and Schmid 2020), SUR (Dvornik, Schmid, and Mairal 2020), URT (Liu et al. 2020), Simple-CNAPS (Bateni et al. 2020), FLUTE (Triantafillou et al. 2021), tri-M (Liu et al. 2021), URL (Li, Liu, and Bilen 2021), TSA (Li, Liu, and Bilen 2022), TA^2-Net (Guo et al. 2023), ALFA (Baik et al. 2023)+Proto-MAML, GAP+Proto-MAML (Kang et al. 2023), and MOKD (Tian et al. 2024).

5.2 Performance Comparison to State-of-The-Art Methods

Following the experimental setup in (Li, Liu, and Bilen 2022), we first evaluate our method using multi-domain and single-domain feature extractors in Varying-Way Varying-Shot setting (i.e., Multi-domain and Single-domain setting). Then, we assess our approach with the multi-domain feature extractor in more challenging Varying-Way Five-Shot and Five-Way One-Shot settings. We provide the performance comparison results for Varying-Way Five-Shot and Five-Way One-Shot settings in the Appendix F.

Multi-Domain Setting In Table 1, we evaluate TSP by applying it to TSA and TA^2-Net, both of which employ URL (Liu et al. 2021) as the multi-domain feature extractor. We report average accuracies over seen, unseen, and all domains, along with average rank following the previous works (Liu et al. 2021; Li, Liu, and Bilen 2022; Guo et al. 2023). TSP† denotes TSP applied on TSA, while TSP†† indicates TSP applied on TA^2-Net. TSP† outperforms the previous state-of-the-art methods on 11 out of 13 datasets, and TSP†† achieves the best results on all datasets. For example, TSP†† outperforms the state-of-the-art method (TA^2-Net) by 1.7%, 3.4%, 2.0%, and 1.9% on Textures, Fungi, Traffic Sign, and MSCOCO respectively. These results imply that TSP can construct a desirable task-specific optimizer that effectively adapt the task-specific parameters for a given target task.

Single-Domain Setting We evaluate TSP by applying it to TSA and TA^2-Net, both of which employ the single-domain feature extractor pretrained solely on the ImageNet dataset. In Table 2, TSP†† achieves the best results for 12 out of 13 datasets, while TSP† leads in the remaining 1 datasets. Compared to recently proposed meta-learning methods based on PGD, such as Approximate GAP+Proto-MAML and GAP+Proto-MAML (Kang et al. 2023), both TSP† and TSP†† consistently outperform them across all 13 datasets by a significant margin. Furthermore, TSP†† outperforms the previous best methods by a clear margin in several datasets such as Quick Draw ($+3.1\%$), Omniglot ($+4.1\%$), and Traffic Sign ($+4.6\%$). Despite being trained only on single dataset, TSP improves performance by effectively constructing a task-specific optimizer tailored to the target task.

5.3 Ablation Studies

In this section, all ablation studies are performed using TSP† in the multi-domain setting to isolate the effects originating from the RL model in TSP††. Additional ablation studies are provided in Appendix D.

Matrix Design for DSP To design Domain-Specific Preconditioner (DSP), we consider three matrix designs that guarantee positive definiteness. The first one is the product of a real-valued lower triangular matrix and its transpose (i.e., $\mathbf{L}\mathbf{L}^\mathbf{T}$), where the lower triangular matrix $\mathbf{L}$ is constrained to have positive diagonals. This form is commonly known as the Cholesky factorization (Horn and Johnson 2012). The second one is the addition of $\mathbf{L}\mathbf{L}^\mathbf{T}$ and the identity matrix (i.e., $\mathbf{L}\mathbf{L}^\mathbf{T} + \mathbf{I}$). The last one is the addition of the Gram matrix (Horn and Johnson 2012) and the identity matrix (i.e., $\mathbf{M}^\mathbf{T}\mathbf{M} + \mathbf{I}$). In Table 3, we compare three TSPs with these three DSP designs. Among them, the

Test Dataset	ALFA+ Proto-MAML	BOHB	GAP+ Proto-MAML	FLUTE	TSA	TA^2-Net	MOKD	TSP†	TSP††
ImageNet	52.8±1.1	51.9±1.1	56.7	46.9±1.1	59.5±1.1	59.3±1.1	57.3±1.1	60.1±1.1	**60.6±1.1**
Omniglot	61.9±1.5	67.6±1.2	77.6	61.6±1.4	78.2±1.2	81.1±1.1	70.9±1.3	83.3±1.1	**85.2±1.1**
Aircraft	63.4±1.1	54.1±0.9	68.5	48.5±1.0	72.2±1.0	72.6±0.9	59.8±1.0	73.2±1.0	**73.5±1.1**
Birds	69.8±1.1	70.7±0.9	73.5	47.9±1.0	74.9±0.9	75.1±0.9	73.6±0.9	76.0±0.9	**76.6±0.9**
Textures	70.8±0.9	68.3±0.8	71.4	63.8±0.8	77.3±0.7	76.8±0.8	76.1±0.7	78.2±0.7	**78.3±0.7**
Quick Draw	59.2±1.2	50.3±1.0	65.4	57.5±1.0	67.6±0.9	68.4±0.9	61.2±1.0	70.8±0.9	**71.5±0.9**
Fungi	41.5±1.2	41.4±1.1	38.6	31.8±1.0	44.7±1.0	45.3±1.0	**47.0±1.1**	46.6±1.0	**47.0±1.0**
VGG Flower	86.0±0.8	87.3±0.6	86.8	80.1±0.9	90.9±0.6	91.0±0.6	88.5±0.6	91.8±0.5	**92.2±0.6**
Traffic Sign	60.8±1.3	51.8±1.0	66.9	46.5±1.1	82.5±0.8	84.1±0.7	61.6±1.1	87.5±0.8	**88.7±0.8**
MSCOCO	48.1±1.1	48.0±1.0	46.8	41.4±1.0	59.0±1.0	58.0±1.0	55.3±1.0	**59.4±1.0**	58.6±1.0
MNIST	-	-	94.0	80.8±0.8	93.9±0.6	94.9±0.5	88.3±0.7	94.5±0.5	**95.3±0.6**
CIFAR-10	-	-	74.5	65.4±0.8	82.1±0.7	82.0±0.7	72.2±0.8	83.1±0.5	**83.2±0.7**
CIFAR-100	-	-	63.2	52.7±1.1	70.7±0.9	70.8±0.9	63.1±1.0	71.2±0.9	**72.8±0.9**
Avg Seen	52.8	51.9	56.7	46.9	59.5	59.3	57.3	60.1	**60.6**
Avg Unseen	62.4	59.9	68.9	56.5	74.5	75.0	68.1	76.3	**76.9**
Avg All	61.4	59.1	68.0	55.8	73.3	73.8	67.3	75.0	**75.7**
Avg Rank	7.0	7.5	6.1	8.9	3.7	3.4	5.1	2.0	**1.2**

Table 2: Performance comparison to state-of-the-art methods in a single-domain setting. Mean accuracy and 95% confidence interval are reported. The best results are highlighted in **bold**. TSP† denotes TSP applied on TSA. TSP†† denotes TSP applied on TA^2-Net.

DSP designs	$\mathbf{L}\mathbf{L}^\mathbf{T}$	$\mathbf{L}\mathbf{L}^\mathbf{T}+\mathbf{I}$	$\mathbf{M}^\mathbf{T}\mathbf{M}+\mathbf{I}$
Avg Seen	80.8	81.2	**81.6**
Avg Unseen	79.0	79.4	**79.8**
Avg All	80.1	80.5	**80.9**

Table 3: Performance comparison of three TSPs with different DSP designs.

Preconditioner	w/ PD constraint	w/o PD constraint
Avg Seen	**81.6**	80.0
Avg Unseen	**79.8**	73.8
Avg All	**80.9**	77.6

Table 4: Performance comparison of TSPs with and without a PD constraint. α is set to 0.1.

Gram matrix design achieves the highest average accuracies in both seen and unseen domains compared to the others. Therefore, we choose the Gram matrix design for DSP.

6 Discussion

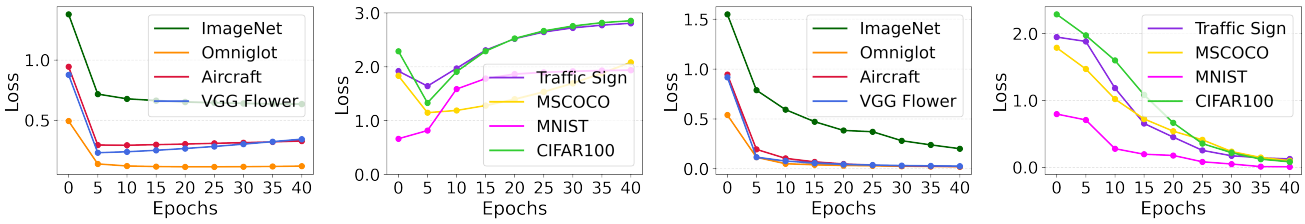
In this section, all experiments are conducted using TSP†.

The Necessity of Positive Definite Constraint Even without a specific constraint of PD, one might assume that initializing the preconditioner as positive definite, such as $\alpha \cdot \mathbf{I}$, would maintain its positive definiteness throughout meta-training due to its significant role. However, as illustrated in Table 4 and Table 5, this assumption does not hold. In Table 4, we compare preconditioners with and without a PD constraint, both initialized as positive definite. Specifically, the former adopts Task-Specific Preconditioner (See Eq. (15)), while the latter employs Task-Specific Preconditioner with DSP designed as $\mathbf{P}_k^l = \mathbf{M}_k^l$ and initialized as $\mathbf{M}_k^l = 0.1 \cdot \mathbf{I}$. Evaluations are conducted using the multi-domain feature extractor (URL) in the multi-domain setting. After meta-training, DSPs without a PD constraint tend to lose positive definiteness as shown in Table 5, leading to poor performance as shown in Table 4. These findings underscore the necessity of explicitly constraining the preconditioner to maintain positive definiteness, as relying solely on optimization fails to preserve this crucial property.

Positive Definite DSP Designs with and without the Identity Matrix Apart from ensuring positive definiteness, a notable characteristic of our Gram matrix design $\mathbf{M}^\mathbf{T}\mathbf{M}+\mathbf{I}$ is its inclusion of the identity matrix. To explore the impact of this inclusion, we compare two positive definite DSP designs: $\mathbf{L}\mathbf{L}^\mathbf{T}$ and $\mathbf{L}\mathbf{L}^\mathbf{T}+\mathbf{I}$. We focus on these two DSP designs because $\mathbf{M}^\mathbf{T}\mathbf{M}$ does not guarantee positive definiteness. However, we also provide a comparison between $\mathbf{M}^\mathbf{T}\mathbf{M}+\mathbf{I}$ and $\mathbf{M}^\mathbf{T}\mathbf{M}$ in Appendix G. The experiments are conducted using the multi-domain feature extractor (URL). In Table 6, we observe that the DSP design with the added identity matrix performs better in the Varying-Way Varying-Shot setting but worse in the Varying-Way Five-Shot setting. This outcome aligns with prior theoretical findings (Amari et al. 2020) indicating that PGD performs better than GD in noisy gradient conditions, while GD excels when gradients are accurate. With more shots, gradients tend to be more accurate due to increased data. In the Varying-Way Varying Shot setting, where tasks typically involve more than five shots, gradients are more accurate, making GD more beneficial compared to the other setting. Including the identity matrix can be viewed as a regularization of PGD towards GD. Consequently, $\mathbf{L}\mathbf{L}^\mathbf{T}+\mathbf{I}$ aligns closer to GD compared to $\mathbf{L}\mathbf{L}^\mathbf{T}$, resulting in improved performance due to the abundance of shots in Varying-Way Varying-Shot setting. Conversely, in the Varying-Way Five-Shot setting, where tasks involve fewer shots, $\mathbf{L}\mathbf{L}^\mathbf{T}$ exhibits superior performance to $\mathbf{L}\mathbf{L}^\mathbf{T}+\mathbf{I}$ due to the scarcity of shots.

DSP	ImageNet	Omniglot	Aircraft	Birds	Textures	Quick Draw	Fungi	VGG Flower	Average
Non-PD rate	0.24	0.29	0.35	0.24	0.35	0.35	0.18	0.29	0.29

Table 5: The rates of non-PD Domain-Specific Preconditioners (DSPs) after meta-training without a positive definite constraint. For the ResNet-18 backbone, there are 17 DSP preconditioners for each domain. All DSPs are initialized as $0.1 \cdot \mathbf{I}$. The average rate is provided in the right column.



(a) w/o PD, Seen (b) w/o PD, Unseen (c) w/ PD, Seen (d) w/ PD, Unseen

Figure 4: Learning curves of PGD with and without the PD constraint across both seen and unseen domains. Further details on the preconditioners used in this figure can be found in Appendix A.

Setting	Varying-Way Varying-Shot		Varying-Way Five-Shot	
DSP designs	$\mathbf{LL^T}$	$\mathbf{LL^T + I}$	$\mathbf{LL^T}$	$\mathbf{LL^T + I}$
Avg Seen	80.8	**81.2**	**76.8**	76.6
Avg Unseen	79.0	**79.4**	**72.1**	71.5
Avg All	80.1	**80.5**	**75.0**	74.6

Table 6: Performance comparison of two positive definite DSP designs with and without adding an identity matrix.

Effectiveness of Positive Definiteness in Cross-Domain Tasks A positive definite preconditioner is known to mitigate the negative effects of pathological loss curvature and accelerate optimization, thereby facilitating convergence (Nocedal and Wright 1999; Saad 2003; Li 2017). This leads to a consistent reduction in the objective function. However, without positive definiteness, this effect is not guaranteed and may result in failure to converge. In Figure 4, we compare the learning curves of PGD with and without a PD constraint across both seen and unseen domains. Without the PD constraint, PGD fails to converge in some of the seen domains and in all the unseen domains. With the PD constraint, PGD successfully converges in all the seen and unseen domains. These results suggest that, in cross-domain tasks, a PD constraint of a preconditioner is crucial for achieving convergence and is beneficial for improving performance, which is also related to Figure 1b.

TSP vs. Previous PGD Methods: Leveraging Multi-Domain Knowledge for Task-Specific Preconditioner Compared to previous PGD methods like GAP (Kang et al. 2023), TSP is specifically designed for cross-domain few-shot learning (CDFSL), where unseen domains are not accessed during meta-training. The key challenge in CDFSL is to effectively leverage information from multiple seen domains to quickly adapt to each unseen domain. Previous PGD methods fall short in this regard because they rely on a single preconditioner, even when multiple seen domains are available. For example, GAP uses only one preconditioner to extract information from multiple seen domains, which limits its adaptability to unseen domains with distinct characteristics. In contrast, TSP meta-trains a distinct domain-specific preconditioner (DSP) for each seen domain and combines them to construct a Task-Specific Preconditioner that better suited to each unseen domain. TSP produces this Task-Specific Preconditioner *effectively*, as shown in Tables 1 and 2, and *time-efficiently*, as further detailed in Appendix H.

7 Conclusion

In this study, we have introduced a robust and effective adaptation mechanism called Task-Specific Preconditioned gradient descent (TSP) to enhance CDFSL performance. Thanks to the meta-trained Domain-Specific Preconditioners (DSPs) and Task-coefficients, TSP can flexibly adjust the optimization strategy according to the geometric characteristics of the parameter space for the target task. Owing to these components, the proposed TSP demonstrates notable performance improvements on Meta-Dataset across various settings.

Acknowledgements

This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2020R1A2C2007139) and in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) ([NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], [No. RS-2023-00235293, Development of autonomous driving big data processing, management, search, and sharing interface technology to provide autonomous driving data according to the purpose of usage]).

References

Amari, S. 1967. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, (3): 299–307.

Amari, S.-i. 1996. Neural learning in structured parameter spaces-natural Riemannian gradient. *Advances in neural information processing systems*, 9.

Amari, S.-I. 1998. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276.

Amari, S.-i.; Ba, J.; Grosse, R.; Li, X.; Nitanda, A.; Suzuki, T.; Wu, D.; and Xu, J. 2020. When does preconditioning help or hurt generalization? *arXiv preprint arXiv:2006.10732*.

Amari, S.-I.; and Douglas, S. C. 1998. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 2, 1213–1216. IEEE.

Baik, S.; Choi, M.; Choi, J.; Kim, H.; and Lee, K. M. 2023. Learning to learn task-adaptive hyperparameters for few-shot learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bateni, P.; Barber, J.; Van de Meent, J.-W.; and Wood, F. 2022. Enhancing few-shot image classification with unlabelled examples. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2796–2805.

Bateni, P.; Goyal, R.; Masrani, V.; Wood, F.; and Sigal, L. 2020. Improved few-shot visual classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14493–14502.

Chen, W.-Y.; Liu, Y.-C.; Kira, Z.; Wang, Y.-C. F.; and Huang, J.-B. 2019. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Dvornik, N.; Schmid, C.; and Mairal, J. 2020. Selecting relevant features from a multi-domain representation for few-shot classification. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, 769–786. Springer.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Garnelo, M.; Rosenbaum, D.; Maddison, C.; Ramalho, T.; Saxton, D.; Shanahan, M.; Teh, Y. W.; Rezende, D.; and Eslami, S. A. 2018. Conditional neural processes. In *International conference on machine learning*, 1704–1713. PMLR.

Guo, Y.; Du, R.; Dong, Y.; Hospedales, T.; Song, Y.-Z.; and Ma, Z. 2023. Task-aware Adaptive Learning for Cross-domain Few-shot Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1590–1599.

Himmelblau, D. M.; et al. 2018. *Applied nonlinear programming*. McGraw-Hill.

Horn, R. A.; and Johnson, C. R. 2012. *Matrix analysis*. Cambridge university press.

Kakade, S. M. 2001. A natural policy gradient. *Advances in neural information processing systems*, 14.

Kang, S.; Hwang, D.; Eo, M.; Kim, T.; and Rhee, W. 2023. Meta-Learning with a Geometry-Adaptive Preconditioner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16080–16090.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lake, B.; Salakhutdinov, R.; Gross, J.; and Tenenbaum, J. 2011. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33.

LeCun, Y.; Bottou, L.; Orr, G. B.; and Müller, K.-R. 2002. Efficient backprop. In *Neural networks: Tricks of the trade*, 9–50. Springer.

Lee, Y.; and Choi, S. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, 2927–2936. PMLR.

Li, W.-H.; Liu, X.; and Bilen, H. 2021. Universal representation learning from multiple domains for few-shot classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9526–9535.

Li, W.-H.; Liu, X.; and Bilen, H. 2022. Cross-domain few-shot learning with task-specific adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7161–7170.

Li, X.-L. 2017. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5): 1454–1466.

Li, Z.; Zhou, F.; Chen, F.; and Li, H. 2017. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Liu, L.; Hamilton, W.; Long, G.; Jiang, J.; and Larochelle, H. 2020. A universal representation transformer layer for few-shot image classification. *arXiv preprint arXiv:2006.11702*.

Liu, Y.; Lee, J.; Zhu, L.; Chen, L.; Shi, H.; and Yang, Y. 2021. A multi-mode modulator for multi-domain few-shot classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8453–8462.

Mishra, N.; Rohaninejad, M.; Chen, X.; and Abbeel, P. 2017. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*.

Munkhdalai, T.; and Yu, H. 2017. Meta networks. In *International conference on machine learning*, 2554–2563. PMLR.

Nocedal, J.; and Wright, S. J. 1999. *Numerical optimization*. Springer.

Oreshkin, B.; Rodríguez López, P.; and Lacoste, A. 2018. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31.

Park, E.; and Oliva, J. B. 2019. Meta-curvature. *Advances in Neural Information Processing Systems*, 32.

Rajasegaran, J.; Khan, S.; Hayat, M.; Khan, F. S.; and Shah, M. 2020. Meta-learning the learning trends shared across tasks. *arXiv preprint arXiv:2010.09291*.

Rajeswaran, A.; Finn, C.; Kakade, S. M.; and Levine, S. 2019. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32.

Ravi, S.; and Larochelle, H. 2016. Optimization as a model for few-shot learning. In *International conference on learning representations*.

Requeima, J.; Gordon, J.; Bronskill, J.; Nowozin, S.; and Turner, R. E. 2019. Fast and flexible multi-task classification using conditional neural adaptive processes. *Advances in Neural Information Processing Systems*, 32.

Saad, Y. 2003. *Iterative methods for sparse linear systems*. SIAM.

Saikia, T.; Brox, T.; and Schmid, C. 2020. Optimized generic feature learning for few-shot classification across domains. *arXiv preprint arXiv:2001.07926*.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, 1842–1850. PMLR.

Simon, C.; Koniusz, P.; Nock, R.; and Harandi, M. 2020. On modulating the gradient for meta-learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, 556–572. Springer.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P. H.; and Hospedales, T. M. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1199–1208.

Tian, H.; Liu, F.; Liu, T.; Du, B.; Cheung, Y.-m.; and Han, B. 2024. MOKD: Cross-domain Finetuning for Few-shot Classification via Maximizing Optimized Kernel Dependence. *arXiv preprint arXiv:2405.18786*.

Tian, Y.; Wang, Y.; Krishnan, D.; Tenenbaum, J. B.; and Isola, P. 2020. Rethinking few-shot image classification: a good embedding is all you need? In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, 266–282. Springer.

Triantafillou, E.; Larochelle, H.; Zemel, R.; and Dumoulin, V. 2021. Learning a universal template for few-shot dataset generalization. In *International Conference on Machine Learning*, 10424–10433. PMLR.

Triantafillou, E.; Zhu, T.; Dumoulin, V.; Lamblin, P.; Evci, U.; Xu, K.; Goroshin, R.; Gelada, C.; Swersky, K.; Manzagol, P.-A.; et al. 2019. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*.

Von Oswald, J.; Zhao, D.; Kobayashi, S.; Schug, S.; Caccia, M.; Zucchet, N.; and Sacramento, J. 2021. Learning where to learn: Gradient sparsity in meta and continual learning. *Advances in Neural Information Processing Systems*, 34: 5250–5263.

Zhao, D.; Kobayashi, S.; Sacramento, J.; and von Oswald, J. 2020. Meta-learning via hypernetworks. In *4th Workshop on Meta-Learning at NeurIPS 2020 (MetaLearn 2020)*. NeurIPS.