Meta-Learning with Self-Improving Momentum Target

Anonymous Author(s)
Affiliation
Address
email

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.

1 Introduction

Meta-learning [38] 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 [41, 25]: 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.

Recently, however, an alternative paradigm—called $\mathcal{S}/\mathcal{T}$ (support/target) protocol—has received much attention [44, 47, 23]. 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 [31, 15] helps to improve the meta-generalization performance [47], in a similar way that such teacher-student framework helps to avoid overfitting under non-meta-learning contexts [21, 16].

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 amount of compute, especially when the number of tasks is large. Prior works aim to alleviate this issue by proposing methods to generate target models in

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

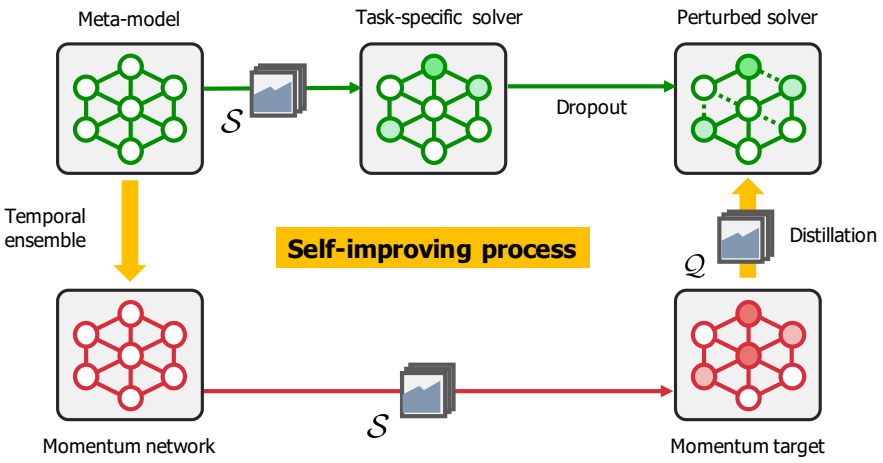


Figure 1: An overview of the proposed *Self-improving Momentum Target (SiMT)*. Overall, the momentum network efficiently generates the target model, and by distilling knowledge to the task-specific solver, it forms a self-improving process. $\mathcal{S}$ and $\mathcal{Q}$ denote the support and query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

a compute-efficient manner. For instance, Lu et al. [23] 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 the 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 at hand. For example, a globally pre-trained model is usually not available in reinforcement learning, as collecting "global" data is a nontrivial task [6].

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 [37, 11, 4] that the temporal ensemble of a model, i.e., the momentum network [18], 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). 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 [33].
- *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 [35], 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 [7, 22, 27, 33]. In particular, our method improves the few-shot classification accuracy of Conv4 [41] trained with MAML [7] on mini-ImageNet [41] 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 [21, 26, 50, 39]. As a follow-up, recent papers demonstrate that meta-learning can also be the case [44, 47]. 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 [23]; 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 [33] of ResNet-12 [25]) on the same GPU. Another recent relevant work is bootstrapped meta-learning [8], 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 [7]. 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 [2, 4]. For example, Mean Teacher [37] first showed that the momentum network improves the performance of semi-supervised image classification, and recent advanced approaches [1, 34] 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 [14, 11] to prevent collapse by providing smoother changes in the representations. 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; [7]) 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., [41, 25]) 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 $(\mathcal{S}_1, \mathcal{Q}_1), (\mathcal{S}_2, \mathcal{Q}_2), \ldots, (\mathcal{S}_N, \mathcal{Q}_N)$,[1] 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 $\mathcal{Q}$.

[1] 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 [44, 23]. 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 [23]. In other words, given the task datasets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N$ and the corresponding target models $\phi^{\mathcal{T}_1}_{\texttt{target}}, \phi^{\mathcal{T}_2}_{\texttt{target}}, \ldots, \phi^{\mathcal{T}_N}_{\texttt{target}}$, the $\mathcal{S}/\mathcal{T}$ protocol updates the meta-model by solving

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

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 [37] 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 $\phi^{\mathcal{T}_1}_{\texttt{target}}, \phi^{\mathcal{T}_2}_{\texttt{target}}, \ldots, \phi^{\mathcal{T}_N}_{\texttt{target}}$ 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

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

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 measure the discrepancy $\mathcal{L}_{\texttt{teach}}$ using the query set $\mathcal{Q}^{\mathcal{T}_i}$ instead of the support set. By doing so, the predictions of target models tend to generate "soft predictions" than on the support set, which is known to improve the generalization of the student model in the knowledge distillation literature [49, 36].

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 5.4), which can be a

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

hint for understanding the generalization improvement [20, 9]. 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 5.4), e.g., ProtoNet [33] requires a single forward of a support set, and MAML [7] 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 [15], which is simple yet effective across various domains, including meta-learning [23]. 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 [12], 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 the Appendix. 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 [4, 5].

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 [35] due to its simplicity and generality across architectures and also have shown its effectiveness under distillation research [46]: $\phi_{\texttt{drop}} := \texttt{Dropout}(\phi, p)$ 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 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.

	ShapeNet		Pascal	
Method	10-shot	15-shot	10-shot	15-shot
MAML [7]	29.555±0.600	22.286±3.369	2.612±0.280	2.513±0.250
MAML [7] + SiMT	**18.913±2.655**	**16.100±1.318**	**1.462±0.230**	**1.229±0.074**
ANIL [27]	39.915±0.665	38.202±1.388	6.600±0.360	6.517±0.420
ANIL [27] + SiMT	**37.424±0.951**	**29.478±0.212**	**5.339±0.321**	**5.007±0.145**
MetaSGD [22]	17.353±1.110	15.768±1.266	3.532±0.381	2.833±0.216
MetaSGD [22] + 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 mini-ImageNet, 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	
Model	Method	1-shot	5-shot	1-shot	5-shot
Conv4 [41]	MAML [7]	47.33±0.45	63.27±0.14	50.19±0.21	66.05±0.19
	MAML [7] + SiMT	**51.49±0.18**	**68.74±0.12**	**52.51±0.21**	**69.58±0.11**
	ANIL [27]	47.71±0.47	63.13±0.43	49.57±0.04	66.34±0.28
	ANIL [27] + SiMT	**50.81±0.56**	**67.99±0.19**	**51.66±0.26**	**68.88±0.08**
	MetaSGD [22]	50.66±0.18	65.55±0.54	52.48±1.22	71.06±0.20
	MetaSGD [22] + SiMT	**51.70±0.80**	**69.13±1.40**	**52.98±0.07**	**71.46±0.12**
	ProtoNet [33]	47.97±0.29	65.16±0.67	51.90±0.55	71.51±0.25
	ProtoNet [33] + SiMT	**51.25±0.55**	**68.71±0.35**	**53.25±0.27**	**72.69±0.27**
ResNet-12 [25]	MAML [7]	52.66±0.60	68.69±0.33	57.32±0.59	73.78±0.27
	MAML [7] + SiMT	**56.28±0.63**	**72.01±0.26**	**59.72±0.22**	**74.40±0.90**
	ANIL [27]	51.80±0.59	68.38±0.20	57.52±0.68	73.50±0.35
	ANIL [27] + SiMT	**54.44±0.27**	**69.98±0.66**	**58.18±0.31**	**75.59±0.50**
	MetaSGD [22]	54.95±0.11	70.65±0.43	58.97±0.89	76.37±0.11
	MetaSGD [22] + SiMT	**55.72±0.96**	**74.01±0.79**	**61.03±0.05**	**78.04±0.48**
	ProtoNet [33]	52.84±0.21	68.35±0.29	61.16±0.17	79.94±0.20
	ProtoNet [33] + 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 [24, 42]. For RL, we chose it based on the best average return during the training [7]. 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 the Appendix. 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 [10] and Pascal [48] 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 with mini-ImageNet and tiered-ImageNet. We consider CUB and Cars as cross-domain datasets. 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 →	
Problem	Method	CUB	Cars	CUB	Cars
1-shot	MAML [7]	39.50±0.91	32.87±0.20	42.32±0.69	36.62±0.12
	MAML [7] + SiMT	**42.32±0.62**	**33.73±0.63**	**44.33±0.43**	**37.21±0.35**
	ANIL [27]	37.30±0.89	31.28±1.03	42.29±0.33	36.27±0.58
	ANIL [27] + SiMT	**38.86±0.98**	**32.34±0.95**	**44.53±1.21**	**36.92±0.56**
	MetaSGD [22]	41.98±0.18	**34.52±0.56**	46.48±2.10	38.09±1.21
	MetaSGD [22] + SiMT	**43.50±0.89**	33.92±0.30	**46.62±0.41**	**38.69±0.26**
	ProtoNet [33]	41.22±0.81	32.79±0.61	47.75±0.56	37.59±0.80
	ProtoNet [33] + SiMT	**44.13±0.30**	**34.53±0.40**	**48.89±0.65**	**38.07±0.42**
5-shot	MAML [7]	56.17±0.92	44.56±0.79	65.00±0.89	51.08±0.28
	MAML [7] + SiMT	**59.22±0.39**	**46.59±0.21**	**67.58±0.61**	**51.88±0.52**
	ANIL [27]	53.42±0.97	41.65±0.67	62.48±0.85	50.50±1.18
	ANIL [27] + SiMT	**56.03±1.40**	**45.88±0.82**	**66.30±0.99**	**54.60±0.91**
	MetaSGD [22]	58.90±1.30	47.44±1.55	70.38±0.27	56.28±0.07
	MetaSGD [22] + SiMT	**65.07±1.89**	**49.86±0.84**	**73.93±0.42**	**57.97±1.34**
	ProtoNet [33]	57.87±0.77	48.06±1.10	74.35±0.93	57.23±0.25
	ProtoNet [33] + 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 [48, 10]. For the backbone meta-learning schemes we use gradient-based approaches, including MAML [7], ANIL [27], and MetaSGD [22]. For all methods, we train the convolutional neural network with 7 layers [48] and apply dropout regularization [35] 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., $\sum_{(x,y)\in\mathcal{Q}} l_{\mathsf{ce}}(f_\phi(x), y)$ where l_{ce} is the cross-entropy loss. We train the meta-model on mini-ImageNet [41] and tiered-ImageNet [28] datasets, following the prior works [23, 42]. 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 [33]. We train each method on Conv4 [41] and ResNet-12 [25], 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 the Appendix 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 [43] and Cars [17] datasets. Such tasks are known to be challenging, as there exists a large distribution shift between training and testing domains [13]. 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.

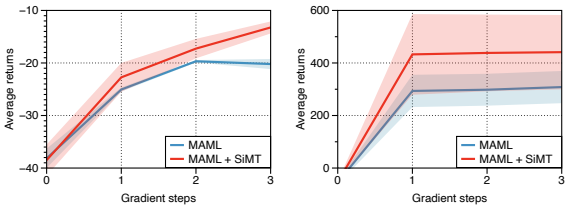
Table 4: Ablation study on each component of our proposed method. We report the few-shot in-domain adaptation (%) results on Conv4 trained with mini-ImageNet. We report the adaptation performance on the momentum network, except for the first experiment of the table. The reported results are averaged over three trials, subscripts denote the standard deviation, and bold denotes the best result.

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

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 [45], and trust-region policy optimization (TRPO; [29]) for the task-specific solver and meta-model, respectively, following MAML [7]. The overall training objective of meta-RL is in the Appendix, 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 [3] 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 [7]. 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 [35]. We provide more experimental details in the Appendix.

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 3 gradient updates. These results demonstrate that SiMT (red) could consistently improve the adaptation performance of MAML (blue). Also, SiMT makes faster performance improvements than vanilla MAML with additional gradient updates.



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

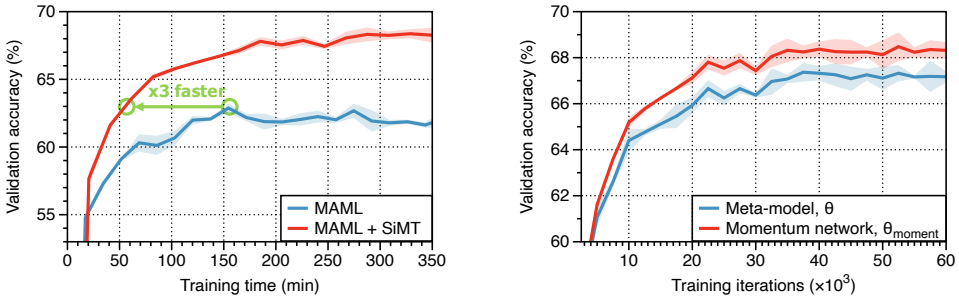
Figure 2: Meta-RL results for (a) 2D navigation and (b) Half-cheetah locomotion tasks. The solid line and shaded regions represent the truncated mean and standard deviation, respectively, across five runs.

Locomotion. To further demonstrate our method, we also study high-dimensional, complex locomotion tasks based on the MuJoCo [40] simulator. We choose a set of goal direction tasks with a planar cheetah ("Half-cheetah"), following previous works [7, 27]. 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 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.

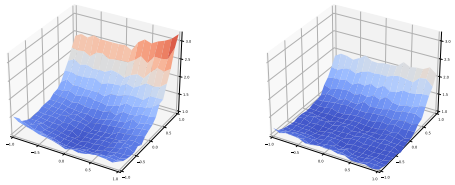
Component analysis. We perform an analysis on each component of our method in both 1-shot and 5-shot classification on mini-ImageNet: namely, the use of (a) the momentum network $\theta_{\texttt{moment}}$, (b) the distillation loss $\mathcal{L}_{\texttt{teach}}$ (6), and (c) the dropout regularization $\texttt{Dropout}(\cdot)$, by comparing the accuracies. The results in Table 4 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.



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

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

Loss landscape of the momentum network. 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 [20] (See the Appendix 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 [9].



(a) Meta-model (b) Momentum network

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

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 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.

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) [32], 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 [30]. As SiMT is built upon the meta-learning methods, practitioners may need to consider some computation for successful training. To address this issue, efficient meta-learning schemes [42] or lightweight methods for meta-learning [19] would be required for the applications.

References

[1] 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.

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

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

[4] 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.

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

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

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

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

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

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

[17] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

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

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

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

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

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

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

[40] 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.

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

[42] 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.

[43] 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.

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

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

[46] 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.

[47] H. Ye, L. Ming, D. Zhan, and W. Chao. Few-shot learning with a strong teacher. *CoRR*, abs/2107.00197, 2021.

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

[49] 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.

[50] 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 and instructions are in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and the Appendix 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 the Appendix 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]