Learning to Learn from APIs: Black-Box Data-Free Meta-Learning

Zixuan Hu [1]   **Li Shen** [2]   **Zhenyi Wang** [3]   **Baoyuan Wu** [4]   **Chun Yuan** [1]   **Dacheng Tao** [5]

Abstract

Data-free meta-learning (DFML) aims to enable efficient learning of new tasks by meta-learning from a collection of pre-trained models without access to the training data. Existing DFML work can only meta-learn from (i) white-box and (ii) small-scale pre-trained models (iii) with the same architecture, neglecting the more practical setting where the users only have inference access to the APIs with arbitrary model architectures and model scale inside. To solve this issue, we propose a **Bi**-level **D**ata-**f**ree **M**eta **K**nowledge **D**istillation (**BiDf-MKD**) framework to transfer more general meta knowledge from a collection of black-box APIs to one single meta model. Specifically, by just querying APIs, we inverse each API to recover its training data via a zero-order gradient estimator and then perform meta-learning via a novel bi-level meta knowledge distillation structure, in which we design a boundary query set recovery technique to recover a more informative query set near the decision boundary. In addition, to encourage better generalization within the setting of limited API budgets, we propose task memory replay to diversify the underlying task distribution by covering more interpolated tasks. Extensive experiments in various real-world scenarios show the superior performance of our BiDf-MKD framework.

1. Introduction

Data-free meta-learning (DFML) aims to meta-learn the useful prior knowledge from a collection of pre-trained models to enable efficient learning of new tasks without access to the training data due to privacy issues. Existing DFML work (Wang et al., 2022c) can only deal with the white-box pre-trained models, assuming access to the underlying model architecture and parameters. However, this assumption is not always satisfied. Recently, the concept of Model as A Service (MaaS) (Roman et al., 2009) comes to reality. Without access to the underlying models, users only have inference access to the corresponding APIs provided by the service providers like OpenAI or Google. For example, Cloud Vision API of Google, Amazon AI and Alibaba Cloud provide thousands of APIs designed for solving various specific tasks. TensorFlow Lite APIs provides numerous lightweight APIs deployed on mobiles, microcontrollers and edge devices. In this paper, we argue that these APIs can not only be the black-box tools for solving specific tasks, but can also serve as the training resources of meta-learning to enable efficient learning of new unseen tasks. The significance of doing this is to remove the need for large volumes of labeled data to perform meta-learning and reliably protect data privacy and security. This motivation leads to our valuable but challenging topic, i.e., black-box DFML, which aims to meta-learn the meta-initialization from a collection of black-box APIs without access to the training data and with only inference access, to enable efficient learning of new tasks without data privacy leakage.

The main challenges of black-box DFML lie in three aspects: (i) *data-free*: we have no access to the original training data of each API; (ii) *black-box*: we have no prior knowledge of the underlying model architecture and parameters inside each API; (iii) *model-agnostic*: each API may correspond to arbitrary underlying model architectures and model scale. Existing DFML work (Wang et al., 2022c) only tries to handle the first challenge, which can not meta-learn from black-box APIs with arbitrary underlying model architecture and scale. Concretely, Wang et al. (2022c) propose to meta-learn a neural network to predict the meta-initialization given a collection of white-box pre-trained models. However, this method requires the exact parameter of each pre-trained model, and it requires all pre-trained models share the same architecture. Besides, it can not scale to large-scale

---

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China [2]JD Explore Academy, Beijing, China [3]Department of Computer Science and Engineering, University at Buffalo, NY, USA [4]School of Data Science, the Chinese University of Hong Kong, Shenzhen, China [5]School of Computer Science, the University of Sydney, Sydney, Australia. Correspondence to: Li Shen <mathshenli@gmail.com>, Zhenyi Wang <zhenyiwa@buffalo.edu>, Chun Yuan <yuanc@sz.tsinghua.edu.cn>.

pre-trained models because it directly uses a hyper neural network to output all parameters of the meta-initialization.

In this work, we solve all these issues in a unified framework (see Fig. 2). We propose a novel **Bi**-level **D**ata-free **M**eta **K**nowledge **D**istillation (**BiDf-MKD**) framework to transfer more general meta knowledge from a collection of black-box APIs to one single meta model, which serves as the meta-initialization to initialize the task-specific models of new unseen tasks. Specifically, by just querying the API, we first "inverse" each API to recover the label-conditional data starting from latent standard Gaussian noise via a zero-order gradient estimator. With the aid of the recovered data, we then perform meta-learning by transferring the general meta knowledge from a collection of black-box APIs to the meta model through our proposed bi-level meta knowledge distillation structure. Then, we formally define the *knowledge vanish* (see Definition 4.1) issue involved in the bi-level characteristic of meta-learning from the perspective of information theory, i.e., the outer-level optimization of meta-learning could be useless. We argue that the knowledge vanish issue in the data-free setting is more significant than in the data-based setting due to the relatively low diversity of recovered data. As illustrated in Fig. 3, to alleviate such issue, we design a boundary query set recovery technique to amplify the diversity by recovering a more informative query set near the decision boundary. In addition, to encourage better generalization to the unseen tasks within the setting of limited API budgets, we propose task memory replay on more interpolated tasks. The interpolated tasks do not correspond to any API so that we can diversify the underlying task distribution associated with the given APIs by covering more new tasks. Overall, our proposed framework can effectively solve the black-box DFML problem (i) without the need for real data, (ii) with only inference access to the APIs, (iii) regardless of the underlying model architecture and model scale inside each API, and (iv) without data privacy leakage. Thus, it substantially expands the real-world application scenarios of black-box DFML.

We perform extensive experiments in three real-world black-box scenarios (see Fig. 1), including (i) **API-SS**. All APIs are designed for solving tasks from the **S**ame meta training subset with the **S**ame architecture inside. (ii) **API-SH**. All APIs are designed for solving tasks from the **S**ame meta training subset but with **H**eterogeneous architectures inside. (iii) **API-MH**. All APIs are designed for solving tasks from **M**ultiple meta training subsets with **H**eterogeneous architectures inside. For benchmarks of three scenarios on CIFAR-FS, MiniImageNet and CUB, our framework achieves significant performance gains in the range of 8.09% to 21.46%. We summarize the main contributions as three-fold:

- For the first time, we propose a new practical and valuable setting of DFML, i.e., black-box DFML, whose goal is to meta-learn the meta-initialization from a collection of black-box APIs without access to the original training data and with only inference access, to enable efficient learning of new tasks without privacy leakage.

- We propose BiDf-MKD to meta-learn the meta-initialization by transferring general meta knowledge from a collection of black-box APIs to one single model. We formally define the knowledge vanish issue of meta-learning and design the boundary query set recovery technique to alleviate it. We also propose task memory replay to boost the generalization ability for the setting of limited API budgets.

- We propose three real-world black-box scenarios (API-SS, API-SH, and API-MH) for a complete and practical evaluation of black-box DFML. We are the first to propose a data-free, inference-based and model-agnostic framework, simultaneously applicable to all three scenarios without any change and outperforming the SOTA baselines by a large margin.

2. Related Work

Meta-learning & Data-free meta-learning. Meta-learning (Schmidhuber, 1987), a.k.a. *learning to learn*, aims to meta-learn useful prior knowledge from a collection of tasks, which can be generalized to new unseen tasks efficiently. MAML (Finn et al., 2017) and its variants (Abbas et al., 2022; Jeong & Kim, 2020; Raghu et al., 2019; Behl et al., 2019; Rajeswaran et al., 2019) meta-learn a sensitive meta-initialization to initialize the task-specific model, while other works (Santoro et al., 2016; Mishra et al., 2017; Garnelo et al., 2018; Munkhdalai & Yu, 2017) meta-learn a hyper neural network to output the task-specific model parameters conditioned on the support set. Existing meta-learning works (Vinyals et al., 2016; Wang et al., 2021; Finn et al., 2018; Yao et al., 2021; Wang et al., 2022b; Harrison et al., 2020; Ye et al., 2020a; Li et al., 2020; Yang et al., 2021; Simon et al., 2022; Liu et al., 2019; Wang et al., 2022a; 2020; Zhou et al., 2021) assume the access to the training data associated with each task. More recently, Wang et al. (2022c) propose a new meta-learning paradigm, data-free meta-learning, which aims to meta-learn the meta-initialization from a collection of pre-trained modes without access to their training data. However, it imposes strict restrictions on pre-trained models: (i) white-box, (ii) small-scale, and (iii) with the same architecture, thus reducing its applicable scenarios in real applications.

Knowledge distillation for meta-learning. Our work is reminiscent of knowledge distillation (KD) (Hinton et al., 2015) as we leverage a collection of APIs to supervise the meta-learning. Below, we first briefly review KD and compare ours with existing KD works for meta-learning. KD
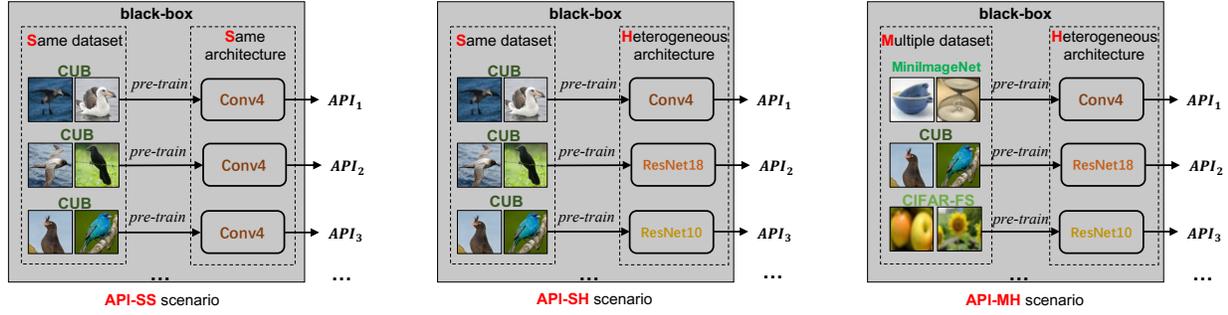
Figure 1. According to the datasets and model architectures inside the APIs, we propose three real-world black-box scenarios for a complete and practical evaluation of black-box DFML. We are the first to propose a unified framework simultaneously applicable to all three scenarios without any change, thus greatly expanding the real-world application scope of black-box DFML.

aims to supervise the training process of the student model with the knowledge of the teacher model. The knowledge can be the soft-label predictions (Hinton et al., 2015), hidden layer activation (Romero et al., 2014; Koratana et al., 2019), embedding (Chen et al., 2018; Ahn et al., 2019), or relationship (Ye et al., 2020b). Existing meta-learning works based on KD differ greatly from ours in motivation, setting and manner. Ye et al. (2022) associate each task with an additional teacher classifier to provide additional supervision for meta-learning. They conduct KD by minimizing the prediction disagreement on real data, thus not applicable to the data-free setting. Besides, it relies on separate training with real data to obtain the teacher classifiers. REFILLED (Ye et al., 2020b) performs KD between one teacher model and one student model in different label spaces. It relies on the relationship among embeddings, which are unavailable in our black-box setting.

Model inversion. Model inversion (Fredrikson et al., 2015; Wu et al., 2016; Zhang et al., 2020) aims to recover the training data from the pre-trained model. Our framework also involves recovering data from black-box APIs to transfer meta knowledge. Existing techniques (Fredrikson et al., 2015; Wu et al., 2016; Zhang et al., 2020; Deng & Zhang, 2021; Lopes et al., 2017; Chawla et al., 2021; Zhu et al., 2021; Liu et al., 2021; Zhang et al., 2022b) about model inversion from white-box pre-trained models are not applicable to our black-box setting. Recent works DFME (Truong et al., 2021) and MAZE (Kariyappa et al., 2021) leverage the black-box model inversion technique to perform model extraction. Key differences include our meta-learning objective and loss formulation for label-conditional data recovery. We also design a novel boundary data recovery technique to recover more informative data near the decision boundary (see Fig. 3).

3. Problem Setup

In this section, we first clarify the definition of black-box DFML, followed by its meta testing procedure.

3.1. Black-box DFML Setup

We are given a collection of APIs $\{A_i\}$ solving different tasks, with only inference access and without accessing their original training data. We aim to meta-learn the meta-initialization $\boldsymbol{\theta}$, which can be adapted fast to new unseen tasks $\{\mathcal{T}_i^{new}\}$. Note that each API may correspond to arbitrary underlying model architecture and model scale.

3.2. Meta Testing

During meta testing, several unseen N-way K-shot tasks $\{\mathcal{T}_i^{new} = \{\boldsymbol{S}_i^{new}, \boldsymbol{Q}_i^{new}\}\}$ arrive together. The classes appearing in meta testing tasks are unseen during meta training. Each task contains a support set $\boldsymbol{S}_i^{new}$ with N classes and K instances per class. We use the support set $\boldsymbol{S}_i^{new}$ to adapt the meta initialization to the task-specific task (i.e., $\boldsymbol{\theta} \to \boldsymbol{\theta}_i^{new}$). The query set $\boldsymbol{Q}_i^{new}$ is what we actually need to predict. The final accuracy is measured by the average accuracy for those meta testing tasks.

4. Methodology

In this section, we propose a unified framework (Fig. 2) to solve the black-box DFML problem, including (i) BiDf-MKD to transfer meta knowledge (Sec. 4.1) and (ii) task memory replay to boost generalization ability (Sec. 4.2).

4.1. Bi-level data-free meta knowledge distillation (BiDf-MKD)

Task recovery via API inversion. For the API A_i, we aim to recover the task-specific training data set $\hat{\boldsymbol{X}}$, with which general meta knowledge can be transferred from the API $A_i(\cdot)$ to the meta model $F(\cdot; \boldsymbol{\theta})$. As illustrated in Fig. 2, it consists of four components: a generator $G(\cdot; \boldsymbol{\theta}_G)$, an API A_i, a gradient estimator and a memory bank $\mathcal{B}$. The generator $G(\cdot; \boldsymbol{\theta}_G)$ takes the standard Gaussian noise $\boldsymbol{z}$ as input and outputs the recovered data $\hat{\boldsymbol{x}} = G(\boldsymbol{z}; \boldsymbol{\theta}_G)$. To make the recovery process label-conditional, we update the
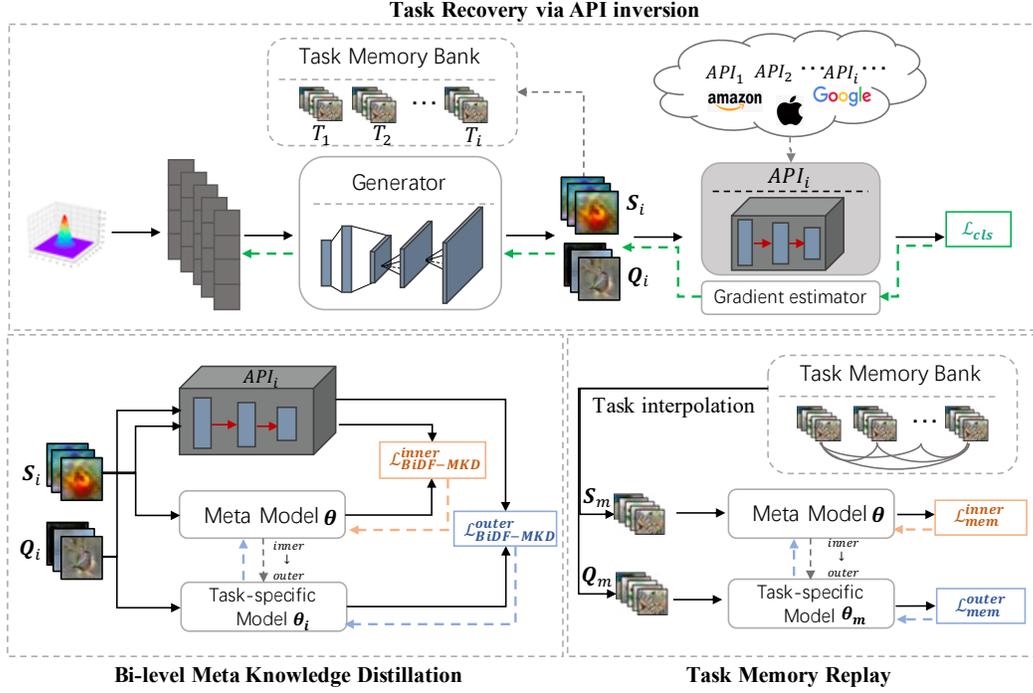
Figure 2. The whole pipeline of our proposed BiDf-MKD framework. For each API A_i, we recover its training data starting from the random standard Gaussian noise $\boldsymbol{Z}_i$. By continually querying the black-box API A_i, we gradually update the noise to label-conditional data. We then split the recovered data into the support set $\boldsymbol{S}_i$ and query set $\boldsymbol{Q}_i$ to perform meta-learning via our bi-level meta knowledge distillation structure. Alternatively, we can perform task memory replay with MAML over more interpolated tasks.

corresponding $\boldsymbol{z}$ and $\boldsymbol{\theta}_G$ simultaneously by minimizing the per datum cross-entropy loss

$$\min_{\boldsymbol{z}, \boldsymbol{\theta}_G} \ell_{cls}(\hat{\boldsymbol{x}}, y) = CE(A_i(\hat{\boldsymbol{x}}), y), \quad \text{s.t.} \quad \hat{\boldsymbol{x}} = \mathrm{G}(\boldsymbol{z}; \boldsymbol{\theta}_G), \quad (1)$$

where y is the pre-defined class label. To recover a batch of data $\hat{\boldsymbol{X}}$ of class labels $\boldsymbol{Y}$, we update $\boldsymbol{Z}$ and $\boldsymbol{\theta}_G$ by minimizing the batch-wise loss

$$\min_{\boldsymbol{Z}, \boldsymbol{\theta}_G} \mathcal{L}_{cls}(\hat{\boldsymbol{X}}) = \frac{1}{|\hat{\boldsymbol{X}}|} \sum_{(\hat{\boldsymbol{x}}, y) \in (\hat{\boldsymbol{X}}, \boldsymbol{Y})} l_{cls}(\hat{\boldsymbol{x}}, y),$$
$$\text{s.t.} \quad \hat{\boldsymbol{X}} = \mathrm{G}(\boldsymbol{Z}; \boldsymbol{\theta}_G). \quad (2)$$

After obtaining a certain number of data, we feed these recovered $\hat{\boldsymbol{X}}$ into the memory bank $\mathcal{B}$, i.e., a first-in-first-out (FIFO) container with a certain volume.

Zero-order gradient estimation. Recall that our objective of task recovery is to update $\boldsymbol{z} \in \boldsymbol{Z}$ and $\boldsymbol{\theta}_G$ simultaneously by minimizing $\mathcal{L}_{cls}$.

$$\boldsymbol{\theta}_G^{t+1} = \boldsymbol{\theta}_G^t - \eta \nabla_{\boldsymbol{\theta}_G} \mathcal{L}_{cls} \quad (3a)$$
$$\boldsymbol{z}^{t+1} = \boldsymbol{z}^t - \eta \nabla_{\boldsymbol{z}} \mathcal{L}_{cls}. \quad (3b)$$

Updating $\boldsymbol{z}$ and $\boldsymbol{\theta}_G$ in such way involves calculating $\nabla_{\boldsymbol{\theta}_G} \mathcal{L}_{cls}$ and $\nabla_{\boldsymbol{z}} \mathcal{L}_{cls}$. With the use of the chain rule, we

decompose each gradient into two components:

$$\nabla_{\boldsymbol{\theta}_G} \mathcal{L}_{cls} = \frac{\partial \mathcal{L}_{cls}}{\partial \boldsymbol{\theta}_G} = \frac{1}{|\hat{\boldsymbol{X}}|} \sum_{\hat{\boldsymbol{x}} \in \hat{\boldsymbol{X}}} \left[\frac{\partial \ell_{cls}}{\partial \hat{\boldsymbol{x}}} \times \frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{\theta}_G} \right] \quad (4a)$$

$$\nabla_{\boldsymbol{z}} \mathcal{L}_{cls} = \frac{\partial \mathcal{L}_{cls}}{\partial \boldsymbol{z}} = \frac{\partial \ell_{cls}}{\partial \hat{\boldsymbol{x}}} \times \frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{z}}. \quad (4b)$$

The second factors ($\frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{\theta}_G}$ and $\frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{z}}$) in Eq. (4a) and Eq. (4b) can be automatically calculated via the automatic differentiation mechanism in PyTorch (Paszke et al., 2017) or Tensor-Flow (Abadi et al., 2015). However, it is not applicable for calculating the first factor ($\frac{\partial \ell_{cls}}{\partial \hat{\boldsymbol{x}}}$), because we have no access to the underlying model parameters inside the API. To this end, we adopt a zero-order gradient estimator to obtain an approximation of the first-order gradient by just querying the API. To explain how the first-order gradient ($\frac{\partial \ell_{cls}}{\partial \hat{\boldsymbol{x}}}$) is estimated, consider the random noise vector $\boldsymbol{z} \in \mathbb{R}^{d_{\boldsymbol{z}}}$, which yields the recovered data $G(\boldsymbol{z}; \boldsymbol{\theta}_G) = \hat{\boldsymbol{x}} \in \mathbb{R}^{d_{\hat{\boldsymbol{x}}}}$ with label y and the loss value $\ell_{cls}(\hat{\boldsymbol{x}}, y) \in \mathbb{R}$. We can feed the recovered data plus a set of random direction vectors to query the API and estimate the gradient according to the difference of two loss values. This leads to the randomized gradient estimation (Liu et al., 2020b):

$$\hat{\nabla}_{\hat{\boldsymbol{x}}} \ell_{cls} = \frac{1}{q} \sum_{i=1}^{q} \left[\frac{d_{\hat{\boldsymbol{x}}}}{\mu} \left(\ell_{cls} \left(\hat{\boldsymbol{x}} + \mu \boldsymbol{u}_i, y \right) - \ell_{cls}(\hat{\boldsymbol{x}}, y) \right) \boldsymbol{u}_i \right], \quad (5)$$

where $\{\boldsymbol{u}_i\}_{i=1}^{q}$ are q random direction vectors sampled in-

dependently and uniformly from the sphere of a unit ball. $\mu > 0$, a.k.a the smoothing parameter, is a given small step size. The estimation $\hat{\nabla}_{\hat{x}}\ell_{cls}$ is reasonable because Eq. (5) provides an unbiased estimate of the first-order gradient $\nabla_{\hat{x}}\ell_{cls}$ of the Gaussian smoothing version (Gao et al., 2018; Zhang et al., 2022c). With the zero-order gradient estimator, we can obtain the estimated gradient according to Eq. (4):

$$\nabla_{\boldsymbol{\theta_G}}\mathcal{L}_{cls} \approx \frac{1}{|\hat{\boldsymbol{X}}|}\sum_{\hat{\boldsymbol{x}} \in \hat{\boldsymbol{X}}}\left[\hat{\nabla}_{\hat{\boldsymbol{x}}}\ell_{cls} \times \frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{\theta}_G}\right] \quad (6a)$$

$$\nabla_{\boldsymbol{z}}\mathcal{L}_{cls} \approx \hat{\nabla}_{\hat{\boldsymbol{x}}}\ell_{cls} \times \frac{\partial \hat{\boldsymbol{x}}}{\partial \boldsymbol{z}}, \quad (6b)$$

where $\frac{\partial \hat{x}}{\partial \theta_G} \in \mathbb{R}^{d_{\hat{x}} \times d_{\theta_G}}$ and $\frac{\partial \hat{x}}{\partial z} \in \mathbb{R}^{d_{\hat{x}} \times d_z}$ are the Jacobian matrices and $\hat{\nabla}_{\hat{x}}\ell_{cls} \in \mathbb{R}^{1 \times d_{\hat{x}}}$ is the zero-order gradient estimation. Then we can perform gradient descent by updating $\boldsymbol{Z}$ and $\boldsymbol{\theta}_G$ according to Eq. (3) to produce the recovered data required to perform meta knowledge distillation.

Bi-level meta knowledge distillation for meta-learning. We propose a bi-level meta knowledge distillation structure to perform meta-learning by transferring general meta knowledge from a collection of black-box APIs into one single meta model with the recovered data. Different from the common knowledge distillation methods requiring the teacher and student designed for the same task, the meta model is not tailored to any specific task. Thus, it is not appropriate to directly transfer the task-specific knowledge from the API (viewed as the teacher) to the meta model (viewed as the student). To this end, our bi-level structure controls the knowledge flow from each API to the meta model via an intermediate task-specific model, which transfers more general meta knowledge. The meta knowledge enables fast knowledge distillation of the inner loop, thus not task-specific and approximate to be transferred to the meta model.

Our proposed BiDf-MKD involves a bi-level structure, i.e., the inner level and the outer level. For API A_i, we split its recovered data $\hat{\boldsymbol{X}}_i$ into two non-overlap support set $\boldsymbol{S}_i$ and query set $\boldsymbol{Q}_i$. For the inner level, we transfer the task-specific knowledge from A_i to a task-specific model $F(\cdot; \boldsymbol{\theta}_i)$ initialized by $\boldsymbol{\theta}$ so that the task-specific model $F(\cdot; \boldsymbol{\theta}_i)$ can act like A_i on $\boldsymbol{S}_i$. We clone this task-specific model $F(\cdot; \boldsymbol{\theta}_i)$ from API A_i by minimizing the disagreement of predictions between them:

$$\boldsymbol{\theta}_i = \min_{\theta}\mathcal{L}_{BiDf-MKD}^{inner} \triangleq \min_{\theta}\sum_{\hat{\boldsymbol{x}} \in \boldsymbol{S}_i}\ell_{KL}(F(\hat{\boldsymbol{x}}; \boldsymbol{\theta}), A_i(\hat{\boldsymbol{x}})), \quad (7)$$

where $\ell_{KL}(p, q)$ measures the Kullback–Leibler (KL) divergence (MacKay et al., 2003) between distributions p and q. One can apply other measures that can characterize the difference of distributions. The objective of the inner level is only to transfer the task-specific knowledge from API A_i to the task-specific model $F(\cdot; \boldsymbol{\theta}_i)$. Note that the task-specific knowledge is not desired for the meta model,
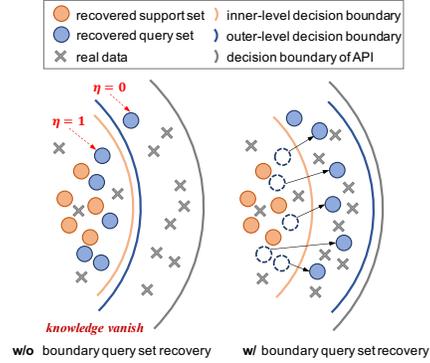
Figure 3. Knowledge vanish issue of meta-learning occurs when the outer-level optimization can be ignored.

because the meta model should possess the potential to work well with all tasks after adaptation and one certain task-specific knowledge can not directly adapt to other different tasks. Then, with the aid of the inner level, we resort to the outer level to explore more general meta knowledge, which is beneficial to the meta model.

The core idea of the outer level is to explore more general meta knowledge, with which we can make the best use of the task-specific knowledge in the inner level. In other words, it is hard to obtain an excellent task-specific model relying solely on the task-specific knowledge in the inner level with few recovered data. We desire more general meta knowledge to facilitate the inner-level knowledge distillation so as to narrow the gap between the task-specific model $F(\cdot; \boldsymbol{\theta}_i)$ and the API A_i as much as possible. The gap is evaluated by testing the task-specific model $F(\cdot; \boldsymbol{\theta}_i)$ on a wider range of hold-out data $\boldsymbol{Q}_i$, which is equivalent to minimizing

$$\min_{\boldsymbol{\theta}}\mathcal{L}_{BiDf-MKD}^{outer}(\boldsymbol{\theta}) = \sum_{\hat{\boldsymbol{x}} \in \boldsymbol{Q}_i}\ell_{KL}(F(\hat{\boldsymbol{x}}; \boldsymbol{\theta}_i), A_i(\hat{\boldsymbol{x}})),$$
$$\text{s.t. } \boldsymbol{\theta}_i = \min_{\theta}\mathcal{L}_{BiDf-MKD}^{inner}, \quad (8)$$

where $\boldsymbol{\theta}_i$ is obtained after the inner level following Eq. (7) and $\boldsymbol{\theta}$ is the meta model parameters we truly update. $F(\hat{\boldsymbol{x}}; \boldsymbol{\theta}_i)$ and $A_i(\hat{\boldsymbol{x}})$ outputs the prediction (after softmax) on $\hat{\boldsymbol{x}}$ from the task-specific model and API, respectively.

Boundary query set recovery for knowledge vanish. The ideal BiDf-MKD is conducted in such a way that narrows the gap between the task-specific model and the API on the hold-out query set. However, the outer-level knowledge distillation could be too "lazy" to explore the meta knowledge. This issue is more significant in the data-free meta-learning setting than in the data-based meta-learning setting because of the relatively low diversity of the recovered data. Consider an extreme example with $\boldsymbol{S}_i = \boldsymbol{Q}_i$. The task-specific model distilled from the API on $\boldsymbol{S}_i$ in the inner level can perform perfectly on $\boldsymbol{Q}_i$ in the outer level. This could lead to an illusion where the task-specific model is so "perfect"

that we can explore no more knowledge to facilitate the inner-level knowledge distillation. We then formally define the *complete knowledge vanish* issue of meta-learning from the perspective of information theory.

Definition 4.1. The *complete knowledge vanish* of meta-learning occurs when the outer-level optimization can be ignored, namely the mutual information $I(\theta; Q_i|\theta_i, S_i) = 0$ (or $H(\theta|\theta_i, S_i) = H(\theta|\theta_i, S_i, Q_i)$).

Refer to App. A for the relation between the mutual information I and entropy H. Explicitly maximizing $I(\theta; Q_i|\theta_i)$ requires an unknown posterior distribution over θ. Instead, we implicitly encourage $H(\theta|\theta_i, S_i) > H(\theta|\theta_i, S_i, Q_i)$ by recovering Q_i with more information. Zhang et al. (2022a) point out the samples near the decision boundary contained more valuable information for classification. This motivation leads to our proposed boundary query set recovery technique, which urges the generator to recover the query set between the decision boundaries of the task-specific model $F(\cdot; \theta_i)$ and API $A_i(\cdot)$ (see Fig. 3). Specifically, we first recover the support set S_i (orange circles) by minimizing Eq. (2). Then, we use S_i to conduct the inner-level knowledge distillation following Eq. (7) to distill the task-specific model parameters θ_i with the inner-level decision boundary (orange arc). To recover a more informative query set (blue circles) for the outer-level knowledge distillation, we incorporate $F(\cdot|\theta_i)$ to query set recovery by maximizing the disagreement between $F(\cdot|\theta_i)$ and the API $A_i(\cdot)$ following Eq. (9). Note that large disagreement may guide to generate just some outliers. Therefore, we only pay more attention to those boundary samples and introduce the loss for boundary query set recovery:

$$\min_{z, \theta_G} \ell_Q(\hat{x}, y)$$
$$= CE(A_i(\hat{x}), y) - \lambda_Q \cdot \eta \cdot \ell_{KL}(F(\hat{x}; \theta_i), A_i(\hat{x})),$$
$$\text{s.t. } \hat{x} = G(z; \theta_G),$$
$$\eta = \mathbb{I}\{\arg\max F(\hat{x}; \theta_i) = \arg\max A_i(\hat{x})\}.$$
(9)

The function $\mathbb{I}(\cdot)$ is an indicator to enable $\hat{x}$ with the same prediction from the API and the task-specific model ($\eta = 1$), otherwise disable it ($\eta = 0$). Unlike the loss Eq. (1), the loss Eq. (9) guides to recover $\hat{x}$ between the decision boundaries of the task-specific model $F(\cdot; \theta_i)$ (orange arc) and API $A_i(\cdot)$ (grey arc), which provides more information for the outer-level knowledge distillation.

4.2. Task memory replay

The basic BiDf-MKD aims to transfer the meta knowledge of a collection of APIs to one single meta model. A small number of APIs (e.g., 100 APIs) are insufficient to represent the actual underlying task distributions and makes it easy to overfit, leading to poor generalization ability for the new unseen tasks. To make our method work well within the setting of limited API budgets, we propose task memory

replay to diversify the underlying task distribution by covering more interpolated tasks. We design a memory bank with the first-in-first-out structure to store the previous recovered task data from each API. We then generate new tasks that interpolate between the previous tasks.

Suppose the memory bank $\mathcal{B}$ has stored the recovered task data $\{S_i, Q_i\}_{i=0}^T$ recovered from the APIs $\{A_i\}_{i=0}^T$. Each task corresponds to a different label space. We generate a new task with a new label space by randomly resampling the class labels and the corresponding support set (S_m, Y_{S_m}) and query set (Q_m, Y_{Q_m}) from the memory bank. These new interpolated tasks do not correspond to any given API and thus diversify the task distribution, leading to better generalization to unseen tasks. For these interpolated tasks, we adopt MAML (Finn et al., 2017), consistent with the bi-level structure of BiDf-MKD, to update the meta model by minimizing

$$\min_{\theta} \mathcal{L}_{mem}^{outer} = \mathcal{L}_{cls}(F(Q_m; \theta_m), Y_{Q_m}),$$
$$\text{s.t. } \theta_m = \min_{\theta} \mathcal{L}_{mem}^{inner} \triangleq \min_{\theta} \mathcal{L}_{cls}(F(S_m; \theta), Y_{S_m}).$$
(10)

The moment to perform task memory replay is flexible. For example, each API may come in a sequential way and we can perform task memory replay at the interval of two adjacent APIs or in exceptional cases where network interruption happens. Task memory replay does not need to query the APIs online, thus requiring no network connection and making our framework more stable. Overall, we integrate BiDf-MKD and task memory replay in an end-to-end manner, which is summarized in Alg. 1 of App. D.

5. Experiments

We verify the effectiveness of our proposed BiDf-MKD framework in various real-world scenarios (API-SS, API-SH, and API-MH) with comprehensive ablation studies.

5.1. Experimental Setup

Baselines. (i) Random. Randomly parameterize the meta-initialization for meta testing. **(ii) Best-API.** We select the API with the highest reported accuracy to directly predict the query set during meta testing. **(iii) Single-DFKD.** Single-level data-free knowledge distillation. Update the meta model in a sequential manner, with only single-level data-free knowledge distillation (Eq. (7)). This baseline only transfers the task-specific knowledge sequentially instead of meta knowledge from APIs to meta model. **(iv) Distill-Avg.** We perform single-level data-free knowledge distillation (Eq. (7)) for each API to obtain a surrogate white-box model. We average all surrogate model parameters layer-wise as the meta-initialization. **(v) White-box DFML.** Perform our BiDf-MKD in an ideal white-box setting, where the actual first-order (FO) gradients of the parameters inside the APIs
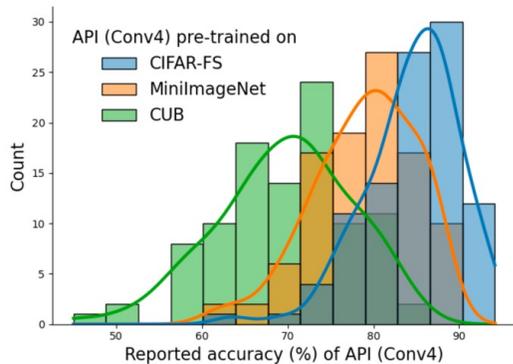
Figure 4. Histogram of the reported accuracy of APIs.

are available for task recovery. This baseline provides an upper bound of performance for black-box DFML.

API quality. Fig. 4 shows the histogram of the reported accuracy of APIs (Conv4) pre-trained on CIFAR-FS, Mini-ImageNet and CUB, respectively. We provide more statistical results of APIs with other architectures (ResNet10 and ResNet18) in App. C.1. We argue that our framework should work well in the real-world scenario, where some APIs may be with relatively low accuracy. Refer to App. C.1 for more discussions on robustness against the API quality variation.

Implementation details & Datasets. We evaluate our BiDf-MKD framework on the meta-testing subsets of **CIFAR-FS** (Bertinetto et al., 2018), **MiniImageNet** (Vinyals et al., 2016), and **CUB-200-2011** (CUB) (Wah et al., 2011). Refer to App. E.2 and App. E.1 for detailed dataset setup and implementation details for three scenarios.

5.2. Experiments of black-box DFML in API-SS

Overview. We first perform experiments in **API-SS** scenario, where all APIs are designed for solving different 5-way tasks from the same meta training subset (CIFAR-FS, MiniImageNet or CUB) with the same architecture (Conv4).

Results. Tab. 1 shows the results for 5-way classification in API-SS scenario. For 1-shot learning, ours outperforms the best baselines by 11.24%, 8.15% and 8.09% on three datasets, respectively. For 5-shot learning, ours outperforms the best baselines by 20.02%, 18, 23% and 21.46% on three datasets, respectively. The results show that simply parameterizing the meta-initialization does not work for meta-learning. We observe a significant performance reduction of the best API from about 90% to about 20% because of the non-overlapping label space between the best API and the meta testing tasks and we can not fine-tune the API with the support set of new unseen tasks because of the black-box setting. The single-DFKD simply transfers the task-specific knowledge to the meta model in a sequential way; its bad

performance reveals the accumulated task-specific knowledge is not beneficial to the efficient learning of new unseen tasks. Distill-Avg fuse all surrogate white-box models layer-wise and then fine-tune the meta-initialization; it also dose not perform well because those surrogate models train on different tasks, thus lacking precise correspondence among them. Ours performs the best because of the strong generalization ability of our transferred meta knowledge, which enables the efficient learning of new unseen tasks.

Table 1. Compare to baselines in API-SS scenario.

API-SS	Method	1-shot	5-shot
CIFAR-FS 5-way	Random	20.35 ± 0.42	20.59 ± 0.45
	Best-API	19.04 ± 0.68	19.04 ± 0.67
	Single-DFKD	20.04 ± 0.63	20.14 ± 0.64
	Distill-Avg	24.24 ± 0.46	27.56 ± 0.51
	Ours	**35.48 ± 0.67**	**47.58 ± 0.74**
MiniImageNet 5-way	Random	21.20 ± 0.38	21.13 ± 0.37
	Best-API	20.51 ± 0.63	20.39 ± 0.62
	Single-DFKD	20.03 ± 0.60	20.14 ± 0.66
	Distill-Avg	20.53 ± 0.20	21.24 ± 0.24
	Ours	**29.35 ± 0.60**	**39.47 ± 0.64**
CUB 5-way	Random	21.09 ± 0.38	21.11 ± 0.37
	Best-API	19.99 ± 0.69	19.95 ± 0.70
	Single-DFKD	19.56 ± 0.64	20.06 ± 0.64
	Distill-Avg	21.07 ± 0.25	21.97 ± 0.30
	Ours	**29.10 ± 0.64**	**43.43 ± 0.66**

5.3. Experiments of black-box DFML in SH

Overview. We then perform experiments in a more realistic scenario, **API-SH**, where all APIs are designed for solving different tasks from the same meta training subset (CIFAR-FS, MiniImageNet or CUB) but with heterogeneous architectures (Conv4, ResNet10 and ResNet18).

Results. Tab. 2 shows the result for 5-way classification in API-SH scenario. For 1-shot learning, ours outperforms the best baselines by 12.76%, 9.35% and 9.02% on three datasets, respectively. For 5-shot learning, ours outperforms the best baselines by 21.01%, 18.61% and 22.87% on three datasets, respectively. All baselines can not effectively solve the black-box DFML problem in API-SH scenario with the similar reasons of API-SS discussed in Sec. 5.2. Ours is far better than all baselines and can apply to the API-SH scenario without any change because the meta knowledge distillation involved in our BiDf-MKD framework imposes no restriction on the underlying model architectures and scale inside each black-box API.

5.4. Experiments of black-box DFML in API-MH

Overview. We further perform experiments in a more challenging **API-MH** scenario, where all APIs are designed for solving different tasks from multiple meta training subsets (CIFAR-FS, MiniImageNet and CUB) with heterogeneous architectures (Conv4, ResNet10 and ResNet18) inside. For meta testing, we evaluate the meta-learned meta-

Table 2. Compare to baselines in API-SH scenario.

API-SH	Method	1-shot	5-shot
CIFAR-FS 5-way	Random	20.35 ± 0.42	20.59 ± 0.45
	Best-API	19.04 ± 0.68	19.04 ± 0.67
	Single-DFKD	19.56 ± 0.67	20.06 ± 0.60
	Distill-Avg	22.82 ± 0.38	25.91 ± 0.45
	Ours	$\mathbf{35.58 \pm 0.79}$	$\mathbf{46.92 \pm 0.77}$
MiniImageNet 5-way	Random	21.20 ± 0.38	21.13 ± 0.37
	Best-API	20.51 ± 0.63	20.39 ± 0.62
	Single-DFKD	20.11 ± 0.64	20.23 ± 0.66
	Distill-Avg	20.32 ± 0.22	20.67 ± 0.24
	Ours	$\mathbf{30.55 \pm 0.62}$	$\mathbf{39.74 \pm 0.65}$
CUB 5-way	Random	21.09 ± 0.38	21.11 ± 0.37
	Best-API	19.99 ± 0.69	19.95 ± 0.70
	Single-DFKD	20.13 ± 0.66	20.24 ± 0.64
	Distill-Avg	20.46 ± 0.24	21.02 ± 0.26
	Ours	$\mathbf{30.11 \pm 0.58}$	$\mathbf{43.98 \pm 0.64}$

initialization on unseen tasks from CIFAR-FS, MiniImageNet and CUB simultaneously.

Results. Tab. 3 shows the results for 5-way classification in API-MH scenario. Ours has significant performance advantages (11.21% and 17.13% for 1-shot and 5-shot, respectively) compared with all other baselines, which shows the superiority and broad applicability of our BiDf-MKD framework to work well with black-box APIs from multiple datasets with heterogeneous architectures.

Table 3. Compare to baselines in API-MH scenario.

API-MH	Method	1-shot	5-shot
5-way	Random	20.88 ± 0.39	21.00 ± 0.40
	Best-API	19.44 ± 0.65	19.64 ± 0.66
	Single-DFKD	19.04 ± 0.66	19.68 ± 0.64
	Distill-Avg	21.57 ± 0.25	23.11 ± 0.29
	Ours	$\mathbf{32.78 \pm 0.60}$	$\mathbf{40.24 \pm 0.65}$

5.5. Ablation Studies

Effectiveness of each component of our framework. Tab. 4 analyzes the effectiveness of each component on CIFAR-FS in API-SS scenario. We first introduce the **vanilla** only performing meta-learning via task memory replay (Sec. 4.2). The vanilla still achieves a significant performance gain compared with the best baselines in Tab. 1 by 8.89% and 16.89% for 1-shot and 5-shot learning, which hints the feasibility for meta-learning on synthetic data. By adding **BiDf-MKD** to transfer the meta knowledge, we observe a performance gain of 1.96% and 2.49%. The reason is that BiDf-MKD provides a way to leverage richer supervision (i.e., the semantic class relationship in the soft-label prediction from those black-box APIs) instead of the only hard-label supervision from the synthetic data. We also observe an improvement (0.89% and 0.79%) from our **boundary** query set recovery technique (Fig. 3), which verifies its effectiveness for alleviating the knowledge vanish issue. With all components, we achieve the best performance with a boosting improvement of 2.35% and 3.13%, thus demonstrating the effectiveness of the joint schema.

Table 4. Ablation studies on CIFAR-FS in API-SS scenario.

API-SS	Component		Accuracy	
	BiDf-MKD	Boundary	5-way 1-shot	5-way 5-shot
Vanilla			33.13 ± 0.66	44.45 ± 0.76
	✓		35.09 ± 0.71	46.94 ± 0.74
		✓	34.02 ± 0.70	45.24 ± 0.74
Ours	✓	✓	$\mathbf{35.48 \pm 0.67}$	$\mathbf{47.58 \pm 0.74}$
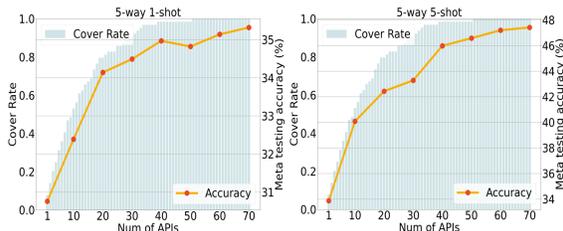
Figure 5. Effect of the number of APIs in API-SS scenario.

Effectiveness of zero-order gradient estimator. Tab. 5 provides an unfair comparison with an unfair baseline, white-box DFML, by performing our BiDf-MKD in an ideal white-box setting where the actual first-order (FO) gradients of the parameters inside the APIs are available for task recovery. The accuracy of white-box DFML serves as the upper bound for that of black-box DFML. The minor performance gaps on three datasets demonstrate our zero-order (ZO) gradient estimator can provide reasonable gradient estimation, which can achieve comparable meta-learning performance.

Table 5. Effectiveness of zero-order gradient estimator. Grey: unfair comparison with white-box DFML.

API-SS	Method	1-shot	5-shot
CIFAR-FS 5-way	FO	37.66 ± 0.75	51.16 ± 0.79
	ZO	35.48 ± 0.67	47.58 ± 0.74
MiniImageNet 5-way	FO	30.66 ± 0.59	42.30 ± 0.64
	ZO	29.35 ± 0.60	39.47 ± 0.64
CUB 5-way	FO	31.62 ± 0.60	44.32 ± 0.69
	ZO	29.10 ± 0.64	43.43 ± 0.66

Effect of the number of APIs. Fig. 5 shows the performance difference with the different number of black-box APIs in API-SS scenario. Here, we further introduce an intrinsic factor, i.e., cover rate, which indicates the coverage rate of classes in the meta training subset. As the number of APIs increases, the cover rate increases, boosting the generalization ability for unseen tasks with higher meta testing accuracy. When the cover rate reaches 100% (more than 50 APIs), we can still observe a performance improvement because the additional APIs provide richer supervision of semantic relationships among different classes.

Effect of the number of query times. Tab. 8 shows the effect of the number of query times (i.e., the value of q in Eq. (5)) on the accuracy and time cost. More query times
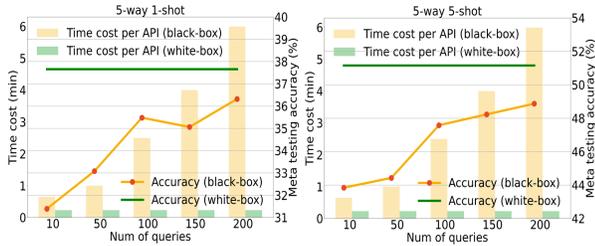
Figure 6. Effect of the number of query times on the accuracy and time cost. Here, white-box DFML provides unfair bounds of accuracy and time cost.
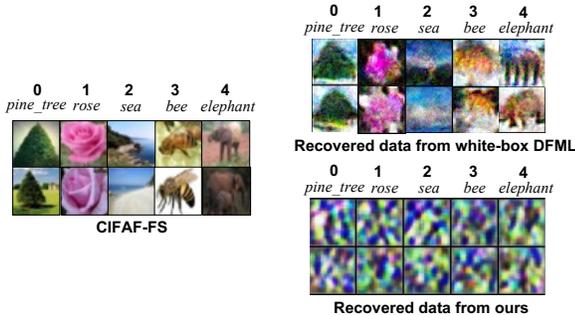
Figure 7. Visualization of recovered data on CIFAR-FS.

can lead to more accurate zero-order gradient estimation, thus leading to more accurate task recovery results and better meta-learning performance. Considering the time cost, we set $q = 100$ in practice with comparable performance and tolerable time cost compared with the unfair performance bound of white-box DFML.

Data privacy. A profound significance of black-box DFML is that we can achieve comparable meta-learning performance compared with the unfair white-box DFML without data privacy leakage. We argue that the real intention for releasing APIs (or pre-trained models) without data is to protect data privacy and security. However, as shown in Fig. 7, the recovered data from white-box DFML is highly similar to the original data, leaking sensitive data information and violating the original intentions. In contrast, ours recovers the data visually distinct from the original data, thus avoiding privacy leakage. Note that although the recovered data from ours look much different from the original data, ours still achieves a comparable meta-learning performance compared with the white-box DFML (see Tab. 5).

6. Conclusion

For the first time, we propose a practical and valuable setting of DFML, i.e., black-box DFML, which aims to meta-learn the meta-initialization from a collection of black-box APIs without access to the original training data and with only inference access, to enable efficient learning of new tasks without privacy leakage. To solve this challenging

problem, we propose a novel BiDf-MKD framework integrated with task memory replay to transfer the general meta knowledge into one single model. At last, we propose three real-world scenarios for a complete and practical evaluation of black-box DFML, where extensive experiments verify our framework's effectiveness and superiority.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Abbas, M., Xiao, Q., Chen, L., Chen, P.-Y., and Chen, T. Sharp-maml: Sharpness-aware model-agnostic meta learning. *arXiv preprint arXiv:2206.03996*, 2022.

Ahn, S., Hu, S. X., Damianou, A., Lawrence, N. D., and Dai, Z. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9163–9171, 2019.

Behl, H. S., Baydin, A. G., and Torr, P. H. Alpha maml: Adaptive model-agnostic meta-learning. *arXiv preprint arXiv:1905.07435*, 2019.

Ben-Naim, A. *A farewell to entropy: Statistical thermodynamics based on information: S*. World Scientific, 2008.

Bertinetto, L., Henriques, J. F., Torr, P. H., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

Chawla, A., Yin, H., Molchanov, P., and Alvarez, J. Data-free knowledge distillation for object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3289–3298, 2021.

Chen, H., Wang, Y., Xu, C., Xu, C., and Tao, D. Learning student networks via feature embedding. *arXiv: Learning*, 2018.

Chen, J., Zhan, L.-M., Wu, X.-M., and Chung, F.-l. Variational metric scaling for metric-based meta-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3478–3485, 2020.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C., and Huang, J.-B. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

Deng, X. and Zhang, Z. Graph-free knowledge distillation for graph neural networks. *arXiv preprint arXiv:2105.07519*, 2021.

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

Finn, C., Xu, K., and Levine, S. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

Fredrikson, M., Jha, S., and Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333, 2015.

Gao, X., Jiang, B., and Zhang, S. On the information-adaptive variants of the admm: an iteration complexity perspective. *Journal of Scientific Computing*, 76(1):327–363, 2018.

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

Harrison, J., Sharma, A., Finn, C., and Pavone, M. Continuous meta-learning without tasks. *Advances in neural information processing systems*, 33:17571–17581, 2020.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

Hu, Z., Shen, L., Wang, Z., Liu, T., Yuan, C., and Tao, D. Architecture, dataset and model-scale agnostic data-free meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7736–7745, 2023.

Jeong, T. and Kim, H. Ood-maml: Meta-learning for few-shot out-of-distribution detection and classification. *Advances in Neural Information Processing Systems*, 33: 3907–3916, 2020.

Kariyappa, S., Prakash, A., and Qureshi, M. K. Maze: Data-free model stealing attack using zeroth-order gradient estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13814–13823, 2021.

Koratana, A., Kang, D., Bailis, P., and Zaharia, M. Lit: Learned intermediate representation training for model compression. *International Conference on Machine Learning*, 2019.

Kreer, J. A question of terminology. *IRE Transactions on Information Theory*, 3(3):208–208, 1957.

Li, A., Huang, W., Lan, X., Feng, J., Li, Z., and Wang, L. Boosting few-shot learning with adaptive margin loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12576–12584, 2020.

Liu, C., Wang, Z., Sahoo, D., Fang, Y., Zhang, K., and Hoi, S. C. Adaptive task sampling for meta-learning. In *European Conference on Computer Vision*, pp. 752–769. Springer, 2020a.

Liu, L., Zhou, T., Long, G., Jiang, J., and Zhang, C. Learning to propagate for graph meta-learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Liu, S., Chen, P.-Y., Kailkhura, B., Zhang, G., Hero III, A. O., and Varshney, P. K. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020b.

Liu, Z., Shen, Z., Long, Y., Xing, E., Cheng, K.-T., and Leichner, C. Data-free neural architecture search via recursive label calibration. *arXiv preprint arXiv:2112.02086*, 2021.

Lopes, R. G., Fenu, S., and Starner, T. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

MacKay, D. J., Mac Kay, D. J., et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

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

Munkhdalai, T. and Yu, H. Meta networks. In *International Conference on Machine Learning*, pp. 2554–2563. PMLR, 2017.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

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

Roman, D., Schade, S., Berre, A., Bodsberg, N. R., and Langlois, J. Model as a service (maas). In *AGILE Workshop: Grid Technologies for Geospatial Applications, Hannover, Germany*, 2009.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

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

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Simon, C., Koniusz, P., and Harandi, M. Meta-learning for multi-label few-shot classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3951–3960, 2022.

Sun, T., He, Z., Qian, H., Zhou, Y., Huang, X.-J., and Qiu, X. Bbtv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3916–3930, 2022a.

Sun, T., Shao, Y., Qian, H., Huang, X., and Qiu, X. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855. PMLR, 2022b.

Truong, J.-B., Maini, P., Walls, R. J., and Papernot, N. Data-free model extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4771–4780, 2021.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The caltech-ucsd birds-200-2011 dataset. 2011.

Wang, Y., Chao, W.-L., Weinberger, K. Q., and van der Maaten, L. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

Wang, Z., Zhao, Y., Yu, P., Zhang, R., and Chen, C. Bayesian meta sampling for fast uncertainty adaptation. In *International Conference on Learning Representations*, 2020.

Wang, Z., Duan, T., Fang, L., Suo, Q., and Gao, M. Meta learning on a sequence of imbalanced domains with difficulty awareness. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8947–8957, 2021.

Wang, Z., Shen, L., Duan, T., Zhan, D., Fang, L., and Gao, M. Learning to learn and remember super long multi-domain task sequence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7982–7992, June 2022a.

Wang, Z., Shen, L., Fang, L., Suo, Q., Zhan, D., Duan, T., and Gao, M. Meta-learning with less forgetting on large-scale non-stationary task distributions. In *European Conference on Computer Vision*, pp. 221–238. Springer, 2022b.

Wang, Z., Wang, X., Shen, L., Suo, Q., Song, K., Yu, D., Shen, Y., and Gao, M. Meta-learning without data via wasserstein distributionally-robust model fusion. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022c.

Wu, X., Fredrikson, M., Jha, S., and Naughton, J. F. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pp. 355–370. IEEE, 2016.

Yang, S., Liu, L., and Xu, M. Free lunch for few-shot learning: Distribution calibration. *arXiv preprint arXiv:2101.06395*, 2021.

Yao, H., Wang, Y., Wei, Y., Zhao, P., Mahdavi, M., Lian, D., and Finn, C. Meta-learning with an adaptive task scheduler. *Advances in Neural Information Processing Systems*, 34:7497–7509, 2021.

Ye, H.-J., Hu, H., Zhan, D.-C., and Sha, F. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8808–8817, 2020a.

Ye, H.-J., Lu, S., and Zhan, D.-C. Distilling cross-task knowledge via relationship matching. *Computer Vision and Pattern Recognition*, 2020b.

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

Yin, M., Tucker, G., Zhou, M., Levine, S., and Finn, C. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.

Zhang, J., Chen, C., Li, B., Lyu, L., Wu, S., Ding, S., Shen, C., and Wu, C. Dense: Data-free one-shot federated learning. In *Advances in Neural Information Processing Systems*, 2022a.

Zhang, L., Shen, L., Ding, L., Tao, D., and Duan, L.-Y. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10174–10183, 2022b.

Zhang, Y., Jia, R., Pei, H., Wang, W., Li, B., and Song, D. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 253–261, 2020.

Zhang, Y., Yao, Y., Jia, J., Yi, J., Hong, M., Chang, S., and Liu, S. How to robustify black-box ML models? a zeroth-order optimization perspective. In *International Conference on Learning Representations*, 2022c. URL https://openreview.net/forum?id=W9G_ImpHlQd.

Zhou, Y., Wang, Z., Xian, J., Chen, C., and Xu, J. Meta-learning with neural tangent kernels. In *International Conference on Learning Representations*, 2021.

Zhu, Z., Hong, J., and Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 12878–12889. PMLR, 2021.

Appendix

A. Mutual Information and Entropy

A.1. Mutual information

Mutual information (Shannon, 1948; Kreer, 1957) $I(X, Y)$ measures the mutual dependence of two random variables X and Y. Intuitively, it quantifies how much observing one random variable can reduce the uncertainty about the other random variable (i.e., uncertainty reduction) or measures how much observing one random variable can obtain the information about the other random variable (i.e., information gain). Below, we give the definition of discrete mutual information and continuous mutual information, respectively.

Definition A.1. *Discrete mutual information.* The definition of mutual information $I(X, Y)$ of two discrete random variables X and Y is given by

$$I(X; Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} P_{(X,Y)}(x, y) \log \frac{P_{(X,Y)}(x, y)}{P_X(x) P_Y(y)}.$$

X and Y are random variables with the values over the space $\mathcal{X}$ and $\mathcal{Y}$, respectively. $P_{(X,Y)}$ is their joint distribution. P_X and P_Y are their marginal distributions.

For two **continuous** random variables X and Y, we replace the summations with the integrals.

Definition A.2. *Continuous mutual information.* The definition of mutual information $I(X, Y)$ of two continuous random variables X and Y is given by

$$I(X; Y) = \int_{\mathcal{Y}} \int_{\mathcal{X}} P_{(X,Y)}(x, y) \log \frac{P_{(X,Y)}(x, y)}{P_X(x) P_Y(y)} \, dx \, dy.$$

A.2. Entropy

Entropy (Ben-Naim, 2008) $H(X)$ is a measure of uncertainty of the random variable X. Below, we give the definition of discrete entropy and continuous entropy, respectively.

Definition A.3. *Discrete entropy.* The definition of the entropy $H(X)$ of a discrete random variable X is given by

$$H(X) = - \sum_{x \in \mathcal{X}} P_X(x) \log P_X(x).$$

We can also extend it to the **continuous** random variable by replacing the summations with the integrals.

Definition A.4. *Continuous entropy.* The definition of the entropy $H(X)$ of a continuous random variable X is given by

$$H(X) = - \int_{\mathcal{X}} P_X(x) \log P_X(x) \, dx.$$

A.3. Conditional entropy

Conditional entropy quantifies the uncertainty of one random variable given that the value of the other random variable is known. Below, we give the definition of discrete conditional entropy and conditional continuous entropy, respectively.

Definition A.5. *Discrete conditional entropy.* The definition of the conditional entropy $H(X|Y)$ of a discrete random variable X given the other discrete random variable Y is given by

$$H(Y|X) = - \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} P_{(X,Y)}(x, y) \log P_{(Y|X=x)}(y).$$

Definition A.6. *Continuous conditional entropy.* The definition of the conditional entropy $H(X|Y)$ of a continuous random variable X given the other continuous random variable Y is given by

$$H(Y|X) = - \int_{\mathcal{Y}} \int_{\mathcal{X}} P_{(X,Y)}(x, y) \log P_{(Y|X=x)}(y) \, dx \, dy.$$

A.4. Relation between mutual information and entropy

Here, we give a detailed deduction of the relation between mutual information and entropy for the case of discrete random variables X and Y. The deduction for the case of continuous random variables is the same except for replacing the summations with the integrals.

$$
\begin{aligned}
\mathrm{I}(X;Y) &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log\frac{P_{(X,Y)}(x,y)}{P_X(x)P_Y(y)} \\
&= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log\frac{p_{(X,Y)}(x,y)}{P_X(x)} - \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log P_Y(y) \\
&= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_X(x)P_{(Y|X=x)}(y)\log P_{Y|X=x}(y) - \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{(X,Y)}(x,y)\log P_Y(y) \\
&= \sum_{x\in\mathcal{X}} P_X(x)\left(\sum_{y\in\mathcal{Y}} P_{(Y|X=x)}(y)\log P_{(Y|X=x)}(y)\right) - \sum_{y\in\mathcal{Y}}\left(\sum_{x\in\mathcal{X}} P_{(X,Y)}(x,y)\right)\log P_Y(y) \\
&= -\sum_{x\in\mathcal{X}} P_X(x)\mathrm{H}(Y\mid X=x) - \sum_{y\in\mathcal{Y}} P_Y(y)\log P_Y(y) \\
&= -\mathrm{H}(Y\mid X) + \mathrm{H}(Y) \\
&= \mathrm{H}(Y) - \mathrm{H}(Y\mid X).
\end{aligned}
$$

B. Full Architecture of Generator

Tab. 6 lists the structure of the generator in our proposed BiDf-MKD framework. The generator takes the standard Gaussion noise as input and outputs the recovered data. Here, $d_{\boldsymbol{z}}$ is dimension of Gaussian noise data $\boldsymbol{z}$, which is set as 256 in practice. The $negative\_slope$ of LeakyReLU is 0.2. We set img_size as 32 for APIs pre-trained on CIFAR-FS and 84 for APIs 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.

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	batch normalization layer	
$Conv2D(input, output, filter\_size, stride, padding)$	convolutional layer	

Structure	Dimension	
	Before	**After**
$\boldsymbol{z}\in\mathbb{R}_{d_{\boldsymbol{z}}}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{1})$	—	$[\,bs, d_{\boldsymbol{z}}\,]$
$FC(\boldsymbol{z})$	$[\,bs, d_{\boldsymbol{z}}\,]$	$[\,bs, 2\times nf\times(img\_size//4)\times(img\_size//4)\,]$
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	$[\,bs, nc, img\_size, img\_size\,]$	$[\,bs, nc, img\_size, img\_size\,]$

Table 6. Detailed structure of generator in our proposed BiDf-MKD framework. We highlight the dimension change in red.

C. More Discussions

C.1. Discussions on robustness against API quality variation

Fig. 8 shows the reported accuracy of given APIs (Conv4, ResNet10 and ResNet18, respectively) pre-trained on CIFAR-FS, MiniImageNet and CUB, respectively. We collect these black-box APIs to simulate the real-world scenario where various MaaS providers provide black-box APIs pre-trained on multiple datasets with heterogeneous architectures for solving different tasks. In addition, since we only have inference access to these black-box APIs without the original training data, we can not further improve the accuracy of these APIs by fine-tuning or re-training. In other words, our proposed framework should work well in a setting where a small number of APIs can be with relatively low accuracy. Therefore, as shown in Fig. 8, we perform our proposed BiDf-MKD method with the APIs of both high quality and relatively low quality. The results in Tabs. 1 to 3 show the superiority of our BiDf-MKD compared with the SOTA baselines against the API quality variation.
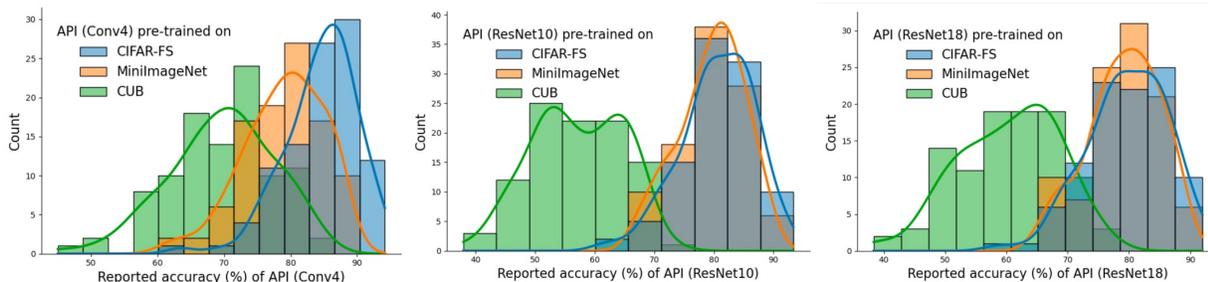
Figure 8. Histogram of reported accuracy of given APIs (Conv4, ResNet10, and ResNet18, respectively) pre-trained on CIFAR-FS, MiniImageNet, and CUB, respectively.

C.2. Comparison between knowledge vanish issue and memorization issue of meta-learning

The meta-learning memorization (Yin et al., 2019) describes an issue that the meta model overfits the training tasks, thus leading to poor generalization ability to unseen tasks. The memorization issue occurs the meta model can ignore the support set and directly perform well on the query set, i.e., the support set is useless. Our knowledge vanish issue is remarkably distinct from the memorization issue because the former occurs when the query set is useless. In addition, the knowledge vanish issue is more significant in the data-free meta-learning setting than in the data-based meta-learning setting because of the relatively low diversity of recovered data.

D. Summarized Algorithm

We summarize the algorithm of our end-to-end BiDf-MKD framework in Alg. 1.

E. Detailed Experimental Setup

E.1. Implementation details

API-SS. For API-SS scenario, all APIs are designed for solving different tasks from the same meta training subset with the same model architecture inside. We take Conv4 as the architecture of the model inside each API and the meta model for API-SS. Conv4 is commonly used in meta-learning works (Finn et al., 2017; Chen et al., 2020; Liu et al., 2020a; Wang et al., 2019), which consists of four convolutional blocks. Each block consists of 32 3×3 filters, a BatchNorm, a ReLU and a 2×2 max-pooling. All APIs are designed for solving different 5-way tasks, which are constructed by randomly sampling 5 classes from the same meta training subset (CIFAR-FS, MiniImageNet or CUB). We adopt Adam optimizer to pre-train the model inside each black-box API via standard supervised learning with a learning rate of 0.01. In practice, we collect 100 black-box APIs. For BiDf-MKD, we recover 30 images for the support set and query set, respectively. We adopt Adam optimizer to optimize the generator parameters $\boldsymbol{\theta}_G$ and input $\boldsymbol{z}$ simultaneously by minimizing Eq. (2) with the learning rate of 0.001 for 200 epochs. We adopt Adam optimizer to optimize the meta model parameters $\boldsymbol{\theta}$ by minimizing Eq. (8) with the inner-level learning rate of 0.01 and the outer-level learning rate of 0.001. For boundary query set recovery, we empirically set the coefficient λ_Q as 1. For task memory replay, we adopt MAML to perform meta-learning on the interpolated tasks.

Algorithm 1: Black-box data-free meta-learning.

Input: Max iterations N; a collection of T APIs $\{A_i\}_{i=0}^{T-1}$; the meta model $F(\cdot; \boldsymbol{\theta})$; memory bank $\mathcal{B}$; the batch size BatchSize.
Output: meta-initialization $\boldsymbol{\theta}$.

Randomly initialize $\boldsymbol{\theta}$
$\mathcal{B} \leftarrow [\,]$
for $t \leftarrow 0$ **to** N **do**
    for $bs \leftarrow 0$ **to** BatchSize **do**
        if no task memory replay **then**   `// BiDf-MKD`
            Sample an API A_i
            `// support set recovery`
            Recover the support set $\boldsymbol{S}_i$ by minimizing Eq. (2) via zero-order gradient estimator
            `// inner level of BiDf-MKD`
            Minimize Eq. (7) via gradient descent w.r.t. to $\boldsymbol{\theta}$ to obtain the task-specific model parameters $\boldsymbol{\theta}_i$
            `// boundary query set recovery`
            Recover the query set $\boldsymbol{Q}_i$ by minimizing Eq. (9) via zero-order gradient estimator
            `// outer level of BiDf-MKD`
            Minimize Eq. (8) via gradient descent to update the meta model parameters $\boldsymbol{\theta}$
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{\boldsymbol{S}_i, \boldsymbol{Q}_i\}$
        else   `// task memory replay`
            `// task interpolation`
            Construct interpolated task $\{\boldsymbol{S}_m, \boldsymbol{Q}_m\}$ from $\mathcal{B}$
            `// memory replay`
            Minimize Eq. (10) via gradient descent to update the meta model parameters $\boldsymbol{\theta}$
        $bs \leftarrow bs + 1$
    $t \leftarrow t + 1$

We conduct MAML with the Adam optimizer with the inner-level learning rate of 0.01 and the outer-level learning rate of 0.001. For the zero-order gradient estimator, we query each API with 100 random direction vectors drawn from the sphere of a unit ball. We set the smoothing parameter μ as 0.005 in Eq. (5).

API-SH. For API-SH scenario, all APIs are designed for solving different tasks from the same meta training subset with heterogeneous model architectures inside. We take Conv4, ResNet-10 and ResNet-18 as the architectures inside the given APIs. Compared to Conv4, ResNet-10 and ResNet-18 are larger-scale neural networks. We take Conv4 as the meta model architecture. All APIs are designed for solving different 5-way tasks, which are constructed by randomly sampling 5 classes from the same meta training subset. The other configurations are the same as those of API-SS.

API-MH. For API-MH scenario, all APIs are designed for solving different tasks from multiple meta training subsets with heterogeneous model architectures inside. We take Conv4, ResNet-10 and ResNet-18 as the architectures inside the given APIs. We take Conv4 as the meta model architecture. All APIs are designed for solving different 5-way tasks, which are constructed by randomly sampling 5 classes from multiple meta training subsets, including CIFAR-FS, MiniImageNet and CUB. For meta testing, we evaluate the meta-learned meta-initialization on unseen tasks from CIFAR-FS, MiniImageNet and CUB, respectively. The other configurations are the same as those of API-SS.

E.2. Datasets for Meta Testing

CIFAR-FS (Bertinetto et al., 2018), **MiniImageNet** (Vinyals et al., 2016) are commonly used in meta-learning, consisting of 100 classes with 600 images per class. We split each dataset into three subsets following (Wang et al., 2022c): 64 classes for meta training, 16 classes for meta validation and 20 classes for meta testing. In addition to these, we investigate **CUB-200-2011** (CUB) birds dataset (Wah et al., 2011), composing of 11,788 images of 200 bird species, to evaluate the effectiveness of our BiDf-MKD on fine-grained classification. We split into three subsets following (Chen et al., 2019): 100 classes for meta training, 50 classes for meta validation and 50 classes for meta testing. For CIFAR-FS, MiniImageNet and CUB, all splits are non-overlapping. Note that we have no access to the meta training subset in the DFML setting, and we only use meta testing subset for evaluation.

E.3. Evaluation metric.

We evaluate the performance by the average accuracy and standard deviation over 600 unseen target tasks sampled from meta testing subset. For API-SS and API-SH, we construct several meta testing tasks from one specific meta testing subset (i.e., CIFAR-FS, MiniImageNet or CUB), while for API-MH, we construct several meta testing tasks from all meta testing subsets (i.e., CIFAR-FS, MiniImageNet and CUB) equally.

F. More Results

Effect of the number of APIs. Tab. 7 shows the effect of the number of black-box APIs on the meta testing accuracy. The results in Tab. 7 are consistent with Fig. 5. We additionally introduce an intrinsic factor, i.e., cover rate, which indicates the coverage rate of classes of meta training subset, to better illustrate the relationship between the number of APIs and the meta-learning performance. Refer to Sec. 5.5 for detailed result analysis.

Table 7. Effect of the number of APIs in API-SS scenario.

API-SS	APIs	1-shot	5-shot
	1	30.76 ± 0.64	33.87 ± 0.72
	10	32.39 ± 0.66	40.08 ± 0.74
	20	34.14 ± 0.74	42.43 ± 0.74
	30	34.49 ± 0.66	43.28 ± 0.81
CIFAR-FS	40	34.97 ± 0.64	45.98 ± 0.76
5-way	50	34.82 ± 0.71	46.58 ± 0.81
	60	35.14 ± 0.62	47.20 ± 0.74
	70	35.32 ± 0.64	47.42 ± 0.79
	100	35.48 ± 0.67	47.58 ± 0.74

Effect of the number of queris. Tab. 8 shows the effect of the number of query times (i.e., the value of q in Eq. (5)) on the meta testing accuracy. The results in Tab. 8 are consistent with Fig. 6. As we can see, more query times can lead to more accurate zero-order gradient estimation, thus leading to more accurate data recovery results and better meta-learning performance. Considering the time cost shown in Fig. 6, we set $q = 100$ in practice with comparable performance and tolerable time cost compared with the unfair performance bound of white-box DFML.

Table 8. Effect of the number of queries in API-SS scenario.

API-SS	Queries	1-shot	5-shot
	10	31.39 ± 0.66	43.84 ± 0.74
CIFAR-FS	50	33.08 ± 0.68	44.43 ± 0.74
5-way	100	35.48 ± 0.67	47.58 ± 0.74
	150	35.06 ± 0.77	48.23 ± 0.81
	200	36.32 ± 0.77	48.87 ± 0.81

Effect of the number of training classes for each task. To evaluate the effect of the number of training class for each black-box API, we conduct the experiments in API-SS scenario, where each API trains for a 10-way classification problem. During meta-testing, we construct several 10-way meta testing tasks for evaluation. As shown in Tab. 9, for CIFAR-FS, ours outperforms the best baseline by 9.45% and 20.45% for 1-shot learning and 5-shot learning, respectively. Compared to 5-way classification (Tab. 1), the meta testing accuracy of 10-way classification is relatively lower because it is more challenging. Ours consistently outperforms all baselines in both 5-way and 10-way classification problems.

Table 9. Effect of the number of training classes for each task.

API-SS	Method	1-shot	5-shot
	Random	10.12 ± 0.30	10.20 ± 0.26
CIFAR-FS	Best-API	9.86 ± 0.62	9.94 ± 0.63
10-way	Single-DFKD	10.02 ± 0.60	10.08 ± 0.64
	Distill-Avg	11.70 ± 0.21	12.22 ± 0.24
	Ours	$\mathbf{21.15 \pm 0.37}$	$\mathbf{32.67 \pm 0.41}$

Hyperparameter sensitivity. In Tab. 10, we evaluate the performance sensitivity of our BiDf-MKD framework for different values of λ_Q (0.1, 1.0 and 10.0) in Eq. (9). The meta testing accuracy is stable and not sensitive to λ_Q value variations for both 1-shot and 5-shot learning, which verifies the consistent superiority with different λ_Q values. This advantage also makes it easy to apply our BiDf-MKD framework in practice.

Table 10. Hyperparameter sensitivity on CIFAR-FS 5-way classification in API-SS scenario.

λ_Q	5-way 1-shot	5-way 5-shot
$\lambda_Q = 0.1$	35.32 ± 0.78	47.72 ± 0.76
$\lambda_Q = 1.0$	35.48 ± 0.67	47.58 ± 0.74
$\lambda_Q = 10.0$	35.06 ± 0.78	47.02 ± 0.74

Larger-shot comparisons. We further investigate the meta-learning performance of our BiDf-MKD framework in the setting where there are more shots (i.e., the value of K) in the support set for the meta testing tasks (i.e., during meta-testing). In Fig. 9, we conduct 5-way classification experiments on CIFAR-FS in API-SS scenario under different numbers of shots of meta testing tasks. We consider $K = \{1, 5, 10, 20, 30, 40\}$. Note that $K = \{1, 5\}$ is relatively larger than the common meta-learning setting where $K = \{10, 20, 30, 40\}$, but is smaller than the traditional supervised learning to train a complex neural network. For meta training, we set the same K value for task memory replay. We consider the strong baseline "Distill-Avg" discussed in Sec. 5.1 considering its relatively better performance shown in Tabs. 1 to 3. As shown in Fig. 9, we can obtain a higher meta testing accuracy when K increases for all methods and our BiDf-MKD framework outperforms the baseline at all the different shot settings. Simply averaging all surrogate models (i.e., Distill-Avg) lacking the meta-learning objective, leading to the bad performance in the low-shot setting (i.e., $K = \{1, 5\}$). Besides, Distill-Avg also does not perform well in the larger-shot setting (i.e., $K = \{10, 20, 30, 40\}$) because all surrogate models train on different tasks, thus lacking precise correspondence among them. In contrast, our BiDf-MKD framework outperforms the baseline at every K value, showing its effectiveness across a broad spectrum of K values.
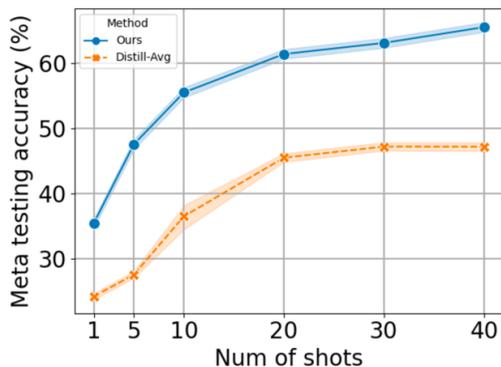
Figure 9. Larger-shot comparisons.