FREE: Faster and Better Data-Free Meta-Learning

Supplementary Material

A. Implementation Details

The architecture for each pre-trained model and the meta-learner is based on Conv4. Common in meta-learning studies [1, 2, 7], Conv4 consists of four convolutional blocks, with each block comprising 32 3×3 filters, a BatchNorm layer, a ReLU function, and a 2×2 max-pooling layer. All pre-trained models are constructed to solve various 5-way tasks by randomly choosing 5 classes from the meta-training set (be it *CIFAR-FS*, *miniImageNet*, or *CUB*). With the Adam optimizer, these models are pre-trained using standard supervised learning at a learning rate of 0.01. In our experiments, we collect a set of 100 pre-trained models. For the task recovery, we generate 20 images for each class. The meta-generator parameters $\boldsymbol{\theta}_G$ and input $\boldsymbol{Z}$ are optimized simultaneously using the Adam optimizer, aiming to minimize Eq. (1) over k steps with a learning rate of 0.001. Similarly, the cloned meta-learner parameters $\widetilde{\boldsymbol{\theta}}$ are optimized by minimizing Eq. (3) with an inner loop learning rate set at 0.001. For the cross task replay, we employ MAML to execute meta-learning on tasks sampled in the memory bank. Following [4, 5], we use the Adam optimizer, set with inner and outer loop learning rates of 0.01 and 0.001, respectively.

In the multi-domain scenario, all pre-trained models are designed to address diverse tasks spanning several meta-training sets. We also take Conv4 as the architecture of pre-trained models and the meta-learner. Pre-trained models are designed for solving different 5-way tasks, which are devised by randomly sampling 5 classes from multiple meta-training sets, including *CIFAR-FS*, *miniImageNet*, and *CUB*. For meta-testing, we evenly construct 1800 meta-testing tasks across all meta-testing sets, including *CIFAR-FS*, *miniImageNet*, and *CUB*. All other configurations remain consistent with the primary results.

B. Architecture of the Generator

Tab. 1 lists the structure of the meta-generator in our proposed FIVE. The meta-generator takes the standard Gaussian noise as inputs and outputs the recovered data. Here, $d_{\boldsymbol{z}}$ is the dimension of Gaussian noise data $\boldsymbol{z}$, which is set as 256 in practice. The $negative\_slope$ of LeakyReLU is 0.2. We set image size as 32 for models pre-trained on *CIFAR-FS* and 84 for models pre-trained on *miniImageNet* and *CUB*. We set the number of channels nc as 3 for color image recovery and the number of convolutional filters nf as 64.

Table 1. Detailed structure of the meta-generator. We highlight the dimension change in red.

Notion	Description
$img\_size \times img\_size$	resolution of recovered image
bs	batch size
nc	number of channels of recovered image
nf	number of convolutional filters
FC($\cdot$)	fully connected layer;
BN($\cdot$)	batch normalization layer
Conv2D($input$, $output$, $filter\_size$, $stride$, $padding$)	convolutional layer

Structure	Dimension	
	Before	After
$\boldsymbol{z} \in \mathbb{R}_{d_{\boldsymbol{z}}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$	$[\,1, d_{\boldsymbol{z}}\,]$	$[\,bs, d_{\boldsymbol{z}}\,]$
FC($\boldsymbol{Z}$)	$[\,bs, d_{\boldsymbol{z}}\,]$	$[\,bs, 2 \times nf \times (img\_size//4) \times (img\_size//4)\,]$
Reshape	$[\,bs, 2 \times nf \times (img\_size//4) \times (img\_size//4)\,]$	$[\,bs, 2 \times nf, (img\_size//4), (img\_size//4)\,]$
BN	$[\,bs, 2 \times nf, (img\_size//4), (img\_size//4)\,]$	$[\,bs, 2 \times nf, (img\_size//4), (img\_size//4)\,]$
Upsampling	$[\,bs, 2 \times nf, (img\_size//4), (img\_size//4))\,]$	$[\,bs, 2 \times nf, (img\_size//2), (img\_size//2))\,]$
Conv2D($2 \times nf$, $2 \times nf$, 3, 1, 1)	$[\,bs, 2 \times nf, (img\_size//2), (img\_size//2))\,]$	$[\,bs, 2 \times nf, (img\_size//2), (img\_size//2))\,]$
BN, LeakyReLU	$[\,bs, 2 \times nf, (img\_size//2), (img\_size//2))\,]$	$[\,bs, 2 \times nf, (img\_size//2), (img\_size//2))\,]$
Upsampling	$[\,bs, 2 \times nf, (img\_size//2), (img\_size//2))\,]$	$[\,bs, 2 \times nf, img\_size, img\_size\,]$
Conv2D($2 \times nf$, nf, 3, 1, 1)	$[\,bs, 2 \times nf, img\_size, img\_size\,]$	$[\,bs, nf, img\_size, img\_size\,]$
BN, LeakyReLU	$[\,bs, nf, img\_size, img\_size\,]$	$[\,bs, nf, img\_size, img\_size\,]$
Conv2D(nf, nc, 3, 1, 1)	$[\,bs, nf, img\_size, img\_size\,]$	$[\,bs, nc, img\_size, img\_size\,]$
Sigmoid	$[\,bs, nc, img\_size, img\_size\,]$	$[\,bs, nc, img\_size, img\_size\,]$

C. Theoretical Proofs

Lemma 1. *If $\mathcal{L}_{KD}$ has Lipschitz Hessian, then:*

$$\begin{aligned}\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}}) = \nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta}) + O(\alpha^2)\\ - \alpha\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta}),\end{aligned}$$

where α is the step size of the inner loop.

Proof. Applying the fundamental theorem of Taylor's expansion to the gradient $\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})$, we have:

$$\begin{aligned}&\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})\\ =&\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta}) + \nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta})(\widetilde{\boldsymbol{\theta}}-\boldsymbol{\theta}) + \underbrace{O(\|\widetilde{\boldsymbol{\theta}}-\boldsymbol{\theta}\|^2)}_{=O(\alpha^2)}\\ =&\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta}) + \nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta}) \underbrace{(\widetilde{\boldsymbol{\theta}}-\boldsymbol{\theta})}_{-\alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta})} + O(\alpha^2)\\ =&\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta}) - \alpha\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta}) + O(\alpha^2).\end{aligned}$$

□

Theorem 1. *If $\mathcal{T}_i$ can be regarded as independent identically distributed samples from the distribution $\mathcal{P}_M$, then:*

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}}) = \nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta}) + O(\alpha^2)\\ - \alpha\nabla \underbrace{(\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta}))}_{Gradient\ Alignment},\end{aligned}$$

i.e.*, the inner product between gradients of different tasks.*

Proof. We have:

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})\\ =&\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})\frac{\partial\widetilde{\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}}\\ =&\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})\frac{\partial(\boldsymbol{\theta}-\alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta}))}{\partial\boldsymbol{\theta}}\\ =&\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})(\boldsymbol{I}-\alpha\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta}))\\ =&\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}}) - \alpha\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta}).\end{aligned}$$

Then, we can replace $\nabla_{\widetilde{\boldsymbol{\theta}}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})$ using Lemma 1. Concurrently, any term that is a high-order term in α is classified into the $O(\alpha^2)$ notation:

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\widetilde{\boldsymbol{\theta}})\\ =&\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta}) + O(\alpha^2)\\ &-\alpha\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta})\\ &-\alpha\nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta})\\ =&\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta}) + O(\alpha^2)\\ &-\alpha\nabla \underbrace{(\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_i;\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\mathcal{L}_{KD}(\mathcal{T}_j;\boldsymbol{\theta}))}_{Gradient\ Alignment}.\end{aligned}$$

□

D. Effects of the Knowledge Distillation

In Sec. 4.2, we propose to transfer the task-specific knowledge from the pre-trained model (acting as the teacher) to the meta-learner (acting as the student). Our experimental analysis (in Tab. 2) contrasts the utilization of hard labels in the meta-learning with soft labels in the knowledge distillation. We observe that knowledge distillation facilitates more nuanced supervision by capitalizing on the semantic class relationships inherent in the soft-label predictions provided by the teacher models, resulting in superior performance.

Table 2. Effects of the knowledge distillation.

Method	*CIFAR-FS*	
	5-way 1-shot	5-way 5-shot
Hard label	37.81 $\pm$ 0.75	50.35 $\pm$ 0.79
Ours	**39.13** $\pm$ **0.85**	**52.58** $\pm$ **0.77**

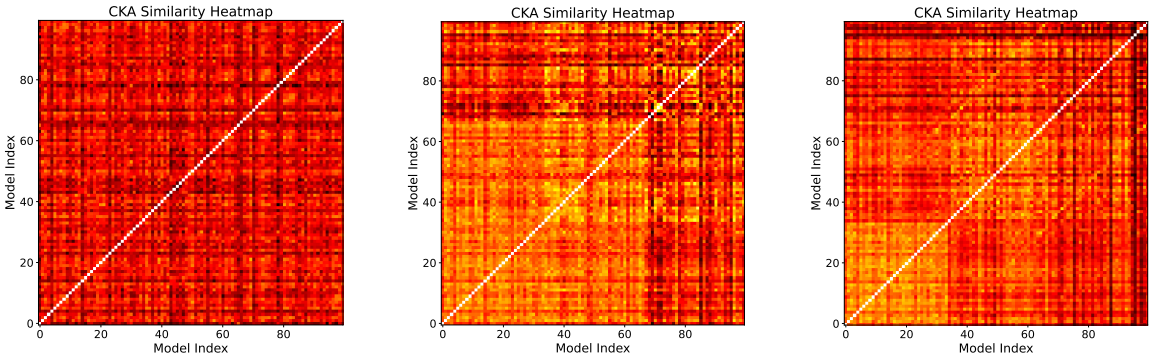
E. Discussions about Non-Inversion Methods

For data-free meta-learning, non-inversion methods attempt to solve the problem within the parameter space by integrating multiple pre-trained models. Nevertheless, they ignore the underlying data knowledge that can be extracted from these pre-trained models. Therefore, their performance is usually suboptimal and falls considerably short of inversion-based methods (*cf*. Tab. 2 in the main text). Moreover, their use cases are limited to situations where all pre-trained models share the same architecture, which does not address the problem of model heterogeneity—our key contribution.

By the way, it is difficult to compare their efficiency. Although non-inversion methods do not involve a model inversion process, they may require the learning of a black-box neural network to predict model parameters [8], thereby lacking a unified metric for time comparison.

F. Centered Kernel Alignment Analysis

To further illustrate the heterogeneity among pre-trained models, we employed Centered Kernel Alignment (CKA) [3, 6] to assess the features extracted by various collected pre-trained models. CKA, a measure of feature similarity, is particularly suited for multi-architecture comparisons because it can handle inputs of differing dimensions. The similarity heatmaps, as shown in Fig. 1, clearly indicate significant differences in the feature distribution of heterogeneous models. This CKA analysis suggests that directly applying meta-learning methods to recovered tasks without considering their heterogeneity may lead to suboptimal outcomes. Consequently, new strategies to align recovered tasks are necessary for data-free meta-learning.



(a) Multi-task scenario (b) Multi-domain scenario (c) Multi-architecture scenario

Figure 1. Similarity heatmaps of heterogeneous models measured by CKA. We compare (a) pre-trained models for distinct tasks on *CIFAR-FS*, (b) pre-trained models from datasets *CIFAR-FS*/*miniImageNet*/*CUB*, and (c) pre-trained models with architectures Conv4/ResNet-10/ResNet-18. Coordinate axes indicate the indices of corresponding pre-trained models.

References

[1] Jiaxin Chen, Li-Ming Zhan, Xiao-Ming Wu, and Fu-lai Chung. Variational metric scaling for metric-based meta-learning. In *AAAI*, pages 3478–3485, 2020. 1

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017. 1

[3] Zhiwei Hao, Jianyuan Guo, Kai Han, Yehui Tang, Han Hu, Yunhe Wang, and Chang Xu. One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation. In *NeurIPS*, 2023. 3

[4] Zixuan Hu, Li Shen, Zhenyi Wang, Tongliang Liu, Chun Yuan, and Dacheng Tao. Architecture, dataset and model-scale agnostic data-free meta-learning. In *CVPR*, pages 7736–7745, 2023. 1

[5] Zixuan Hu, Li Shen, Zhenyi Wang, Baoyuan Wu, Chun Yuan, and Dacheng Tao. Learning to learn from apis: Black-box data-free meta-learning. In *ICML*, 2023. 1

[6] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, pages 3519–3529, 2019. 3

[7] Chenghao Liu, Zhihao Wang, Doyen Sahoo, Yuan Fang, Kun Zhang, and Steven CH Hoi. Adaptive task sampling for meta-learning. In *ECCV*, pages 752–769, 2020. 1

[8] Zhenyi Wang, Xiaoyang Wang, Li Shen, Qiuling Suo, Kaiqiang Song, Dong Yu, Yan Shen, and Mingchen Gao. Meta-learning without data via wasserstein distributionally-robust model fusion. In *UAI*, pages 2045–2055, 2022. 3