Meta-Learning with Self-Improving Momentum Target

Jihoon Tack[1], **Jongjin Park**[1], **Hankook Lee**[1], **Jaeho Lee**[2], **Jinwoo Shin**[1]
[1]Korea Advanced Institute of Science and Technology (KAIST)
[2]Pohang University of Science and Technology (POSTECH)
{jihoontack,jongjin.park,hankook.lee,jinwoos}@kaist.ac.kr
jaeho.lee@postech.ac.kr

Abstract

The idea of using a separately trained target model (or *teacher*) to improve the performance of the student model has been increasingly popular in various machine learning domains, and meta-learning is no exception; a recent discovery shows that utilizing task-wise target models can significantly boost the generalization performance. However, obtaining a target model for each task can be highly expensive, especially when the number of tasks for meta-learning is large. To tackle this issue, we propose a simple yet effective method, coined *Self-improving Momentum Target (SiMT)*. SiMT generates the target model by adapting from the temporal ensemble of the meta-learner, i.e., the momentum network. This momentum network and its task-specific adaptations enjoy a favorable generalization performance, enabling *self-improving* of the meta-learner through knowledge distillation. Moreover, we found that perturbing parameters of the meta-learner, e.g., dropout, further stabilize this self-improving process by preventing fast convergence of the distillation loss during meta-training. Our experimental results demonstrate that SiMT brings a significant performance gain when combined with a wide range of meta-learning methods under various applications, including few-shot regression, few-shot classification, and meta-reinforcement learning. Code is available at https://github.com/jihoontack/SiMT.

1 Introduction

Meta-learning [51] is the art of extracting and utilizing the knowledge from the distribution of tasks to better solve a relevant task. This problem is typically approached by training a meta-model that can transfer its knowledge to a task-specific solver, where the performance of the meta-model is evaluated on the basis of how well each solver performs on the corresponding task. To learn such meta-model, one should be able to (a) train an appropriate solver for each task utilizing the knowledge transferred from the meta-model, and (b) accurately evaluate the performance of the solver. A standard way to do this is the so-called $\mathcal{S}/\mathcal{Q}$ (support/query) protocol [55, 34]: for (a), use a set of *support set* samples to train the solver; for (b), use another set of samples, called *query set* samples to evaluate the solver[1].

Recently, however, an alternative paradigm—called $\mathcal{S}/\mathcal{T}$ (support/target) protocol—has received much attention [58, 62, 32]. The approach assumes that the meta-learner has an access to task-specific *target models*, i.e., an expert model for each given task, and uses these models to evaluate task-specific solvers by measuring the discrepancy of the solvers from the target models. Intriguingly, it has been observed that such knowledge distillation procedure [43, 21] helps to improve the meta-generalization performance [62], in a similar way that such teacher-student framework helps to avoid overfitting under non-meta-learning contexts [30, 24].

[1]We give an overview of terminologies used in the paper to guide readers new to this field (see Appendix A).

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

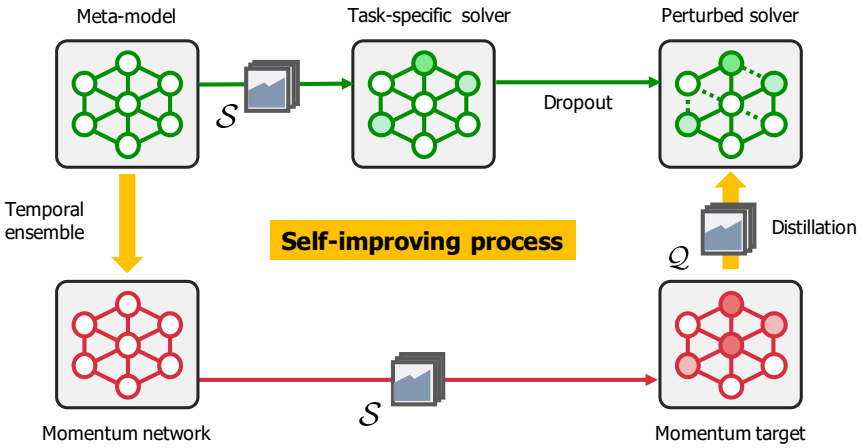


Figure 1: An overview of the proposed *Self-improving Momentum Target (SiMT)*: the momentum network efficiently generates the target model, and by distilling knowledge to the task-specific solver, it forms a self-improving process. $\mathcal{S}$ and $\mathcal{Q}$ denote the support and query datasets, respectively.

Despite such advantage, the $\mathcal{S}/\mathcal{T}$ protocol is difficult to be used in practice, as training target models for each task usually requires an excessive computation, especially when the number of tasks is large. Prior works aim to alleviate this issue by generating target models in a compute-efficient manner. For instance, Lu et al. [32] consider the case where the learner has an access to a model pre-trained on a global data domain that covers most tasks (to be meta-trained upon), and propose to generate task-wise target models by simply fine-tuning the model for each task. However, the method still requires to compute for fine-tuning on a large number of tasks, and more importantly, is hard to be used when there is no effective pre-trained model available, e.g., a globally pre-trained model is usually not available in reinforcement learning, as collecting "global" data is a nontrivial task [9].

In this paper, we ask whether we can generate the task-specific target models by (somewhat ironically) using meta-learning. We draw inspiration from recent observations in semi/self-supervised learning literature [50, 16, 5] that the temporal ensemble of a model, i.e., the momentum network [27], can be an effective teacher of the original model. It turns out that a similar phenomenon happens in the meta-learning scenario: one can construct a momentum network of the meta-model, whose task-specific adaptation is an effective target model from which the task-specific knowledge can be distilled to train the original meta-model.

Contribution. We establish a novel framework, coined *Meta-Learning with Self-improving Momentum Target (SiMT)*, which brings the benefit of the $\mathcal{S}/\mathcal{T}$ protocol to the $\mathcal{S}/\mathcal{Q}$-like scenario where task-specific target models are not available (but have access to query data). The overview of SiMT is illustrated in Figure 1. In a nutshell, SiMT is comprised of two (iterative) steps:

- *Momentum target*: We generate the target model by adapting from the momentum network, which shows better adaptation performance than the meta-model itself. In this regard, generating the target model becomes highly efficient, e.g., one single forward is required when obtaining the momentum target for ProtoNet [45].
- *Self-improving process*: The meta-model enables to improve through the knowledge distillation from the momentum target, and this recursively improves the momentum network by the temporal ensemble. Furthermore, we find that perturbing parameters of the task-specific solver of the meta-model, e.g., dropout [47], further stabilizes this self-improving process by preventing fast convergence of the distillation loss during meta-training.

We verify the effectiveness of SiMT under various applications of meta-learning, including few-shot regression, few-shot classification, and meta-reinforcement learning (meta-RL). Overall, our experimental results show that incorporating the proposed method can consistently and significantly improve the baseline meta-learning methods [10, 31, 36, 45]. In particular, our method improves the few-shot classification accuracy of Conv4 [55] trained with MAML [10] on mini-ImageNet [55] from 47.33% $\rightarrow$ 51.49% for 1-shot, and from 63.27% $\rightarrow$ 68.74% for 5-shot, respectively. Moreover, we show that our framework could even notably improve on the few-shot regression and meta-RL tasks, which supports that our proposed method is indeed domain-agnostic.

2 Related work

Learning from target models. Learning from an expert model, i.e., the target model, has shown its effectiveness across various domains [30, 35, 65, 52]. As a follow-up, recent papers demonstrate that meta-learning can also be the case [58, 62]. However, training independent task-specific target models is highly expensive due to the large space of task distribution in meta-learning. To this end, recent work suggests pre-training a global encoder on the whole meta-training set and finetune target models on each task [32]; however, they are limited to specific domains and still require some computations, e.g., they take more than 6.5 GPU hours to pre-train only 10% of target models while ours require 2 GPU hours for the entire meta-learning process (ProtoNet [45] of ResNet-12 [34]) on the same GPU. Another recent relevant work is bootstrapped meta-learning [11], which generates the target model from the meta-model by further updating the parameters of the task-specific solver for some number of steps with the query dataset. While the bootstrapped target models can be obtained efficiently, their approach is specialized in gradient-based meta-learning schemes, e.g., MAML [10]. In this paper, we suggest an efficient and more generic way to generate the target model during the meta-training.

Learning with momentum networks. The idea of temporal ensembling, i.e., the momentum network, has become an essential component of the recent semi/self-supervised learning algorithms [3, 5]. For example, Mean Teacher [50] first showed that the momentum network improves the performance of semi-supervised image classification, and recent advanced approaches [2, 46] adopted this idea for achieving state-of-the-art performances. Also, in self-supervised learning methods which enforce invariance to data augmentation, such momentum networks are widely utilized as a target network [19, 16] to prevent collapse by providing smoother changes in the representations. In meta-learning, a concurrent work [6] used stochastic weight averaging [23] (a similar approach to the momentum network) to learn a low-rank representation. In this paper, we empirically demonstrate that the momentum network shows better adaptation performance compare to the original meta-model, which motivates us to utilize it for generating the target model in a compute-efficient manner.

3 Problem setup and evaluation protocols

In this section, we formally describe the meta-learning setup under consideration, and $\mathcal{S}/\mathcal{Q}$ and $\mathcal{S}/\mathcal{T}$ protocols studied in prior works.

Problem setup: Meta-learning. Let $p(\tau)$ be a distribution of tasks. The goal of meta-learning is to train a meta-model f_θ, parameterized by the meta-model parameter θ, which can transfer its knowledge to help to train a *solver* for a new task. More formally, we consider some *adaptation subroutine* $\texttt{Adapt}(\cdot,\cdot)$ which uses both information transferred from θ and the task-specific dataset (which we call *support set*) $\mathcal{S}^\tau$ to output a task-specific solver as $\phi^\tau = \texttt{Adapt}(\theta, \mathcal{S}^\tau)$. For example, the model-agnostic meta-learning algorithm (MAML; [10]) uses the adaptation subroutine of taking a fixed number of SGD on $\mathcal{S}^\tau$, starting from the initial parameter θ. In this paper, we aim to give a general meta-learning framework that can be used in conjunction with any adaptation subroutine, instead of designing a method specialized for a specific one.

The objective is to learn a nice meta-model parameter θ from a set of tasks sampled from $p(\tau)$ (or sometimes the task distribution itself), such that the expected loss of the task-specific adaptations is small, i.e., $\min_\theta \mathbb{E}_{\tau\sim p(\tau)}[\ell^\tau(\texttt{Adapt}(\theta, \mathcal{S}^\tau))]$, where $\ell^\tau(\cdot)$ denotes the test loss on task τ. To train such meta-model, we need a mechanism to evaluate and optimize θ (e.g., via gradient descent). For this purpose, existing approaches take one of two approaches: the $\mathcal{S}/\mathcal{Q}$ protocol or the $\mathcal{S}/\mathcal{T}$ protocol.

$\mathcal{S}/\mathcal{Q}$ protocol. The majority of the existing meta-learning frameworks (e.g., [55, 34]) splits the task-specific training data into two, and use them for different purposes. One is the support set $\mathcal{S}^\tau$ which is used to perform the adaptation subroutine. Another is the *query set* $\mathcal{Q}^\tau$ which is used for evaluating the performance of the adapted parameter and compute the gradient with respect to θ. In other words, given the task datasets $(\mathcal{S}_1, \mathcal{Q}_1), (\mathcal{S}_2, \mathcal{Q}_2), \ldots, (\mathcal{S}_N, \mathcal{Q}_N)$,[2] the $\mathcal{S}/\mathcal{Q}$ protocol solves

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(\texttt{Adapt}(\theta, \mathcal{S}^{\tau_i}), \mathcal{Q}^{\tau_i}\big), \tag{1}$$

where $\mathcal{L}(\phi, \mathcal{Q})$ denotes the empirical loss of a solver ϕ on the dataset $\mathcal{Q}$.

[2] Here, while we assumed a static batch of tasks for notational simplicity, the expression is readily extendible to the case of a *stream of tasks* drawn from $p(\tau)$.

$\mathcal{S}/\mathcal{T}$ **protocol.** Another line of work considers the scenario where the meta-learner additionally has an access to a set of *target models* $\phi_{\texttt{target}}$ for each training task [58, 32]. In such case, one can use a teacher-student framework to regularize the adapted solver to behave similarly (or have low *prediction discrepancy*, equivalently) to the target model. Here, a typical practice is to *not split* each task dataset and measure the discrepancy using the support dataset that is used for the adaptation [32]. In other words, given the task datasets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N$ and the corresponding target models $\phi^{\tau_1}_{\texttt{target}}, \phi^{\tau_2}_{\texttt{target}}, \ldots, \phi^{\tau_N}_{\texttt{target}}$, the $\mathcal{S}/\mathcal{T}$ protocol updates the meta-model by solving

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\texttt{teach}}\big(\texttt{Adapt}(\theta, \mathcal{S}^{\tau_i}), \phi^{\tau_i}_{\texttt{target}}, \mathcal{S}^{\tau_i}\big), \tag{2}$$

where $\mathcal{L}_{\texttt{teach}}(\phi, \phi_{\texttt{target}}, \mathcal{S})$ denotes a discrepancy measure between the adapted model ϕ and the target model $\phi_{\texttt{target}}$, measured using the dataset $\mathcal{S}$.

4 Meta-learning with self-improving momentum target

In this section, we develop a compute-efficient framework which bring the benefits of $\mathcal{S}/\mathcal{T}$ protocol to the settings where we do not have access to target-specific tasks or a general pretrained model, as in general $\mathcal{S}/\mathcal{Q}$-like setups. In a nutshell, our framework iteratively generates a *meta-target model* which generalizes well when adapted to the target tasks, by constructing a momentum network [50] of the meta-model itself. The meta-model is then trained, using both the knowledge transferred from the momentum target and the knowledge freshly learned from the query sets. We first briefly describe our meta-model update protocol (Section 4.1), and then the core component, coined *Self-Improving Momentum Target (SiMT)*, which efficiently generates the target model for each task (Section 4.2).

4.1 Meta-model update with a $\mathcal{S}/\mathcal{Q}$-$\mathcal{S}/\mathcal{T}$ hybrid loss

To update the meta-model, we use a hybrid loss function of the $\mathcal{S}/\mathcal{Q}$ protocol (1) and the $\mathcal{S}/\mathcal{T}$ protocol (2). Formally, let $(\mathcal{S}_1, \mathcal{Q}_1), (\mathcal{S}_2, \mathcal{Q}_2), \ldots, (\mathcal{S}_N, \mathcal{Q}_N)$ be given task datasets with support-query split, and let $\phi^{\tau_1}_{\texttt{target}}, \phi^{\tau_2}_{\texttt{target}}, \ldots, \phi^{\tau_N}_{\texttt{target}}$ be task-specific target models generated by our target generation procedure (which will be explained with more detail in Section 4.2). We train the meta-model as

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Big((1-\lambda) \cdot \mathcal{L}(\texttt{Adapt}(\theta, \mathcal{S}^{\tau_i}), \mathcal{Q}^{\tau_i}) + \lambda \cdot \mathcal{L}_{\texttt{teach}}(\texttt{Adapt}(\theta, \mathcal{S}^{\tau_i}), \phi^{\tau_i}_{\texttt{target}}, \mathcal{Q}^{\tau_i}) \Big), \tag{3}$$

where $\lambda \in [0, 1)$ is the weight hyperparameter. We note two things about Eq. 3. First, while we are training using the target model, we also use a $\mathcal{S}/\mathcal{Q}$ loss term. This is because our method trains the meta-target model and the meta-model simultaneously from scratch, instead of requiring fully-trained target models. Second, unlike in the $\mathcal{S}/\mathcal{T}$ protocol, we evaluate the discrepancy $\mathcal{L}_{\texttt{teach}}$ using the query set $\mathcal{Q}^{\tau_i}$ instead of the support set, to improve the generalization performance of the student model. In particular, the predictions of adapted models on query set samples are softer (i.e., having less confidence) than on support set samples, and such soft predictions are known to be beneficial on the generalization performance of the student model in the knowledge distillation literature [64, 49].

4.2 SiMT: Self-improving momentum target

We now describe the algorithm we propose, SiMT (Algorithm 1), to generate the target model in a compute-efficient manner. In a nutshell, SiMT is comprised of two iterative steps: *momentum target* and *self-improving process*. To efficiently generate a target model, SiMT utilizes the temporal ensemble of the network, i.e., the momentum network, then distills the knowledge of the generated target model into the task-specific solver of the meta-model to form a self-improving process.

Momentum target. For the compute-efficient generation of target models, we utilize the momentum network $\theta_{\texttt{moment}}$ of the meta-model. Specifically, after every meta-model training iteration, we compute the exponential moving average of the meta-model parameter θ as

$$\theta_{\texttt{moment}} \leftarrow \eta \cdot \theta_{\texttt{moment}} + (1-\eta) \cdot \theta, \tag{4}$$

where $\eta \in [0, 1)$ is the momentum coefficient. We find that $\theta_{\texttt{moment}}$ can adapt better than the meta-model θ itself and observe that the loss landscape has flatter minima (see Section 5.5), which can

Algorithm 1 SiMT: Self-Improving Momentum Target

Require: Distribution over tasks $p(\tau)$, adaptation subroutine $\texttt{Adapt}(\cdot)$, momentum coefficient η, weight hyperparameter λ, dropout probability p, task batch size N, learning rate β.

Initialize θ using the standard initialization scheme.
Initialize the momentum network with the meta-model parameter, $\theta_{\texttt{moment}} \leftarrow \theta$.
while not done **do**
  Sample N tasks $\{\tau_i\}_{i=1}^{N}$ from $p(\tau)$
  for $i = 1$ to N **do**
    Sample support set $\mathcal{S}^{\tau_i}$ and query set $\mathcal{Q}^{\tau_i}$ from τ_i
    $\phi^{\tau_i}_{\texttt{moment}} = \texttt{Adapt}(\theta_{\texttt{moment}}, \mathcal{S}^{\tau_i})$. ▷ Generate a momentum target.
    $\phi^{\tau_i} = \texttt{Adapt}(\theta, \mathcal{S}^{\tau_i})$. ▷ Adapt a task-specific solver.
    $\phi^{\tau_i}_{\texttt{drop}} = \texttt{Dropout}(\phi^{\tau_i}, p)$. ▷ Perturb the solver.
    $\mathcal{L}^{\tau_i}_{\texttt{total}}(\theta) = (1-\lambda)\cdot\mathcal{L}(\phi^{\tau_i}_{\texttt{drop}}, \mathcal{Q}^{\tau_i}) + \lambda\cdot\mathcal{L}_{\texttt{teach}}(\phi^{\tau_i}_{\texttt{drop}}, \phi^{\tau_i}_{\texttt{moment}}, \mathcal{Q}^{\tau_i})$ ▷ Compute loss.
  end for
  $\theta \leftarrow \theta - \frac{\beta}{N}\cdot\nabla_\theta \sum_{i=1}^{N} \mathcal{L}^{\tau_i}_{\texttt{total}}(\theta)$. ▷ Train the meta-model.
  $\theta_{\texttt{moment}} \leftarrow \eta\cdot\theta_{\texttt{moment}} + (1-\eta)\cdot\theta$. ▷ Update the momentum network.
end while

be a hint for understanding the generalization improvement [29, 12]. Based on this, we propose to generate the task-specific target model, i.e., the momentum target $\phi_{\texttt{moment}}$, by adapting from the momentum network $\theta_{\texttt{moment}}$. For a given support set $\mathcal{S}$, we generate the target model for each task as

$$\phi^{\tau_i}_{\texttt{moment}} = \texttt{Adapt}(\theta_{\texttt{moment}}, \mathcal{S}^{\tau_i}), \qquad \forall i \in \{1, 2, \ldots, N\}. \tag{5}$$

We remark that generating momentum targets does not require an excessive amount of compute (see Section 5.5), e.g., ProtoNet [45] requires a single forward of a support set, and MAML [10] requires few-gradient steps without second-order gradient computation for the adaptation.

Self-improving process via knowledge distillation. After generating the momentum target, we utilize its knowledge to improve the generalization performance of the meta-model. To this end, we choose the knowledge distillation scheme [21], which is simple yet effective across various domains, including meta-learning [32]. Here, our key concept is that the momentum target *self-improves* during the training due to the knowledge transfer. To be specific, the knowledge distillation from the momentum target improves the meta-model itself, which recursively improves the momentum through the temporal ensemble. Formally, for a given query set $\mathcal{Q}$, we distill the knowledge of the momentum target $\phi_{\texttt{moment}}$ to the task-specific solver of the meta-model ϕ as

$$\mathcal{L}_{\texttt{teach}}(\phi, \phi_{\texttt{moment}}, \mathcal{Q}) := \frac{1}{|\mathcal{Q}|}\sum_{(x,y)\in\mathcal{Q}} l_{\texttt{KD}}\big(f_{\phi_{\texttt{moment}}}(x), f_\phi(x)\big), \tag{6}$$

where $l_{\texttt{KD}}$ is the distillation loss and $|\cdot|$ is the cardinality of the set. For regression tasks, we use the MSE loss, i.e., $l_{\texttt{KD}}(z_1, z_2) := \|z_1 - z_2\|_2^2$, and for classification tasks, we use the KL divergence with temperature scaling [17], i.e., $l_{\texttt{KD}}(z_1, z_2) := T^2\cdot\texttt{KL}(\sigma(z_1/T) \parallel \sigma(z_2/T))$, where T is the temperature hyperparameter, σ is the softmax function and z_1, z_2 are logits of the classifier, respectively. We present the detailed distillation objective of reinforcement learning tasks in Appendix C. Also, note that optimizing the distillation loss only propagates gradients to the meta-model θ, not to the momentum network $\theta_{\texttt{moment}}$, i.e., known as the stop-gradient operator [5, 7].

Furthermore, we find that the distillation loss (6) sometimes converges too fast during the meta-training, which can stop the self-improving process. To prevent this, we suggest perturbing the parameter space of ϕ. Intuitively, injecting noise to the parameter space of ϕ forces an asymmetricity between the momentum target's prediction, hence, preventing f_ϕ and $f_{\phi_{\texttt{moment}}}$ from reaching a similar prediction. To this end, we choose the standard dropout regularization [47] due to its simplicity and generality across architectures and also have shown its effectiveness under distillation research [60]: $\phi_{\texttt{drop}} := \texttt{Dropout}(\phi, p)$ where p is the probability of dropping activations. In the end, we use the perturbed task-specific solver $\phi_{\texttt{drop}}$ and the momentum target $\phi_{\texttt{moment}}$ for our evaluation protocol (3).

Table 1: Few-shot regression results on ShapeNet and Pascal datasets. We report the angular error for ShapeNet, and MSE for Pascal. SiMT use the momentum network at meta-test time. Reported results are averaged over three trials, subscripts denote the standard deviation, and bold denotes the best result of each group.

	ShapeNet		Pascal	
Method	10-shot	15-shot	10-shot	15-shot
MAML [10]	29.555±0.600	22.286±3.369	2.612±0.280	2.513±0.250
MAML [10] + SiMT	**18.913±2.655**	**16.100±1.318**	**1.462±0.230**	**1.229±0.074**
ANIL [36]	39.915±0.665	38.202±1.388	6.600±0.360	6.517±0.420
ANIL [36] + SiMT	**37.424±0.951**	**29.478±0.212**	**5.339±0.321**	**5.007±0.145**
MetaSGD [31]	17.353±1.110	15.768±1.266	3.532±0.381	2.833±0.216
MetaSGD [31] + SiMT	**16.121±1.322**	**14.377±0.358**	**2.300±0.871**	**1.879±0.134**

Table 2: Few-shot in-domain adaptation accuracy (%) on 5-way mini- and tiered-ImageNet. SiMT use the momentum network at meta-test time. Reported results are averaged over three trials, subscripts denote the standard deviation, and bold denotes the best result of each group.

		mini-ImageNet		tiered-ImageNet	
Model	Method	1-shot	5-shot	1-shot	5-shot
Conv4 [55]	MAML [10]	47.33±0.45	63.27±0.14	50.19±0.21	66.05±0.19
	MAML [10] + SiMT	**51.49±0.18**	**68.74±0.12**	**52.51±0.21**	**69.58±0.11**
	ANIL [36]	47.71±0.47	63.13±0.43	49.57±0.04	66.34±0.28
	ANIL [36] + SiMT	**50.81±0.56**	**67.99±0.19**	**51.66±0.26**	**68.88±0.08**
	MetaSGD [31]	50.66±0.18	65.55±0.54	52.48±1.22	71.06±0.20
	MetaSGD [31] + SiMT	**51.70±0.80**	**69.13±1.40**	**52.98±0.07**	**71.46±0.12**
	ProtoNet [45]	47.97±0.29	65.16±0.67	51.90±0.55	71.51±0.25
	ProtoNet [45] + SiMT	**51.25±0.55**	**68.71±0.35**	**53.25±0.27**	**72.69±0.27**
ResNet-12 [34]	MAML [10]	52.66±0.60	68.69±0.33	57.32±0.59	73.78±0.27
	MAML [10] + SiMT	**56.28±0.63**	**72.01±0.26**	**59.72±0.22**	**74.40±0.90**
	ANIL [36]	51.80±0.59	68.38±0.20	57.52±0.68	73.50±0.35
	ANIL [36] + SiMT	**54.44±0.27**	**69.98±0.66**	**58.18±0.31**	**75.59±0.50**
	MetaSGD [31]	54.95±0.11	70.65±0.43	58.97±0.89	76.37±0.11
	MetaSGD [31] + SiMT	**55.72±0.96**	**74.01±0.79**	**61.03±0.05**	**78.04±0.48**
	ProtoNet [45]	52.84±0.21	68.35±0.29	61.16±0.17	79.94±0.20
	ProtoNet [45] + SiMT	**55.84±0.57**	**72.45±0.32**	**62.01±0.42**	**81.82±0.12**

5 Experiments

In this section, we experimentally validate the effectiveness of the proposed SiMT by measuring its performance on various meta-learning applications, including few-shot regression (Section 5.1), few-shot classification (Section 5.2), and meta-reinforcement learning (meta-RL; Section 5.3).

Common setup. By following the prior works, we chose the checkpoints and the hyperparameters on the meta-validation set for the few-shot learning tasks [33, 56]. For RL, we chose it based on the best average return during the training [10]. We find that the hyperparameters, e.g., momentum coefficient η or the weight hyperparameter λ, are not sensitive across datasets and architectures but can vary on the type of the meta-learning scheme or tasks. We provide further details in Appendix D. Moreover, we report the adaptation performance of the *momentum network* for SiMT.

5.1 Few-shot regression

For regression tasks, we demonstrate our experiments on ShapeNet [13] and Pascal [63] datasets, where they aim to predict the object pose of a gray-scale image relative to the canonical orientation. To this end, we use the following empirical loss $\mathcal{L}$ to train the meta-model: the angular loss for the ShapeNet ($\sum_{(x,y)\in\mathcal{Q}} \|\cos(f_\phi(x)) - \cos(y)\|^2 + \|\sin(f_\phi(x)) - \sin(y)\|^2$) and the MSE loss for the

Table 3: Few-shot cross-domain adaptation accuracy (%) on ResNet-12 trained under 5-way mini- and tiered-ImageNet. We consider CUB and Cars as cross-domain datasets. SiMT use the momentum network at meta-test time. Reported results are averaged over three trials, subscripts denote the standard deviation, and bold denotes the best result of each group.

		mini-ImageNet →		tiered-ImageNet →	
Problem	Method	CUB	Cars	CUB	Cars
1-shot	MAML [10]	39.50±0.91	32.87±0.20	42.32±0.69	36.62±0.12
	MAML [10] + SiMT	**42.32±0.62**	**33.73±0.63**	**44.33±0.43**	**37.21±0.35**
	ANIL [36]	37.30±0.89	31.28±1.03	42.29±0.33	36.27±0.58
	ANIL [36] + SiMT	**38.86±0.98**	**32.34±0.95**	**44.53±1.21**	**36.92±0.56**
	MetaSGD [31]	41.98±0.18	**34.52±0.56**	46.48±2.10	38.09±1.21
	MetaSGD [31] + SiMT	**43.50±0.89**	33.92±0.30	**46.62±0.41**	**38.69±0.26**
	ProtoNet [45]	41.22±0.81	32.79±0.61	47.75±0.56	37.59±0.80
	ProtoNet [45] + SiMT	**44.13±0.30**	**34.53±0.40**	**48.89±0.65**	**38.07±0.42**
5-shot	MAML [10]	56.17±0.92	44.56±0.79	65.00±0.89	51.08±0.28
	MAML [10] + SiMT	**59.22±0.39**	**46.59±0.21**	**67.58±0.61**	**51.88±0.52**
	ANIL [36]	53.42±0.97	41.65±0.67	62.48±0.85	50.50±1.18
	ANIL [36] + SiMT	**56.03±1.40**	**45.88±0.82**	**66.30±0.99**	**54.60±0.91**
	MetaSGD [31]	58.90±1.30	47.44±1.55	70.38±0.27	56.28±0.07
	MetaSGD [31] + SiMT	**65.07±1.89**	**49.86±0.84**	**73.93±0.42**	**57.97±1.34**
	ProtoNet [45]	57.87±0.77	48.06±1.10	74.35±0.93	57.23±0.25
	ProtoNet [45] + SiMT	**63.85±0.76**	**51.67±0.29**	**75.97±0.09**	**59.01±0.50**

Pascal ($\sum_{(x,y)\in\mathcal{Q}}\|f_\phi(x)-y\|^2$), following the prior works [63, 13]. For the backbone meta-learning schemes we use gradient-based approaches, including MAML [10], ANIL [36], and MetaSGD [31]. For all methods, we train the convolutional neural network with 7 layers [63] and apply dropout regularization [47] before the max-pooling layer for SiMT. Table 1 summarizes the results, showing that SiMT significantly improves the overall meta-learning schemes in all tested cases.

5.2 Few-shot classification

For few-shot classification tasks, we use the cross-entropy loss for the empirical loss term $\mathcal{L}$ to train the meta-model θ., i.e., $\sum_{(x,y)\in\mathcal{Q}} l_{\texttt{ce}}(f_\phi(x), y)$ where $l_{\texttt{ce}}$ is the cross-entropy loss. We train the meta-model on mini-ImageNet [55] and tiered-ImageNet [38] datasets, following the prior works [32, 56]. Here, we consider the following gradient-based and metric-based meta-learning approaches as our backbone algorithm to show the wide usability of our method: MAML, ANIL, MetaSGD, and ProtoNet [45]. We train each method on Conv4 [55] and ResNet-12 [34], and apply dropout before the max-pooling layer for SiMT. For the training details, we mainly follow the setups from each backbone algorithm paper. See Appendix D.1 for more details.

In-domain adaptation. In this setup, we evaluate the adaptation performance on different classes of the same dataset used in meta-training. As shown in Table 2, incorporating SiMT into existing meta-learning methods consistently and significantly improves the in-domain adaptation performance. In particular, SiMT achieves higher accuracy gains on the mini-ImageNet dataset, e.g., 5-shot performance improves from 63.27% → 68.74% on Conv4. We find that this is due to the overfitting issue of backbone algorithms on the mini-ImageNet dataset, where SiMT is more robust to such issues. For instance, when training mini-ImageNet 5-shot classification on Conv4, MAML starts to overfit after the first 40% of the training process, while SiMT does not overfit during the training.

Cross-domain adaptation. We also consider the cross-domain adaptation scenarios. Here, we adapt the meta-model on different datasets from the meta-training: we use CUB [57] and Cars [26] datasets. Such tasks are known to be challenging, as there exists a large distribution shift between training and testing domains [18]. Table 3 shows the results. Somewhat interestingly, SiMT also improves the cross-domain adaptation performance of the base meta-learning methods across the considered datasets. These results indicate that SiMT successfully learns the ability to generalize to unseen tasks even for the distributions that highly differ from the training.

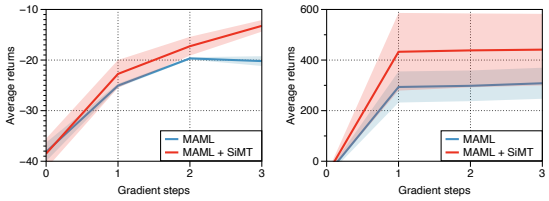
Table 4: Comparison with bootstrapped targets (Bootstrap) [16] on few-shot in-domain adaptation tasks. We report the adaptation accuracy (%) of Conv4 trained under 5-way mini- and tiered-ImageNet. SiMT use the momentum network at meta-test time. Reported results are averaged over three trials, subscripts denote the standard deviation, and bold indicates the best result of each group.

	mini-ImageNet		tiered-ImageNet	
Method	1-shot	5-shot	1-shot	5-shot
MAML [10]	47.33±0.45	63.27±0.14	50.19±0.21	66.05±0.19
MAML [10] + Bootstrap [16]	48.68±0.33	68.45±0.40	49.34±0.26	68.84±0.37
MAML [10] + SiMT	**51.49±0.18**	**68.74±0.12**	**52.51±0.21**	**69.58±0.11**
ANIL [36]	47.71±0.47	63.13±0.43	49.57±0.04	66.34±0.28
ANIL [36] + Bootstrap [16]	47.74±0.44	65.16±0.04	48.85±0.34	66.09±0.07
ANIL [36] + SiMT	**50.81±0.56**	**67.99±0.19**	**51.66±0.26**	**68.88±0.08**

5.3 Reinforcement learning

The goal of meta-RL is training an agent to quickly adapt a policy to maximize the expected return for unseen tasks using only a limited number of sample trajectories. Since the expected return is usually not differentiable, we use policy gradient methods to update the policy. Specifically, we use vanilla policy gradient [59], and trust-region policy optimization (TRPO; [39]) for the task-specific solver and meta-model, respectively, following MAML [10]. The overall training objective of meta-RL is in Appendix C, including the empirical loss $\mathcal{L}$, and the knowledge distillation loss $\mathcal{L}_{\texttt{teach}}$. We evaluate SiMT on continuous control tasks based on OpenAI Gym [4] environments. In these experiments, we choose MAML as our backbone algorithm, and train a multi-layer perceptron policy network with two hidden layers of size 100 by following the prior setup [10]. We find that the distillation loss is already quite effective even without the dropout regularization, and applying it does not improve more. We conjecture that dropout on such a small network may not be effective as it is designed to reduce the overfitting of large networks [47]. We provide more experimental details in Appendix D.1.

2D Navigation. We first evaluate SiMT on a 2D Navigation task, where a point agent moves to different goal positions which are randomly chosen within a 2D unit square. Figure 2 shows the adaptation performance of learned models with up to three gradient steps. These results demonstrate that SiMT could consistently improve the adaptation performance of MAML. Also, SiMT makes faster performance improvements than vanilla MAML with additional gradient steps.



(a) 2D Navigation (b) Half-cheetah Direction

Figure 2: Meta-RL results for (a) 2D navigation and (b) Half-cheetah locomotion tasks. The solid line and shaded regions represent the truncated mean and standard deviation, respectively, across five runs.

Locomotion. To further demonstrate the effectiveness of our method, we also study high-dimensional, complex locomotion tasks based on the MuJoCo [53] simulator. We choose a set of goal direction tasks with a planar cheetah ("Half-cheetah"), following previous works [10, 36]. In the goal direction tasks, the reward is the magnitude of the velocity in either the forward or backward direction, randomly chosen for each task. Figure 2b shows that SiMT significantly improves the adaptation performance of MAML even with a single gradient step.

5.4 Comparison with other target models

In this section, we compare SiMT with other meta-learning schemes that utilize the target model, including bootstrapped [16] and task-wise pre-trained target models [32].

Bootstrapped target model. We compare SiMT with a recent relevant work, bootstrapped meta-learning (Bootstrap) [16]. The key difference is on how to construct the target model $\phi_{\texttt{target}}$: SiMT utilizes the momentum network while Bootstrap generates the target by *further updating* the parameters of the task-specific solver. Namely, Bootstrap relies on gradient-based adaptation while SiMT does not. This advantage allows SiMT to incorporate various non-gradient-based meta-learning approaches, e.g., ProtoNet [45], as shown in Table 2 of the main text. Furthermore, SiMT

Table 5: Comparison with pre-trained target models on few-shot in-domain adaptation tasks. We report the adaptation accuracy (%) of ResNet-12 trained on 5-way mini- and tiered-ImageNet. SiMT use the learned momentum network at meta-test time. Reported results are averaged over three trials, subscripts denote the standard deviation, and bold indicates the best result of each group. * indicates the values from the reference and percentage after the bar denotes the proportion of tasks with pre-trained target models for meta-training [32].

Method	1-shot train cost (GPU hours)	mini-ImageNet		tiered-ImageNet	
		1-shot	5-shot	1-shot	5-shot
MAML [10]*	1.31	58.84±0.25	74.62±0.38	63.02±0.30	67.26±0.32
MAML [10] + Lu et al. [32] - 5%*	5.04	59.14±0.33	75.77±0.29	64.52±0.30	68.39±0.34
MAML [10] + Lu et al. [32] - 10%*	8.32	60.06±0.35	76.34±0.42	**65.23**±0.45	70.02±0.33
MAML [10] + SiMT	1.64	**62.05**±0.39	**78.77**±0.45	63.91±0.32	**77.43**±0.47

Table 6: Ablation study on each component of SiMT. We report the few-shot in-domain adaptation accuracy (%) on Conv4 trained with mini-ImageNet. Here, we use the learned momentum network at meta-test time, except for the first experiment of the table. The reported results are averaged over three trials, subscripts denote the standard deviation, and bold denotes the best result.

Momentum	Distillation	Dropout	1-shot	5-shot
-	-	-	47.33±0.45	63.27±0.14
✓	-	-	48.98±0.32	66.12±0.21
✓	✓	-	49.23±0.24	66.52±0.15
✓	-	✓	49.25±0.41	65.25±0.15
✓	✓	✓	**51.49**±0.18	**68.74**±0.12

shows not only wider applicability but also better performance than Bootstrap. As shown in Table 4, SiMT consistently outperforms Bootstrap in few-show learning experiments, which implies that the momentum target is more effective than the bootstrapped target.

Pre-trained target model. We compare SiMT with Lu et al. [32] which utilizes the task-wise pre-trained target model for meta-learning. To this end, we train SiMT upon a ResNet-12 backbone pre-trained on the meta-training set, by following [32]. As shown in Table 5, SiMT consistently improves over MAML, and more intriguingly, SiMT performs even better than Lu et al. [32]. We conjecture this is because SiMT can fully utilize the target model for all tasks due to its efficiency, while Lu et al. [32] should partially sample the tasks with the target model due to the computational burden: note that when generating target models, SiMT only requires additional 0.3 GPU hours for all tasks while Lu et al. [32] requires more than 3.7 GPU hours for 5% of tasks.

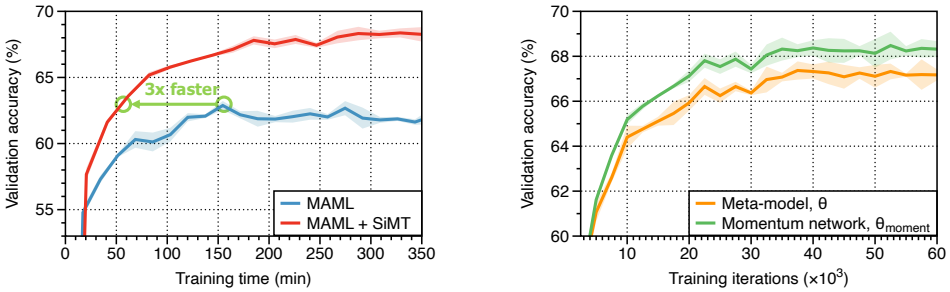
5.5 Ablation study

Throughout this section, unless otherwise specified, we perform the experiments in 5-shot in-domain adaptation on mini-ImageNet with Conv4, where MAML is the backbone meta-learning scheme.

Component analysis. We perform an analysis on each component of our method in both 1-shot and 5-shot classification on mini-ImageNet: namely, the use of (a) the momentum network $\theta_{\texttt{moment}}$, (b) the distillation loss $\mathcal{L}_{\texttt{teach}}$ (6), and (c) the dropout regularization $\texttt{Dropout}(\cdot)$, by comparing the accuracies. The results in Table 6 show each component is indeed important for the improvement. We find that a naïve combination of the distillation loss and the momentum network does not show significant improvements. But, by additionally applying the dropout, the distillation loss becomes more effective and further improves the performance. Note that this improvement does not fully come from the dropout itself, as only using dropout slightly degrades the performance in some cases.

Computational efficiency. Our method may be seemingly compute-inefficient when incorporating meta-learning methods (due to the momentum target generation); however, we show that it is not. Although SiMT increases the total training time of MAML by roughly 1.2 times, we have observed that it is 3 times faster to achieve the best performance of MAML: in Figure 3a, we compare the accuracy under the same training wall-clock time with MAML.

Comparison of the momentum network and meta-model. To understand how the momentum network improves the performance of the meta-model, we compare the adaptation performance of the momentum network and the meta-model during training SiMT. As shown in Figure 3b, we observe

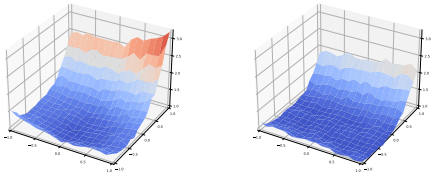


(a) Computation efficiency (b) Network choice for the adaptation

Figure 3: Validation accuracy curves of 5-shot mini-ImageNet on Conv4: we compare the adaptation performance of (a) MAML and SiMT, under the same training wall-clock time, and (b) the meta-model and the momentum network of SiMT, under the same number of training steps. The solid line and shaded regions represent the mean and standard deviation, respectively, across three runs.

that the performance of the momentum network is consistently better than the meta-model, which implies that the proposed momentum target is a nice target model in our self-improving mechanism.

Loss landscape analysis. We visualize the loss landscape of the momentum network $\theta_{\texttt{moment}}$ and the meta-model θ, to give insights into the generalization improvement. To do this, we train MAML with a momentum network (without distillation and dropout) and visualize the loss by perturbing each parameter space [29] (See Appendix D.2 for the detail of the visualization method). As shown in Figure 4, the momentum network forms a flatter loss landscape than the meta-model, where recent studies demonstrate that such a flat landscape is effective under various domains [12].



(a) Meta-model (b) Momentum network

Figure 4: Loss landscape visualization of Conv4 trained on 5-shot mini-ImageNet with MAML.

6 Discussion and conclusion

In this paper, we propose a simple yet effective method, SiMT, for improving meta-learning. Our key idea is to efficiently generate target models using a momentum network and utilize its knowledge to self-improve the meta-learner. Our experiments demonstrate that SiMT significantly improves the performance of meta-learning methods on various applications.

Limitations and future work. While SiMT is a compute-efficient way to use target models in meta-learning, it is still built on top of existing meta-model update techniques. Since existing meta-learning methods have limited scalability (to large-scale scenarios) [44], SiMT is no exception. Hence, improving the scalability of meta-learning schemes is an intriguing future research direction, where we believe incorporating SiMT into such scenarios is worthwhile.

Potential negative impacts. Meta-learning often requires a large computation due to the numerous task adaptation during meta-training, therefore raising environmental concerns, e.g., carbon generation [41]. As SiMT is built upon the meta-learning methods, practitioners may need to consider some computation for successful training. To address this issue, sparse adaptation scheme [42] or lightweight methods for meta-learning [28] would be required for the applications.

Acknowledgements

We thank Younggyo Seo, Jaehyun Nam, and Minseon Kim for providing helpful feedbacks and suggestions in preparing an earlier version of the manuscript. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST), No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH), and No.2022-0-00713, Meta-learning applicable to real-world problems).

References

[1] S. M. R. Arnold, P. Mahajan, D. Datta, I. Bunner, and K. S. Zarkias. learn2learn: A library for Meta-Learning research. *arXiv preprint arXiv:2008.12284*, 2020.

[2] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel. MixMatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, 2019.

[3] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel. ReMixMatch: Semi-supervised learning with distribution alignment and augmentation anchoring. In *International Conference on Learning Representations*, 2020.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *IEEE International Conference on Computer Vision*, 2021.

[6] K. Chen and C.-G. Lee. Meta-free few-shot learning via representation learning with weight averaging. In *International Joint Conference on Neural Networks*, 2022.

[7] X. Chen and K. He. Exploring simple Siamese representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[8] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, 2016.

[9] G. Dulac-Arnold, D. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning. In *International Conference on Machine Learning*, 2019.

[10] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[11] S. Flennerhag, Y. Schroecker, T. Zahavy, H. van Hasselt, D. Silver, and S. Singh. Bootstrapped meta-learning. In *International Conference on Learning Representations*, 2022.

[12] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

[13] N. Gao, H. Ziesche, N. A. Vien, M. Volpp, and G. Neumann. What matters for meta-learning vision regression tasks? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[14] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

[15] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, 2018.

[16] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap your own latent – a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, 2020.

[17] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

[18] Y. Guo, N. C. Codella, L. Karlinsky, J. V. Codella, J. R. Smith, K. Saenko, T. Rosing, and R. Feris. A broader study of cross-domain few-shot learning. In *European Conference on Computer Vision*, 2020.

[19] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[20] N. Hilliard, L. Phillips, S. Howland, A. Yankov, C. D. Corley, and N. O. Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018.

[21] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[22] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[23] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence*, 2018.

[24] Y. Jang, H. Lee, S. J. Hwang, and J. Shin. Learning what and where to transfer. In *International Conference on Machine Learning*, 2019.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[26] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D object representations for fine-grained categorization. In *International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

[27] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017.

[28] J. Lee, J. Tack, N. Lee, and J. Shin. Meta-learning sparse implicit neural representations. In *Advances in Neural Information Processing Systems*, 2021.

[29] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, 2018.

[30] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

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

[32] S. Lu, H.-J. Ye, L. Gan, and D.-C. Zhan. Towards enabling meta-learning from target models. In *Advances in Neural Information Processing Systems*, 2021.

[33] J. Oh, H. Yoo, C. Kim, and S.-Y. Yun. Boil: Towards representation change for few-shot learning. In *International Conference on Learning Representations*, 2021.

[34] B. Oreshkin, P. Rodríguez López, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, 2018.

[35] W. Park, D. Kim, Y. Lu, and M. Cho. Relational knowledge distillation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[36] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

[37] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[38] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018.

[39] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

[40] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

[41] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

[42] J. R. Schwarz and Y. W. Teh. Meta-learning sparse compression networks. *Transactions on Machine Learning Research*, 2022.

[43] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

[44] J. Shin, H. B. Lee, B. Gong, and S. J. Hwang. Large-scale meta-learning with continual trajectory shifting. In *International Conference on Machine Learning*, 2021.

[45] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[46] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, 2020.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014.

[48] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[49] J. Tang, R. Shivanna, Z. Zhao, D. Lin, A. Singh, E. H. Chi, and S. Jain. Understanding and improving knowledge distillation. *arXiv preprint arXiv:2002.03532*, 2020.

[50] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, 2017.

[51] S. Thrun and L. Pratt. *Learning to Learn*. Springer, 1998.

[52] Y. Tian, D. Krishnan, and P. Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

[53] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[54] H.-Y. Tseng, H.-Y. Lee, J.-B. Huang, and M.-H. Yang. Cross-domain few-shot classification via learned feature-wise transformation. In *International Conference on Learning Representations*, 2020.

[55] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

[56] J. Von Oswald, D. Zhao, S. Kobayashi, S. Schug, M. Caccia, N. Zucchet, and J. Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. In *Advances in Neural Information Processing Systems*, 2021.

[57] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[58] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*, 2016.

[59] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

[60] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[61] H. Yao, L.-K. Huang, L. Zhang, Y. Wei, L. Tian, J. Zou, J. Huang, et al. Improving generalization in meta-learning via task augmentation. In *International Conference on Machine Learning*, 2021.

[62] H. Ye, L. Ming, D. Zhan, and W. Chao. Few-shot learning with a strong teacher. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[63] M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn. Meta-learning without memorization. In *International Conference on Learning Representations*, 2020.

[64] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng. Revisiting knowledge distillation via label smoothing regularization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[65] S. Yun, J. Park, K. Lee, and J. Shin. Regularizing class-wise predictions via self-knowledge distillation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code is available at `https://github.com/jihoontack/SiMT`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Appendix D.1 for the training details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.1 for details.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

Meta-Learning with Self-Improving Momentum Target

A Overview of terminologies used in the paper

- **Meta-model θ.** The meta-learner network, i.e., learns to generalize on new tasks.
- **Adaptation subroutine $\texttt{Adapt}(\cdot,\cdot)$.** Algorithm for adapting the meta-model into a task expert by using a given task dataset.
- **Support set $\mathcal{S}$.** A dataset sampled from a given task distribution that is used for the adaptation.
- **Query set $\mathcal{Q}$.** A dataset sampled from a given task distribution (that is disjoint with the support set) to evaluate the adaptation performance of the algorithm.
- **Task-specific solver ϕ.** Network adapted from the meta-model using the support set by using the adaptation subroutine, i.e., $\texttt{Adapt}(\theta, \mathcal{S})$
- **Momentum network $\theta_{\texttt{moment}}$.** Temporal ensemble of the meta-model where we use the exponential moving average of the meta-model parameter to compute the momentum.
- **Momentum target $\phi_{\texttt{moment}}$.** Network adapted from the momentum network using the support set by using the adaptation subroutine, i.e., $\texttt{Adapt}(\theta_{\texttt{moment}}, \mathcal{S})$

B Overview of adaptation subroutine algorithms

MAML [10] and extensions. MAML uses the adaptation subroutine of taking a fixed number of SGD on the support set $\mathcal{S}$, starting from the meta-model parameter θ. Formally, for a given $\mathcal{S}$, MAML with one step SGD obtains the task-specific solver ϕ by minimizing the empirical loss $\mathcal{L}$, as

$$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{S}), \tag{7}$$

where α denotes the step size. Here, we assume the empirical loss $\mathcal{L}$ is averaged over the given set $\mathcal{S}$. One can easily extend Eq. (7) to obtain ϕ with more than one SGD step from the meta-model θ. For the extension, MetaSGD [31] learns the step size α along with the meta-model parameter θ, and ANIL [36] only adapts the last linear layer of the meta-model θ to obtain ϕ.

ProtoNet [45]. The aim of metric-based meta-learning is to perform a non-parametric classifier on top of the meta-model's embedding space f_θ. Specifically, ProtoNet learns a metric space in which classification can be performed by computing distances to prototype vectors of each class, i.e., $c_i := \frac{1}{|\mathcal{S}_i|} \sum_{(x,y)\in\mathcal{S}_i} f_\theta(x)$ where $\mathcal{S}_i$ contains the samples with class i in the support set $\mathcal{S}$. Formally, for a given distance function $d(\cdot,\cdot)$, e.g., l_2 distance, the task-specific solver ϕ of the ProtoNet is as

$$p_\phi(y=i|x) = \frac{\exp(-d(f_\theta(x), c_i))}{\sum_{i'} \exp(-d(f_\theta(x), c_{i'}))}. \tag{8}$$

C Application to reinforcement learning

We consider a reinforcement learning (RL) framework where an agent interacts with an environment in discrete time [48]. At each timestep t, the agent receives a state $\mathbf{s}_t$ from the environment and chooses an action $\mathbf{a}_t$ based on its policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$. Then the environment gives a reward $r(\mathbf{s}_t, \mathbf{a}_t)$ and the agent transitions to the next state $\mathbf{s}_{t+1}$. The return $\mathcal{R} = \sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)$ is defined as discounted cumulative sum of the reward with discount factor $\gamma \in [0, 1)$. As the goal of RL is to train a policy that maximizes the expected return, the loss is defined as a negative expected return as

$$\mathcal{L}(\theta) = -\mathop{\mathbb{E}}_{\mathbf{s}_t, \mathbf{a}_t \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{9}$$

where θ denotes the parameters of the policy. During meta-learning, the empirical version of this loss can be obtained using either the support set $\mathcal{S}^{\tau_i} = \{\mathbf{s}_t, \mathbf{a}_t \sim \pi_\theta\}$ in the adaptation subroutine, or the query set $\mathcal{Q}^{\tau_i} = \{\mathbf{s}_t, \mathbf{a}_t \sim \pi_{\phi^{\tau_i}}\}$ in the meta-update.

Algorithm 2 Self-Improving Momentum Target for Reinforcement Learning

Require: Distribution over tasks $p(\tau)$, adaptation subroutine $\texttt{Adapt}(\cdot)$, momentum coefficient η, weight hyperparameter λ, task batch size N, number of rollouts per task K, learning rate β.

1: Initialize θ using the standard initialization scheme.
2: Initialize the momentum network with the meta-model parameter, $\theta_{\texttt{moment}} \leftarrow \theta$.
3: **while** not done **do**
4: Sample N tasks $\{\tau_i\}_{i=1}^{N}$ from $p(\tau)$
5: **for** $i = 1$ to N **do**
6: Sample K trajectories $\mathcal{S}^{\tau_i}$ using π_θ from τ_i
7: $\phi^{\tau_i}_{\texttt{moment}} = \texttt{Adapt}(\theta_{\texttt{moment}}, \mathcal{S}^{\tau_i})$. ▷ Generate a momentum target.
8: $\phi^{\tau_i} = \texttt{Adapt}(\theta, \mathcal{S}^{\tau_i})$. ▷ Adapt a task-specific solver.
9: Sample trajectories $\mathcal{Q}^{\tau_i}$ using $\pi_{\phi^{\tau_i}}$ from τ_i
10: $\mathcal{L}^{\tau_i}_{\texttt{total}}(\theta) = (1-\lambda)\cdot\mathcal{L}^{\tau_i}_{\texttt{TRPO}} + \lambda\cdot\mathcal{L}_{\texttt{teach}}(\phi^{\tau_i}, \phi^{\tau_i}_{\texttt{moment}}, \mathcal{Q}^{\tau_i})$ ▷ Compute loss.
11: **end for**
12: $\theta \leftarrow \theta - \frac{\beta}{N}\cdot\nabla_\theta \sum_{i=1}^{N}\mathcal{L}^{\tau_i}_{\texttt{total}}(\theta)$. ▷ Train the meta-model.
13: $\theta_{\texttt{moment}} \leftarrow \eta\cdot\theta_{\texttt{moment}} + (1-\eta)\cdot\theta$. ▷ Update the momentum network.
14: **end while**

In meta-RL setup, each task τ_i contains an initial state distribution $q_i(\mathbf{s}_0)$ and a transition distribution $q_i(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. The goal of meta-RL is to optimize a policy π_θ that minimizes $\mathcal{L}(\theta)$ for unseen tasks $\tau_i \sim p(\tau)$ using only a limited number (K) of sampled trajectories. In Algorithm 2, we describe the meta-RL algorithm with our proposed method, SiMT. We note that SiMT can be combined with any (gradient-based) meta-learning approaches and policy gradient methods. In our experiments, we built SiMT upon MAML with vanilla policy gradient [59] and trust-region policy optimization (TRPO; [39]) for the task-specific solver and meta-model, respectively, as we described in Section 5.3.

Here, we describe the detailed objective of the meta-RL used in the experiments. To this end, we use the following standard definitions of the state-action value function Q^π, the value function V^π, and the advantage function A^π.

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathop{\mathbb{E}}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1}, \ldots}\left[\sum_{k=0}^{\infty}\gamma^k r(\mathbf{s}_{t+k})\right], \; V^\pi(\mathbf{s}_t) = \mathop{\mathbb{E}}_{\mathbf{a}_t, \mathbf{s}_{t+1}, \ldots}\left[\sum_{k=0}^{\infty}\gamma^k r(\mathbf{s}_{t+k})\right],$$
$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t), \quad \text{where} \;\; \mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t), \mathbf{s}_{t+1} \sim q(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t). \tag{10}$$

For better clarity, we will omit the notation t if there is no confusion. The gradient of $\mathcal{L}(\theta, \mathcal{S})$ obtained by the vanilla policy gradient method is

$$\nabla_\theta \mathcal{L}(\theta, \mathcal{S}) = -\frac{1}{|\mathcal{S}|}\sum_{\mathbf{s},\mathbf{a}\in\mathcal{S}} \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) A^{\pi_\theta}(\mathbf{s}, \mathbf{a}), \tag{11}$$

where $\mathcal{S}$ is a set of trajectories sampled using the policy π_θ. Then the adaptation subroutine of the parameters θ is performed as follows: $\phi = \texttt{Adapt}(\theta, \mathcal{S}) := \theta - \alpha\cdot\nabla_\theta\mathcal{L}(\theta, \mathcal{S})$. For SiMT, we update the momentum network $\theta_{\texttt{moment}}$ by using the trajectories $\mathcal{S}$ from the policy π_ϕ (line 7 in Algorithm 2), since it performs better and efficient than using additional trajectories from $\pi_{\phi_{\texttt{moment}}}$.

In the meta-update procedure, we use the surrogate objective of TRPO to update the parameters θ. Let θ_{old} and θ denote parameters of current and new policies (for every meta-update iteration of Algorithm 2), respectively. Then the theoretical TRPO update is

$$\theta \leftarrow \arg\min_\theta \; \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}^{\tau_i}_{\texttt{TRPO}}(\theta_{\text{old}}, \theta) \;\; \text{subject to} \;\; \overline{\text{KL}}(\theta_{\text{old}}||\theta) \le \delta, \tag{12}$$

where $\mathcal{L}_{\texttt{TRPO}}(\theta_{\text{old}}, \theta)$ is the (negative) surrogate advantage from adapted parameters ϕ and ϕ_{old}, a measure of how the policy π_ϕ performs relative to the old policy $\pi_{\phi_{\text{old}}}$, using trajectories $\mathcal{Q}$ which is

sampled from the old policy:

$$\mathcal{L}_{\texttt{TRPO}}(\theta_{\text{old}}, \theta) = -\frac{1}{|\mathcal{Q}|} \sum_{\mathbf{s},\mathbf{a}\in\mathcal{Q}} \frac{\pi_\phi(\mathbf{a}|\mathbf{s})}{\pi_{\phi_{\text{old}}}(\mathbf{a}|\mathbf{s})} A^{\pi_{\phi_{\text{old}}}}(\mathbf{s}, \mathbf{a}), \tag{13}$$

and $\overline{\texttt{KL}}(\theta||\theta_{\text{old}})$ is an average KL divergence between policies across states visited by the old policy

$$\overline{\texttt{KL}}(\theta_{\text{old}}||\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|\mathcal{Q}^{\tau_i}|} \sum_{\mathbf{s},\mathbf{a}\in\mathcal{Q}^{\tau_i}} \left[\texttt{KL}\left(\pi_{\phi^{\tau_i}_{\text{old}}}(\cdot|\mathbf{s}) || \pi_{\phi^{\tau_i}}(\cdot|\mathbf{s}) \right) \right]. \tag{14}$$

To utilize SiMT in meta-RL tasks, we define the knowledge distillation loss by using the task-specific policies π_ϕ and $\pi_{\phi_{\texttt{moment}}}$, which are parameterized by ϕ and the momentum target $\phi_{\texttt{moment}}$, respectively. For a given trajectories $\mathcal{Q}$ sampled using π_ϕ, the knowledge distillation loss is as

$$\mathcal{L}_{\texttt{teach}}(\phi, \phi_{\texttt{moment}}, \mathcal{Q}) := \frac{1}{|\mathcal{Q}|} \sum_{\mathbf{s},\mathbf{a}\in\mathcal{Q}} \texttt{KL}\big(\pi_\phi(\mathbf{s}) \parallel \pi_{\phi_{\texttt{moment}}}(\mathbf{s})\big). \tag{15}$$

For meta-learning with SiMT, the objective $\mathcal{L}^{\tau_i}_{\texttt{TRPO}}$ in (12) is replaced with $\mathcal{L}^{\tau_i}_{\texttt{total}}$ in Algorithm 2.

D Experimental details

In this section, we provide the experimental details, including experimental setups and loss landscape visualization. The implementation of all experiments is given in the supplementary material.

D.1 Experimental setup details

Network architecture details. For all few-shot classification experiments, we mainly follow the setups from MAML [10] and also consider the setups from recent papers [33, 56]. The Conv4 classifier consists of four convolutional layers, each with 64 filters, followed by a batch normalization (BN) layer [22] as well as a max-pooling layer with kernel size and stride of 2. The network then projects to its output through a linear layer. We choose to use the 64 channels for each layer by following the recent papers, e.g., ANIL, BOIL [33], and sparse-MAML [56]. For the ResNet-12, we use the residual blocks with channel sizes of 64, 128, 256, and 512, where the architecture is identical to the one used in previous meta-learning studies [34, 33]. For SiMT, we apply dropout before the max-pooling layer. We also observed that the Dropblock regularization [15] (commonly used in ResNet architectures in few-shot classification) shows a similar performance gain as the dropout in ResNet-12. We use additional three convolutional layers with 64 channels in front of Conv4 architecture for all few-shot regression tasks by following the prior works [63, 13].

In the meta-RL setup, the policy is a conditional probability distribution $\pi_\theta(\mathbf{a}|\mathbf{s})$ parameterized by a neural network θ. For a given state vector $\mathbf{s}$, this neural network specifies a distribution over action space. Then one can compute the likelihood $q(\mathbf{a}|\mathbf{s})$ and sample the action $\mathbf{a} \sim q(\cdot|\mathbf{s})$. For our experiments with continuous action spaces, we use a Gaussian distribution, where the covariance matrix is diagonal and independent of the state. Specifically, the policy has a multi-layer perceptron (MLP) which computes the mean and a learnable vector for log standard deviation with the same dimension as $\mathbf{a}$. We use a 2-layer MLP with hidden dimensions of 100 and ReLU activation for the policy, as we described in Section 5.3.

Training details. For gradient-based few-shot classification experiments, we mainly follow MAML [10] and partially follow Von Oswald et al. [56] (e.g., ResNet-12 training setups). For ProtoNet, we follow a recent paper by Yao et al. [61] on Conv4 and extend the setup into ResNet-12 by only changing the training iterations. We follow a recent paper by Gao et al. [13] for few-shot regression tasks. We use Adam optimizer [25] for optimizing the meta-model θ and train 60,000 iterations for Conv4 (including regression models), and 30,000 iterations for ResNet-12. When training, we use the learning rate of $\beta = 1\text{e}{-3}$ for classification and $\beta = 5\text{e}{-4}$ for regression. As for the details of MAML and ANIL, we use 5 step SGD with a fixed step size $\alpha = 1\text{e}{-2}$ for the classification and $\alpha = 2\text{e}{-3}$ for the regression. For all few-shot learning, we use 1 step SGD on MetaSGD. For the classification, we use a task batch size of 4 and 2 tasks for 1-shot and 5-shot gradient-based methods, respectively, and use 1 for the rest. For the regression, we use a task batch size of 10. Note that all adaptation hyperparameters for the task-specific solver ϕ and momentum target $\phi_{\texttt{moment}}$ are the same.

Moreover, we handle BN parameters following the transductive learning setting for gradient-based approaches and use inductive BN parameters for ProtoNet.

For meta-RL experiments, we mainly follow the setups from MAML [10] and use the open-source PyTorch implementation [1] of MAML for RL. To optimize the objective of TRPO (and SiMT), we compute the Hessian-vector products to avoid computing third derivatives and use a line search to ensure improvement of the surrogate objective with the satisfaction of the KL divergence constraint. For both learning and meta-learning updates, we use the standard linear feature baseline proposed by Duan et al. [8]. To estimate the advantage function, we use a generalized advantage estimator (GAE; [40]) with a discount factor of 0.95 and a bias-variance tradeoff of 1.0. In all meta-RL experiments, the policy is trained using a single gradient step with $\alpha = 0.1$, with rollouts $K = 20$ per gradient step. We use a meta batch size N of 20 and 40 for the 2D navigation and the locomotion tasks, respectively. For TRPO, we use $\delta = 0.01$ for all experiments. The agent is meta-updated for 500 iterations, and the model with the best average return during training is used for the evaluation.

Hyperparameter details for SiMT. SiMT requires hyperparameters, including weight hyper-parameter λ, momentum coefficient η, and the dropout probability p.

We first provide the hyperparameter details of few-shot learning. For the momentum coefficient η, we use 0.995 except for the 5-shot classification, where 0.999 shows slightly better performance. For the weight hyper-parameter λ, we use 0.5 for MAML and ANIL, 0.1 for MetaSGD, and $5e-3$ for ProtoNet, respectively. Finally, for the dropout probability p, we use 0.2 for MAML and ANIL, and 0.1 for the rest. Also, we find that the temperature value of $T = 4$ works the best as in the prior knowledge distillation works [65, 32].

For meta-RL experiments, we use the momentum coefficient of 0.995 for 2D navigation and 0.99 for locomotion tasks, respectively. For the weight hyperparameter λ, we use 0.1 for 2D navigation and 0.02 for locomotion tasks, respectively. We linearly ramp up λ for locomotion tasks, as the momentum target might poorly perform at the beginning of the training.

Dataset and environment details. For the few-shot regression task, we use Pascal [63], which contains 65 objects from 10 categories. We sample 50 objects for training and the other 15 objects for testing. 128×128 gray-scale images are rendered for each object with a random rotation in azimuth angle normalized between $[0, 10]$. We also use ShapeNet [13], which includes 30 categories. 27 of these are used during training and the other three categories are used for the evaluation.

For few-shot classification, we use four image datasets, including mini-ImageNet [55, 37], tiered-ImageNet [38], CUB [57], and Cars [26]. We consider a 5-way classification for all tasks by following the prior works [10, 33]. The mini-ImageNet consists of training, validation, and test sets with 64, 16, and 20 classes in each, respectively. The tiered-ImageNet consists of datasets with 351 training, 97 validation, and 160 test classes. By following Hilliard et al. [20], we split CUB dataset into 100 training, 50 validation, and 50 test classes. Cars are split into training, validation, and test sets with 98, 49, and 49 classes in each, respectively, by following Tseng et al. [54].

For meta-RL experiments, we consider 2D navigation and Half-cheetah locomotion tasks, which were considered in previous works [10, 36]. In 2D navigation tasks, the observation is the current position in a 2D unit square, and the action is a velocity command which clipped in the range of $[-0.1, 0.1]$. The reward is the negative squared distance to the goal, and episodes terminate when the agent is within 0.01 of the goal or arrives at the horizon. In Half-cheetah goal direction tasks, the reward is the magnitude of the velocity in either the forward or backward direction, randomly chosen for each task. The horizon H is 100 and 200, for 2D navigation and locomotion tasks, respectively.

Evaluation details. All our few-shot learning results are reported for models that are early-stopped by measuring the average validation set accuracy (across 2,000 validation set tasks for classification and 100 validation set tasks for regression). For the test accuracy, we report the mean accuracy of 2,000 test tasks and 100 test tasks for classification and regression, respectively. By following the original papers, we use 10 SGD steps for the adaptation on MAML and ANIL, and use 1 SGD step on MetaSGD. For meta-RL experiments, the step size during adaptation was set to $\alpha = 0.1$ for the first step, and $\alpha = 0.05$ for all future steps, following the evaluation setup of MAML. We report the truncated mean and standard deviation of the performance using five random seeds, i.e., the statistics after discarding the best and worst seeds, for meta-RL experiments.

Table 7: Few-shot image classification accuracy (%) on ResNet-10 trained with mini-ImageNet. We consider in-domain and cross-domain scenarios where CUB, Cars, Places, and Plantae are used as cross-domain datasets. SiMT utilizes the momentum network for the adaptation. The reported results are 95% confidence intervals averaged over 1,000 meta-test tasks, subscripts denote the standard deviation, and bold denotes the best result of each group. * indicates the values from the reference.

		mini-ImageNet →				
Problem	Method	mini-ImageNet	CUB	Cars	Places	Plantae
1-shot	GNN [14]*	60.77±0.75	45.69±0.68	31.79±0.51	53.10±0.80	35.60±0.56
	GNN [14] + FT [54]*	66.32±0.80	47.47±0.75	31.61±0.53	55.77±0.79	35.95±0.58
	GNN [14] + SiMT	67.22±0.79	48.19±0.71	32.47±0.57	57.41±0.81	**38.13±0.61**
	GNN [14] + FT [54] + SiMT	**68.02±0.80**	**48.75±0.76**	**32.89±0.69**	**58.23±0.86**	38.07±0.60
5-shot	GNN [14]*	80.87±0.56	62.25±0.65	44.28±0.63	70.84±0.65	52.53±0.59
	GNN [14] + FT [54]*	81.98±0.55	66.98±0.68	44.90±0.64	73.94±0.67	53.85±0.62
	GNN [14] + SiMT	84.37±0.56	68.78±0.69	45.61±0.67	76.73±0.66	55.72±0.63
	GNN [14] + FT [54] + SiMT	**85.13±0.55**	**70.09±0.67**	**46.90±0.65**	**78.15±0.64**	**56.60±0.64**

Resource details. For estimating the training time of our method (in Section 2, Section 5.5, and Appendix E.3), we use the same machine and stop other processes: Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz and a single RTX 2080 Ti GPU for the measurement. For the main development, we mainly use Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz and a single RTX 3090 GPU.

D.2 Loss landscape visualization

In this subsection, we provide the loss landscape visualization details of Figure 4. For the experiment, we consider vanilla MAML θ and its momentum network $\theta_{\texttt{moment}}$ without distillation loss and dropout regularization. We change the parameter by adding two direction vectors d_1, d_2 and visualize the resulting loss value. Specifically, for a given task distribution of training set $p(\tau)$, a training loss $\mathcal{L}_{\texttt{train}}(\theta) := \mathbb{E}_{\tau\sim p(\tau)}\left[\mathcal{L}(\texttt{Adapt}(\theta, \mathcal{S}^\tau), \mathcal{Q}^\tau)\right]$, and the center point θ (or $\theta_{\texttt{moment}}$), we visualize

$$\mathcal{L}_{\texttt{train}}(\theta + i \cdot d_1 + j \cdot d_2), \tag{16}$$

where i, j are the step size of the visualization axis. For choosing visualization directions, we sample two random vectors from a unit Gaussian distribution and normalize each filter in the vector to have the same norm of the corresponding filter in θ by following the prior work [29].

E More experimental results

E.1 Comparison with another meta-learning regularization method

We also compare SiMT with a recent regularization method, feature-wise transformation (FT) [54], a regularization method for metric-based meta-learning schemes. FT augments the image features by utilizing the affine transformation in the feature layer to capture the data distribution. While FT shows strong benefits in improving the base meta-learner, it focuses on cross-domain generalization and requires multiple datasets to learn the hyperparameters. In this regard, we consider a more generic scenario and compare SiMT with FT on a standard few-shot classification setup in [54][3]. Here, we train ResNet-10 on the mini-ImageNet dataset and use GNN [14] as a base meta-learning scheme, which shows the best performance when used with FT. Then, we evaluate both methods on few-shot in-domain and cross-domain adaptation scenarios. As shown in Table 7, SiMT consistently outperforms FT in all cases. More importantly, we observed that SiMT and FT have an orthogonal benefit where joint usage further improves the performance.

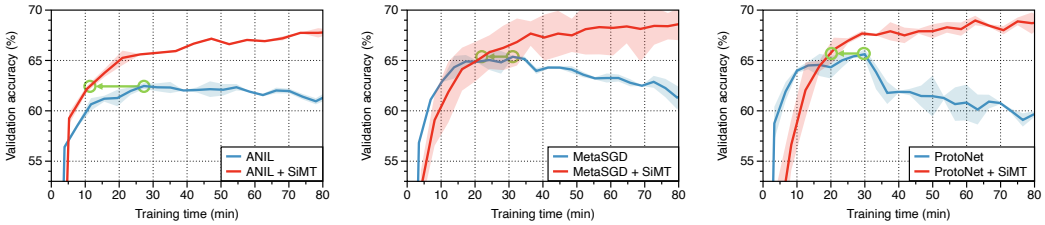
E.2 Momentum targets for a small proportion of tasks

To further investigate the efficacy of the target model, we consider generating target models for a small proportion of tasks when training SiMT. To this end, we train Conv4 on a mini-ImageNet dataset and control the ratio of tasks with the target model. As shown in Table 8, SiMT shows

[3] We use the official implementation from https://github.com/hytseng0509/CrossDomainFewShot

Table 8: Few-show in-domain adaptation accuracy (%) of SiMT under different proportion of tasks with momentum targets. We train Conv4 under 5-way mini-ImageNet dataset. Note that all model utilizes the momentum network for the adaptation. The reported results are averaged over three trials, subscripts denote the standard deviation, and bold indicates the best result of each group. p indicates the proportion of tasks with momentum targets.

p	0%	5%	10%	25%	50%	100%
1-shot	48.98±0.32	49.83±0.12	49.92±0.23	50.42±0.67	50.55±0.41	**51.49±0.18**
5-shot	66.12±0.21	66.81±0.26	66.99±0.24	67.15±0.18	67.49±0.39	**68.74±0.12**



(a) ANIL [36] (b) MetaSGD [31] (c) ProtoNet [45]

Figure 5: Comparison of the computational efficiency between the base adaptation subroutine algorithm and SiMT: we compare the accuracy of mini-ImageNet 5-shot classification (on Conv4) under the same training wall-clock time. The solid line and shaded regions represent the mean and standard deviation, respectively, across three runs.

Table 9: Few-shot in-domain adaptation accuracy (%) on additional adaptation subroutine algorithms. We train Conv4 under 5-way mini- and tiered-ImageNet. SiMT utilizes the momentum network for the adaptation. The reported results are averaged over three trials, subscripts denote the standard deviation, and bold denotes the best result of each group.

	mini-ImageNet		tiered-ImageNet	
Method	1-shot	5-shot	1-shot	5-shot
FOMAML [10]	45.96±0.61	62.58±0.54	47.85±0.46	64.21±0.50
FOMAML [10] + SiMT	**47.78±0.57**	**65.79±0.03**	**48.55±0.08**	**65.79±0.52**
BOIL [33]	49.78±0.65	66.98±0.41	52.19±0.46	68.88±0.43
BOIL [33] + SiMT	**50.83±0.09**	**67.77±0.24**	**52.44±0.28**	**69.05±0.27**

consistent improvement even with a small portion of task models, e.g., 5% of target models show 1% improvement in 1-shot adaptation. However, remark that one can easily generate target models for all tasks with SiMT, hence, one does not need to use few targets in practice.

E.3 Computation efficiency of SiMT

To further analyze the computational efficiency of SiMT, we consider additional comparisons on meta-learning methods, including ANIL [36], MetaSGD [31], and ProtoNet [45] (we train Conv4 on the mini-ImageNet dataset under a 5-shot classification scenario). Here, we also observed consistent efficiency gain on these setups, e.g., 30% of training time reduced to achieve the peak accuracy of ProtoNet when using SiMT. Although SiMT is slower than the base algorithms when comparing the number of optimization steps (due to the momentum target generation), we want to note that SiMT can be more efficient in terms of time to reach the same performance.

E.4 Additional adaptation subroutine algorithms

We consider more adaptation subroutine algorithms to verify the effectiveness of SiMT: first-order approximation of MAML (FOMAML) [10] and BOIL [33]. FOMAML removes the second derivatives of MAML, and BOIL does not adapt the last linear layer of the meta-model θ under MAML. As shown in Table 9, SiMT consistently improves both meta-learning schemes in all tested cases.