# Meta-DMoE: Adapting to Domain Shift by Meta-Distillation from Mixture-of-Experts

**Tao Zhong**[*1], **Zhixiang Chi**[*2], **Li Gu**[*2], **Yang Wang**[2,3], **Yuanhao Yu**[2], **Jin Tang**[2]

[1]University of Toronto, [2]Huawei Noah's Ark Lab, [3]Concordia University
tao.zhong@mail.utoronto.ca yang.wang@concordia.ca
{zhixiang.chi, li.gu, yuanhao.yu, tangjin}@huawei.com

## Abstract

In this paper, we tackle the problem of domain shift. Most existing methods perform training on multiple source domains using a single model, and the same trained model is used on all unseen target domains. Such solutions are sub-optimal as each target domain exhibits its own speciality, which is not adapted. Furthermore, expecting the single-model training to learn extensive knowledge from the multiple source domains is counterintuitive. The model is more biased toward learning only domain-invariant features and may result in negative knowledge transfer. In this work, we propose a novel framework for unsupervised test-time adaptation, which is formulated as a knowledge distillation process to address domain shift. Specifically, we incorporate Mixture-of-Experts (MoE) as teachers, where each expert is separately trained on different source domains to maximize their speciality. Given a test-time target domain, a small set of unlabeled data is sampled to query the knowledge from MoE. As the source domains are correlated to the target domains, a transformer-based aggregator then combines the domain knowledge by examining the interconnection among them. The output is treated as a supervision signal to adapt a student prediction network toward the target domain. We further employ meta-learning to enforce the aggregator to distill positive knowledge and the student network to achieve fast adaptation. Extensive experiments demonstrate that the proposed method outperforms the state-of-the-art and validates the effectiveness of each proposed component. Our code is available at https://github.com/n3il666/Meta-DMoE.

## 1 Introduction

The emergence of deep models has achieved superior performance [32, 40, 47]. Such unprecedented success is built on the strong assumption that the training and testing data are highly correlated (i.e., they are both sampled from the same data distribution). However, the assumption typically does not hold in real-world settings as the training data is infeasible to cover all the ever-changing deployment environments [39]. Reducing such distribution correlation is known as distribution shift, which significantly hampers the performance of deep models. Although human is more robust against the distribution shift, artificial learning-based systems suffer more from performance degradation.

One line of research aims to mitigate the distribution shift by exploiting some unlabeled data from a target domain, which is known as unsupervised domain adaptation (UDA) [24, 51, 26]. The unlabeled data is an estimation of the target distribution [87]. Therefore, UDA normally adapts to the target domain by transferring the source knowledge via a common feature space with less effect from domain discrepancy [79, 50]. However, UDA is less applicable for real-world scenarios as

[*]equal contribution

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

repetitive large-scale training is required for every target domain. In addition, collecting the data samples from a target domain in advance might be unavailable as the target distribution could be unknown during training. Domain generalization (DG) [54, 28, 6] is an alternative line of research but more challenging as it assumes the prior knowledge of the target domains is unknown. DG methods leverage multiple source domains for training and directly use the trained model on all unseen domains. As the domain-specific information for the target domains is not adapted, a generic model is sub-optimal [68, 17].

Test-time adaptation with DG allows the model to exploit the unlabeled data during testing to overcome the limitation of using a flawed generic model for all unseen target domains. In ARM [87], meta-learning [25] is utilized for training the model as an initialization such that it can be adapted using the unlabeled data from the unseen target domain before making the final inference. However, we observed that ARM only trains a single model, which is counterintuitive for the multi-source domain setting. There is a certain amount of correlations among the source domains while each of them also exhibits its own specific knowledge. When the number of source domains rises, data complexity dramatically increases, which impedes the exploration of the dataset thoroughly. Furthermore, real-world domains are not always balanced in data scales [39]. Therefore, the single-model training is more biased toward the domain-invariant features and dominant domains instead of the domain-specific features [12].

In this work, we propose to formulate the test-time adaptation as the process of knowledge distillation [34] from multiple source domains. Concretely, we propose to incorporate the concept of Mixture-of-Experts (MoE), which is a natural fit for the multi-source domain settings. The MoE models are treated as a teacher and separately trained on the corresponding domain to maximize their domain speciality. Given a new target domain, a few unlabeled data are collected to query the features from expert models. A transformer-based knowledge aggregator is proposed to examine the interconnection among queried knowledge and aggregate the correlated information toward the target domain. The output is then treated as a supervision signal to update a student prediction network to adapt to the target domain. The adapted student is then used for subsequent inference. We employ bi-level optimization as meta-learning to train the aggregator at the meta-level to improve generalization. The student network is also meta-trained to achieve fast adaptation via a few samples. Furthermore, we simulate the test-time out-of-distribution scenarios during training to align the training objective with the evaluation protocol.

The proposed method also provides additional advantages over ARM: 1) Our method provides a larger model capability to improve the generalization power; 2) Despite the higher computational cost, only the adapted student network is kept for inference, while the MoE models are discarded after adaptation. Therefore, our method is more flexible in designing the architectures for the teacher or student models. (e.g., designing compact models for the power-constrained environment); 3) Our method does not need to access the raw data of source domains but only needs their trained models. So, we can take advantage of private domains in a real-world setting where their data is inaccessible.

We name our method as **Meta**-**D**istillation of **MoE** (Meta-DMoE). Our contributions are as follows:

- We propose a novel unsupervised test-time adaptation framework that is tailored for multiple sources domain settings. Our framework employs the concept of MoE to allow each expert model to explore each source domain thoroughly. We formulate the adaptation process as knowledge distillation via aggregating the positive knowledge retrieved from MoE.
- The alignment between training and evaluation objectives via meta-learning improves the adaptation, hence the test-time generalization.
- We conduct extensive experiments to show the superiority of the proposed method among the state-of-the-arts and validate the effectiveness of each component of Meta-DMoE.
- We validate that our method is more flexible in real-world settings where computational power and data privacy are the concerns.

## 2 Related work

**Domain shift.** Unsupervised Domain Adaptation (UDA) has been popular to address domain shift by transferring the knowledge from the labeled source domain to the unlabeled target domain [48, 41, 81]. It is achieved by learning domain-invariant features via minimizing statistical discrepancy across

domains [5, 58, 70]. Adversarial learning is also applied to develop indistinguishable feature space [26, 51, 57]. The first limitation of UDA is the assumption of the co-existence of source and target data, which is inapplicable when the target domain is unknown in advance. Furthermore, most of the algorithms focus on unrealistic single-source-single-target adaptation as source data normally come from multiple domains. Splitting the source data into various distinct domains and exploring the unique characteristics of each domain and the dependencies among them strengthen the robustness [89, 76, 78]. Domain generalization (DG) is another line of research to alleviate the domain shift. DG aims to train a model on multiple source domains without accessing any prior information of the target domain and expects it to perform well on unseen target domains. [28, 45, 53] aim to learn the domain-invariant feature representation. [63, 75] exploit data augmentation strategies in data or feature space. A concurrent work proposed bidirectional learning to mitigate domain shift [14]. However, deploying the generic model to all unseen target domains fails to explore domain specialty and yields sub-optimal solutions. In contrast, our method further exploits the unlabeled target data and updates the trained model to each specific unseen target domain at test time.

**Test-time adaptation (TTA).** TTA constructs supervision signals from unlabeled data to update the generic model before inference. Sun *et al.* [68] use rotation prediction to update the model during inference. Chi *et al.* [17] and Li *et al.* [46] reconstruct the input images to achieve internal-learning to better restore the blurry images and estimate the human pose. ARM [87] incorporates test-time adaptation with DG which meta-learns a model that is capable of adapting to unseen target domains before making an inference. Instead of adapting to every data sample, our method only updates once for each target domain using a fixed number of examples.

**Meta-learning.** The existing meta-learning methods can be categorised as model-based [62, 59, 8], metric-based [65, 30], and optimization-based [25]. Meta-learning aims to learn the learning process by episodic learning which is based on bi-level optimization ([13] provides a comprehensive survey). One of the advantages of bi-level optimization is to improve the training with conflicting learning objectives. Utilizing such paradigm, [16, 85] successfully reduce the forgetting issue and improve adaptation for continual learning [49]. In our method, we incorporate meta-learning with knowledge distillation by jointly learning a student model initialization and a knowledge aggregator for fast adaptation.

**Mixture-of-experts.** The goal of MoE [37] is to decompose the whole training set into many subsets, which are independently learned by different models. It has been successfully applied in image recognition models to improve the accuracy [1]. MoE is also popular in scaling up the architectures. As each expert is independently trained, sparse selection methods are developed to select a subset of the MoE during inference to increase the network capacity [42, 23, 29]. In contrast, our method utilizes all the experts to extract and combine the knowledge for positive knowledge transfer.

## 3 Preliminaries

In this section, we describe the problem setting and discuss the adaptive model. We mainly follow the test-time unsupervised adaptation as in [87]. Specifically, we define a set of $N$ source domains $\mathcal{D}_\mathcal{S} = \{{\mathcal{D}_\mathcal{S}}^i\}_{i=1}^N$ and $M$ target domains $\mathcal{D}_\mathcal{T} = \{{\mathcal{D}_\mathcal{T}}^j\}_{j=1}^M$. The exact definition of a domain varies and depends on the applications or data collection methods. It could be a specific dataset, user, or location. Let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ denote the input and the corresponding label, respectively. Each of the source domains contains data in the form of input-output pairs: ${\mathcal{D}_\mathcal{S}}^i = \{(x_\mathcal{S}^z, y_\mathcal{S}^z)\}_{z=1}^{Z_i}$. In contrast, each of the target domains contains only unlabeled data: ${\mathcal{D}_\mathcal{T}}^j = \{(x_\mathcal{T}^k)\}_{k=1}^{K_j}$. For well-designed datasets (e.g. [33, 20]), all the source or target domains have the same number of data samples. Such condition is not ubiquitous for real-world scenarios (i.e. $Z_{i_1} \neq Z_{i_2}$ if $i_1 \neq i_2$ and $K_{j_1} \neq K_{j_2}$ if $j_1 \neq j_2$) where data imbalance always exists [39]. It further challenges the generalization with a broader range of real-world distribution shifts instead of finite synthetic ones. Generic domain shift tasks focus on the out-of-distribution setting where the source and target domains are non-overlapping (i.e. $\mathcal{D}_\mathcal{S} \cap \mathcal{D}_\mathcal{T} = \varnothing$), but the label spaces of both domains are the same (i.e. $\mathcal{Y}_\mathcal{S} = \mathcal{Y}_\mathcal{T}$).

Conventional DG methods perform training on $\mathcal{D}_\mathcal{S}$ and make a minimal assumption on the testing scenarios [67, 3, 35]. Therefore, the same generic model is directly applied to all target domains $\mathcal{D}_\mathcal{T}$, which leads to non-optimal solutions [68]. In fact, for each ${\mathcal{D}_\mathcal{T}}^j$, some unlabeled data are readily available which provides certain prior knowledge for that target distribution. Adaptive Risk Minimization (ARM) [87] assumes that a batch of unlabeled input data $\mathbf{x}$ approximate the

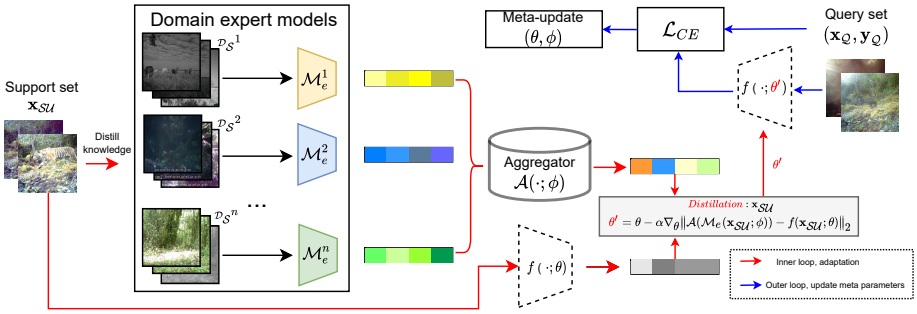



Figure 1: Overview of the training of Meta-DMoE. We first sample disjoint support set $\mathbf{x}_{\mathcal{SU}}$ and query set $(\mathbf{x}_{\mathcal{Q}}, \mathbf{y}_{\mathcal{Q}})$ from a training domain. $\mathbf{x}_{\mathcal{SU}}$ is sent to the expert models $\mathcal{M}$ to query their domain-specific knowledge. An aggregator $\mathcal{A}(\cdot;\phi)$ then combines the information and generates a supervision signal to update the $f(\cdot;\theta)$ via knowledge distillation. The updated $f(\cdot;\theta')$ is evaluated using the labeled query set to update the meta-parameters.

input distribution $p_x$ which provides useful information about $p_{y|x}$. Based on the assumption, an unsupervised test-time adaptation [59, 27] is proposed. The fundamental concept is to adapt the model to the specific domain using $\mathbf{x}$. Overall, ARM aims to minimize the following objective $\mathcal{L}(\cdot,\cdot)$ over all training domains:

$$\sum_{\mathcal{D}_{\mathcal{S}}{}^j \in \mathcal{D}_{\mathcal{S}}} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}_{\mathcal{S}}{}^j} \mathcal{L}(\mathbf{y}, f(\mathbf{x};\theta')), \text{where } \theta' = h(\mathbf{x}, \theta; \phi). \tag{1}$$

$\mathbf{y}$ is the labels for $\mathbf{x}$. $f(\mathbf{x};\theta)$ denotes the prediction model parameterized by $\theta$. $h(\cdot;\phi)$ is an adaptation function parameterized by $\phi$. It receives the original $\theta$ of $f$ and the unlabeled data $\mathbf{x}$ to adapt $\theta$ to $\theta'$.

The goal of ARM is to learn both $(\theta, \phi)$. To mimic the test-time adaptation (i.e., adapt before prediction), it follows the episodic learning as in meta-learning [25]. Specifically, each episode processes a domain by performing unsupervised adaptation using $\mathbf{x}$ and $h(\cdot;\phi)$ in the inner loop to obtain $f(\cdot;\theta')$. The outer loop evaluates the adapted $f(\cdot;\theta')$ using the true label to perform a meta-update. ARM is a general framework that can be incorporated with existing meta-learning approaches with different forms of adaptation module $h(\cdot;\cdot)$ [25, 27].

However, several shortcomings are observed with respect to the generalization. The episodic learning processes one domain at a time, which has clear boundaries among the domains. The overall setting is equivalent to the multi-source domain setting, which is proven to be more effective than learning from a single domain [53, 88] as most of the domains are correlated to each other [2]. However, it is counterintuitive to learn all the domain knowledge in one single model as each domain has specialized semantics or low-level features [64]. Therefore, the single-model method in ARM is sub-optimal due to: 1) some domains may contain competitive information, which leads to negative knowledge transfer [66]. It may tend to learn the ambiguous feature representations instead of capturing all the domain-specific information [80]; 2) not all the domains are equally important [76], and the learning might be biased as data in different domains are imbalanced in real-world applications [39].

# 4 Proposed approach

In this section, we explicitly formulate the test-time adaptation as a knowledge transfer process to distill the knowledge from MoE. The proposed method is learned via meta-learning to mimic the test-time out-of-distribution scenarios and ensure positive knowledge transfer.

## 4.1 Meta-distillation from mixture-of-experts

**Overview.** Fig. 1 shows the method overview. We wish to explicitly transfer useful knowledge from various source domains to achieve generalization on unseen target domains. Concretely, we define MoE as $\mathcal{M} = \{\mathcal{M}^i\}_{i=1}^N$ to represent the domain-specific models. Each $\mathcal{M}^i$ is separately trained using standard supervised learning on the source domain $\mathcal{D}_{\mathcal{S}}{}^i$ to learn its discriminative features.

We propose the test-time adaptation as the unsupervised knowledge distillation [34] to learn the knowledge from MoE. Therefore, we treat $\mathcal{M}$ as the teacher and aim to distill its knowledge to a student prediction network $f(\cdot;\theta)$ to achieve adaptation. To do so, we sample a batch of unlabeled $\mathbf{x}$ from a target domain, and pass it to $\mathcal{M}$ to query their domain-specific knowledge $\{\mathcal{M}^i(\mathbf{x})\}_{i=1}^N$. That knowledge is then forwarded to a knowledge aggregator $\mathcal{A}(\cdot;\phi)$. The aggregator is learned to capture the interconnection among domain knowledge aggregate information from MoE. The output of $\mathcal{A}(\cdot;\phi)$ is treated as the supervision signal to update $f(\mathbf{x};\theta)$. Once the adapted $\theta'$ is obtained, $f(\cdot;\theta')$ is used to make predictions for the rest of the data in that domain. The overall framework follows the effective few-shot learning where $\mathbf{x}$ is treated as unlabeled support set [74, 65, 25].

**Training Meta-DMoE.** Properly training $(\theta,\phi)$ is critical to improve the generalization on unseen domains. In our framework, $\mathcal{A}(\cdot,\phi)$ acts as a mechanism that explores and mixes the knowledge from multiple source domains. Conventional knowledge distillation process requires large numbers of data samples and learning iterations [34, 2]. The repetitive large-scale training is inapplicable in real-world applications. To mitigate the aforementioned challenges, we follow the meta-learning paradigm [25]. Such bi-level optimization enforces the $\mathcal{A}(\cdot,\phi)$ to learn beyond any specific knowledge [86] and allows the student prediction network $f(\cdot;\theta)$ to achieve fast adaptation. Specifically, We first split the data samples in each source domain ${\mathcal{D}_\mathcal{S}}^i$ into disjoint support and query sets. The unlabeled support set ($\mathbf{x}_{\mathcal{SU}}$) is used to perform adaptation via knowledge distillation, while the labeled query set ($\mathbf{x}_\mathcal{Q}$, $\mathbf{y}_\mathcal{Q}$) is used to evaluate the adapted parameters to explicitly test the generalization on unseen data.

**Algorithm 1** Training for Meta-DMoE

**Require:** $\{{\mathcal{D}_\mathcal{S}}^i\}_{i=1}^N$: data of source domains; $\alpha$, $\beta$: learning rates; $B$: meta batch size

1: **// Pretrain domain-specific MoE models**
2: **for** $i$=1,...,$N$ **do**
3: Train the domain-specific model $\mathcal{M}^i$ using ${\mathcal{D}_\mathcal{S}}^i$.
4: **end for**
5: **// Meta-train aggregator** $\mathcal{A}(\cdot;\phi)$ **and student model** $f(\cdot,\theta_e;\theta_c)$
6: **Initialize**: $\phi$, $\theta_e$, $\theta_c$
7: **while** *not converged* **do**
8: Sample a batch of $B$ source domains $\{{\mathcal{D}_\mathcal{S}}^b\}^B$, reset batch loss $\mathcal{L}_\mathcal{B}=0$
9: **for** each ${\mathcal{D}_\mathcal{S}}^b$ **do**
10: Sample support and query set: $(\mathbf{x}_{\mathcal{SU}})$, $(\mathbf{x}_\mathcal{Q},\mathbf{y}_\mathcal{Q})\sim{\mathcal{D}_\mathcal{S}}^b$
11: $\mathcal{M}_e'(\mathbf{x}_{\mathcal{SU}};\phi)=\{\mathcal{M}_e^i(\mathbf{x}_{\mathcal{SU}};\phi)\}_{i=1}^N$, mask $\mathcal{M}_e^i(\mathbf{x}_{\mathcal{SU}};\phi)$ with $\mathbf{0}$ if $b=i$
12: Perform adaptation via knowledge distillation from MoE:
13: $\theta_e'=\theta_e-\alpha\nabla_{\theta_e}\left\|\mathcal{A}(\mathcal{M}_e'(\mathbf{x}_{\mathcal{SU}};\phi))-f(\mathbf{x}_{\mathcal{SU}};\theta_e)\right\|_2$
14: Evaluate the adapted $\theta_e'$ using query set and accumulate the loss:
15: $\mathcal{L}_\mathcal{B}=\mathcal{L}_\mathcal{B}+\mathcal{L}_{CE}(\mathbf{y}_\mathcal{Q},f(\mathbf{x}_\mathcal{Q};\theta_e',\theta_c))$
16: **end for**
17: Update $\phi$, $\theta_e$, $\theta_c$ for the current meta batch:
18: $(\phi,\theta_e,\theta_c)\leftarrow(\phi,\theta_e,\theta_c)-\beta\nabla_{(\phi,\theta_e,\theta_c)}\mathcal{L}_\mathcal{B}$
19: **end while**

The student prediction network $f(\cdot;\theta)$ can be decoupled as a feature extractor $\theta_e$ and classifier $\theta_c$. Unsupervised knowledge distillation can be achieved via the softened output [34] or intermediate features [84] from $\mathcal{M}$. The former one allows the whole student network $\theta=(\theta_e,\theta_c)$ to be adaptive, while the latter one allows partial or complete $\theta_e$ to adapt to $\mathbf{x}$, depending on the features utilized. We follow [56] to only adapt $\theta_e$ in the inner loop while keeping the $\theta_c$ fixed. Thus, the adaptation process is achieved by distilling the knowledge via the aggregated features:

$$DIST(\mathbf{x}_{\mathcal{SU}},\mathcal{M}_e,\phi,\theta_e)=\theta_e'=\theta_e-\alpha\nabla_{\theta_e}\left\|\mathcal{A}(\mathcal{M}_e(\mathbf{x}_{\mathcal{SU}});\phi)-f(\mathbf{x}_{\mathcal{SU}};\theta_e)\right\|_2, \tag{2}$$

where $\alpha$ denotes the adaptation learning rate, $\mathcal{M}_e$ is the feature extractor of MoE models which extracts the features before the classifier, and $\|\cdot\|_2$ measures the $L_2$ distance. The goal is to obtain an updated $\theta_e'$ such that the extracted features of $f(\mathbf{x}_{\mathcal{SU}};\theta_e')$ is closer to the aggregated features. The overall learning objective of Meta-DMoE is to minimize the following expected loss:

$$\arg\min_{\theta_e,\theta_c,\phi}\sum_{{\mathcal{D}_\mathcal{S}}^j\in\mathcal{D}_\mathcal{S}}\sum_{\substack{(\mathbf{x}_{\mathcal{SU}})\in{\mathcal{D}_\mathcal{S}}^j\\(\mathbf{x}_\mathcal{Q},\mathbf{y}_\mathcal{Q})\in{\mathcal{D}_\mathcal{S}}^j}}\mathcal{L}_{CE}(\mathbf{y}_\mathcal{Q},f(\mathbf{x}_\mathcal{Q};\theta_e',\theta_c)),\text{where }\ \theta_e'=DIST(\mathbf{x}_{\mathcal{SU}},\mathcal{M}_e,\phi,\theta_e), \tag{3}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss. Alg. 1 demonstrates our full training procedure. To smooth the meta gradient and stabilize the training, we process a batch of episodes before each meta-update.

Since the training domains overlap for the MoE and meta-training, we simulate the test-time out-of-distribution by excluding the corresponding expert model in each episode. To do so, we multiply the features by $\mathbf{0}$ to mask them out. $\mathcal{M}_e'$ in L11 of Alg. 1 denotes such operation. Therefore, the adaptation is enforced to use the knowledge that is aggregated from other domains.

### 4.2 Fully learned explicit knowledge aggregator

Aggregating the knowledge from distinct domains requires capturing the relation among them to ensure the relevant knowledge transfer. Prior works design hand-engineered solutions to combine the knowledge or choose data samples that are closer to the target domain for knowledge transfer [2, 89]. A superior alternative is to replace the hand-designed pipelines with fully learned solutions [19, 9]. Thus we follow the same trend and allow the aggregator $\mathcal{A}(\cdot;\phi)$ to be fully meta-learned without heavy hand-engineering.

We observe that the self-attention mechanism is quite suitable where interaction among domain knowledge can be computed. Therefore, we use a transformer encoder as the aggregator [22, 73]. The encoder consists of multi-head self-attention and multi-layer perceptron blocks with layernorm [4] applied before each block, and residual connection applied after each block. We refer the readers to the appendix for the detailed architecture and computation. We concatenate the output features from the MoE models as $Concat[\mathcal{M}_e^1(\mathbf{x}), \mathcal{M}_e^2(\mathbf{x}), ..., \mathcal{M}_e^N(\mathbf{x})] \in \mathbb{R}^{N\times d}$, where $d$ is the feature dimension. The aggregator $\mathcal{A}(\cdot;\phi)$ processes the input tensor to obtain the aggregated feature $\mathbf{F} \in \mathbb{R}^d$, which is used as a supervision signal for test-time adaptation.

### 4.3 More constrained real-world settings

In this section, we investigate two critical settings for real-world applications that have drawn less attention from the prior works: limitation on computational resources and data privacy.

**Constraint on computational cost.** In real-world deployment environments, the computational power might be highly constrained (e.g., smartphones). It requires fast inference and compact models. However, the reduction in learning capabilities greatly hinders the generalization as some methods utilize only a single model regardless of the data complexity. On the other hand, when the number of domain data scales up, methods relying on adaptation on every data sample [87] will experience inefficiency. In contrast, our method only needs to perform adaptation once for every unseen domain. Only the final $f(\cdot;\theta')$ is used for inference. To investigate the impact on generalization caused by reducing the model size, we experiment with some lightweight network architectures (only $f(\cdot;\theta)$ for us) such as MobileNet V2 [61].

**Data privacy.** Large-scale training data are normally collected from various venues. However, some venues may have privacy regulations enforced. Their data might not be accessible but the models that are trained using private data are available. To simulate such an environment, we split the training source domains into two splits: private domains (${\mathcal{D}_\mathcal{S}}^{pri}$) and public domains (${\mathcal{D}_\mathcal{S}}^{pub}$). We use ${\mathcal{D}_\mathcal{S}}^{pri}$ to train MoE models and ${\mathcal{D}_\mathcal{S}}^{pub}$ for the subsequent meta-training. Since ARM and other methods only utilize the data as input, we train them on ${\mathcal{D}_\mathcal{S}}^{pub}$.

We conduct experiments to show the superiority of the proposed method in these more constrained real-world settings with computation and data privacy issues. For details on the settings, please refer to the appendix.

## 5 Experiements

### 5.1 Datasets and implementation details

**Datasets and evaluation metrics.** In this work, we mainly evaluate our method on the real-world domain shift scenarios. Drastic variation in deployment conditions normally exists in nature, such as a change in illumination, background, and time. It shows a huge domain gap between deployment environments and imposes challenges to the algorithm's robustness. Thus, we test our methods on the large-scale distribution shift benchmark WILDS [39], which reflects a diverse range of real-world distribution shifts. Following [87], we mainly perform experiments on five image testbeds, iWildCam [10], Camelyon17 [7],RxRx1 [69] and FMoW [18] and PovertyMap [83]. In each benchmark dataset, a domain represents a distribution of data that is similar in some way, such as images collected from the same camera trap or satellite images taken in the same location. We follow the same evaluation metrics as in [39] to compute severl metrics: accuracy, Macro F1, worst-case (WC) accuracy, Pearson correlation (r), and its worst-case counterpart. We also evaluate our method

Table 1: Comparison with the state-of-the-arts on the WILDS image testbeds and out-of-distribution setting. Metric means and standard deviations are reported across replicates. Our proposed method performs well across all problems and achieves the best results on 4 out of 5 datasets.

| | iWildCam | | Camelyon17 | RxRx1 | FMoW | | PovertyMap | |
|---|---|---|---|---|---|---|---|---|
| Method | Acc | Macro F1 | Acc | Acc | WC Acc | Avg Acc | WC Pearson r | Pearson r |
| ERM | 71.6 (2.5) | 31.0 (1.3) | 70.3 (6.4) | 29.9 (0.4) | 32.3 (1.25) | **53.0 (0.55)** | 0.45 (0.06) | 0.78 (0.04) |
| CORAL | 73.3 (4.3) | 32.8 (0.1) | 59.5 (7.7) | 28.4 (0.3) | 31.7 (1.24) | 50.5 (0.36) | 0.44 (0.06) | 0.78 (0.05) |
| Group DRO | 72.7 (2.1) | 23.9 (2.0) | 68.4 (7.3) | 23.0 (0.3) | 30.8 (0.81) | 52.1 (0.5) | 0.39 (0.06) | 0.75 (0.07) |
| IRM | 59.8 (3.7) | 15.1 (4.9) | 64.2 (8.1) | 8.2 (1.1) | 30.0 (1.37) | 50.8 (0.13) | 0.43 (0.07) | 0.77 (0.05) |
| ARM-CML | 70.5 (0.6) | 28.6 (0.1) | 84.2 (1.4) | 17.3 (1.8) | 27.2 (0.38) | 45.7 (0.28) | 0.37 (0.08) | 0.75 (0.04) |
| ARM-BN | 70.3 (2.4) | 23.7 (2.7) | 87.2 (0.9) | **31.2 (0.1)** | 24.6 (0.04) | 42.0 (0.21) | 0.49 (0.21) | **0.84 (0.05)** |
| ARM-LL | 71.4 (0.6) | 27.4 (0.8) | 84.2 (2.6) | 24.3 (0.3) | 22.1 (0.46) | 42.7 (0.71) | 0.41 (0.04) | 0.76 (0.04) |
| Ours (w/o mask) | **74.1 (0.4)** | **35.1 (0.9)** | **90.8 (1.3)** | 29.6 (0.5) | **36.8 (1.01)** | 50.6 (0.20) | **0.52 (0.04)** | 0.80 (0.03) |
| Ours | **77.2 (0.3)** | **34.0 (0.6)** | **91.4 (1.5)** | 29.8 (0.4) | **35.4 (0.58)** | 52.5 (0.18) | **0.51 (0.04)** | 0.80 (0.03) |

on popular benchmarks DomainNet [58] and PACS [44] from DomainBed [31] by computing the accuracy.

**Network architecture.** We follow WILDS [39] to use ResNet18 & 50 [32] or DenseNet101 [36] for the expert models $\{\mathcal{M}^i\}_{i=1}^N$ and student network $f(\cdot,;\theta)$. Also, we use a single-layer transformer encoder block[73] as the knowledge aggregator $\mathcal{A}(\cdot;\phi)$. To investigate the resource-constrained and privacy-sensitive scenarios, we utilize MobileNet V2 [61] with a width multiplier of 0.25. As for DomainNet and PACS, we follow the setting in DomainBed to use ResNet50 for both experts and student networks.

**Pre-training domain-specific models.** The WILDS benchmark is highly imbalanced in data size, and some classes have empty input data se. We found that using every single domain to train an expert is unstable, and sometimes it cannot converge. Inspired by [52], we propose to cluster the training domains into $N$ super domains and use each super-domain to train the expert models. Specifically, we set $N = \{10, 5, 3, 4, 3\}$ for iWildCam, Camelyon17, RxRx1, FMoW and Poverty Map, respectively. We use ImageNet [21] pre-trained model as the initialization and separately train the models using Adam optimizer [38] with a learning rate of $1e^{-4}$ and a decay of 0.96 per epoch.

**Meta-training and testing.** We first pre-train the aggregator and student network [15]. After that, the model is further trained using Alg. 1 for 15 epochs with a fixed learning rate of $3e^{-4}$ for $\alpha$ and $3e^{-5}$ for $\beta$. During meta-testing, we use Line 13 of Alg. 1 to adapt before making a prediction for every testing domain. Specifically, we set the number of examples for adaptation at test time = $\{24, 64, 75, 64, 64\}$ for iWildCam, Camelyon17, RxRx1, FMoW, and Poverty Map, respectively. For both meta-training and testing, we perform one gradient update for adaptation on the unseen target domain. We refer the readers to the supplementary materials for more detailed information.

### 5.2 Main results

**Comparison on WILDS.** We compare the proposed method with prior approaches showing on WILDS leaderboard [39], including non-adaptive methods: CORAL [67], ERM [72], IRM [3], Group DRO [60] and adaptive methods used in ARM [87] (CML, BN and LL). We directly copy the available results from the leaderboard or their corresponding paper. As for the missing ones, we conduct experiments using their provided source code with default hyperparameters. Table 1 reports the comparison with the state-of-the-art. Our proposed method performs well across all datasets and increases both worst-case and average accuracy compared to other methods. Our proposed method achieves the best performance on 4 out of 5 benchmark datasets. ARM [87] applies meta-learning approach to learn how to adapt to unseen domains with unlabeled data. However, their method is greatly bounded by using a single model to exploit knowledge from multiple source domains. Instead, our proposed method is more fitted to multi-source domain settings and meta-trains an aggregator that properly mixtures the knowledge from multiple domain-specific experts. As a result, our method outperforms ARM-CML, BN and LL by 9.5%, 9.8%, 8.1% for iWildCam, 8.5%, 4.8%, 8.5% for Camelyon17 and 14.8%, 25.0%, 22.9% for FMoW in terms of average accuracy. Furthermore, we also evaluate our method without masking the in-distribution domain in MoE models (Ours w/o mask) during meta-training (Line 10-11 of Alg. 1), where the sampled domain is overlapped with MoE. It violates the generalization to unseen target domains during testing. As most of the performance dropped, it reflects the importance of aligning the training and evaluation objectives.

Table 2: Evaluation on DomainNet. Our method performs the best on all experimental settings and outperforms recent SOTA significantly in terms of the average accuracy.

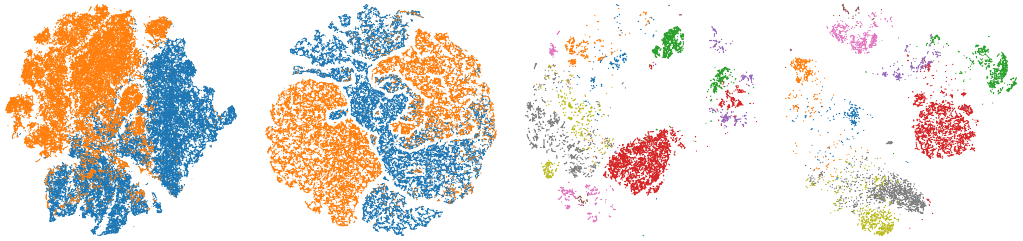
| Method | clip | info | paint | quick | real | sketch | avg |
|---|---|---|---|---|---|---|---|
| ERM [72] | 58.1 (0.3) | 18.8 (0.3) | 46.7 (0.3) | 12.2 (0.4) | 59.6 (0.1) | 49.8 (0.4) | 40.9 |
| IRM [3] | 48.5 (2.8) | 15.0 (1.5) | 38.3 (4.3) | 10.9 (0.5) | 48.2 (5.2) | 42.3 (3.1) | 33.9 |
| Group DRO [60] | 47.2 (0.5) | 17.5 (0.4) | 33.8 (0.5) | 9.3 (0.3) | 51.6 (0.4) | 40.1 (0.6) | 33.3 |
| Mixup [77] | 55.7 (0.3) | 18.5 (0.5) | 44.3 (0.5) | 12.5 (0.4) | 55.8 (0.3) | 48.2 (0.5) | 39.2 |
| MLDG [43] | 59.1 (0.2) | 19.1 (0.3) | 45.8 (0.7) | 13.4 (0.3) | 59.6 (0.2) | 50.2 (0.4) | 41.2 |
| CORAL [67] | 59.2 (0.1) | 19.7 (0.2) | 46.6 (0.3) | 13.4 (0.4) | 59.8 (0.2) | 50.1 (0.6) | 41.5 |
| DANN [26] | 53.1 (0.2) | 18.3 (0.1) | 44.2 (0.7) | 11.8 (0.1) | 55.5 (0.4) | 46.8 (0.6) | 38.3 |
| MTL [11] | 57.9 (0.5) | 18.5 (0.4) | 46.0 (0.1) | 12.5 (0.1) | 59.5 (0.3) | 49.2 (0.1) | 40.6 |
| SegNet [55] | 57.7 (0.3) | 19.0 (0.2) | 45.3 (0.3) | 12.7 (0.5) | 58.1 (0.5) | 48.8 (0.2) | 40.3 |
| ARM [87] | 49.7 (0.3) | 16.3 (0.5) | 40.9 (1.1) | 9.4 (0.1) | 53.4 (0.4) | 43.5 (0.4) | 35.5 |
| Ours | **63.5 (0.2)** | **21.4 (0.3)** | **51.3 (0.4)** | **14.3 (0.3)** | **62.3 (1.0)** | **52.4 (0.2)** | **44.2** |

Table 3: Evaluation on PACS. Our method outperforms other methods on 2 out of 4 experiments but still achieves the SOTA in terms of the average accuracy.

| Method | art | cartoon | photo | sketch | avg |
|---|---|---|---|---|---|
| ERM [72] | 84.7 (0.4) | 80.8 (0.6) | 97.2 (0.3) | 79.3 (1.0) | 85.5 |
| CORAL [67] | **88.3 (0.2)** | 80.0 (0.5) | **97.5 (0.3)** | 78.8 (1.3) | 86.2 |
| Group DRO [60] | 83.5 (0.9) | 79.1 (0.6) | 96.7 (0.3) | 78.3 (2.0) | 84.4 |
| IRM [3] | 84.8 (1.3) | 76.4 (1.1) | 96.7 (0.6) | 76.1 (1.0) | 83.5 |
| ARM [87] | 86.8 (0.6) | 76.8 (0.5) | 97.4 (0.3) | 79.3 (1.2) | 85.1 |
| Ours | 86.1 (0.2) | **82.5 (0.5)** | 96.7 (0.4) | **82.3 (1.4)** | **86.9** |

**Comparison on DomainNet and PACS.** Table 2 and Table 3 report the results on DomainNet and PACS. In DomainNet, our method performs the best on all experimental settings and outperforms recent SOTA significantly in terms of the average accuracy (+2.7). [82] has discovered that the lack of a large number of meta-training episodes leads to the meta-level overfitting/memorization problem. To our task, since PACS has 57$\times$ less number of images than DomainNet and 80$\times$ less number of domains than iWildCam, the capability of our meta-learning-based method is hampered by the less diversity of episodes. As a result, we outperform other methods in 2 out of 4 experiments but still achieve the SOTA in terms of average accuracy.

**Visualization of adapted features.** To evaluate the capability of adaptation via learning discriminative representations on unseen target domains, we compare the t-SNE [71] feature visualization using the same test domain sampled from iWildCam and Camelyon17 datasets. ERM utilizes a single model and standard supervised training without adaptation. Therefore, we set it as the baseline. Figure 2 shows the comparison, where each color denotes a class and each point represents a data sample. It is clear that our method obtains better clustered and more discriminative features.



(a) Camelyon17-ERM (b) Camelyon17-Ours (c) iWildCam-ERM (d) iWildCam-Ours

Figure 2: t-SNE visualization of adapted features at test-time. We directly utilize features adapted to the same unseen target domains from ERM and our proposed method in Camelyon17 and WildCam datasets, respectively. Our resulting features show more discriminative decision boundaries.

### 5.3 Results under constrained real-world settings

In this section, we mainly conduct experiments on iWildCam dataset under two real-world settings.

Table 4: Comparison of WILDS testbeds using MobileNet V2. Reducing the model size hampers the learning capability. Our method shows a better trade-off as the knowledge is distilled from MoE.

| | iWildCam | | Camelyon17 | RxRx1 | FMoW | | PovertyMap | |
|---|---|---|---|---|---|---|---|---|
| Method | Acc | Macro F1 | Acc | Acc | WC Acc | Avg Acc | WC Pearson r | Pearson r |
| ERM | 56.7 (0.7) | 17.5 (1.2) | 69.0 (8.8) | 14.3 (0.2) | 15.7 (0.68) | **40.0 (0.11)** | 0.39 (0.05) | 0.77 (0.04) |
| CORAL | **61.5 (1.7)** | 17.6 (0.1) | 75.9 (6.9) | 12.6 (0.1) | 22.7 (0.76) | 31.0 (0.32) | **0.44 (0.06)** | **0.79 (0.04)** |
| ARM-CML | 58.2 (0.8) | 15.8 (0.6) | 74.9 (4.6) | 14.0 (1.4) | 21.1 (0.33) | 30.0 (0.13) | 0.41 (0.05) | 0.76 (0.03) |
| ARM-BN | 54.8 (0.6) | 13.8 (0.2) | 85.6 (1.6) | 14.9 (0.1) | 17.9 (1.82) | 29.0 (0.69) | 0.42 (0.05) | 0.76 (0.03) |
| ARM-LL | 57.5 (0.5) | 12.6 (0.8) | 84.8 (1.7) | 15.0 (0.2) | 17.1 (0.22) | 30.3 (0.54) | 0.39 (0.07) | 0.76 (0.02) |
| Ours | 59.5 (0.7) | **19.7 (0.5)** | **87.1 (2.3)** | **15.1 (0.4)** | **26.9 (0.67)** | 37.9 (0.31) | **0.44 (0.04)** | 0.77 (0.03) |

**Constraint on computational cost.** Computational power is always limited in real-world deployment scenarios, such as edge devices. Efficiency and adaptation ability should be both considered. Thus, we replace our student model and the models in other methods with MobileNet V2. As reported in Table 4, our proposed method still outperforms prior methods. Since the MoE model is only used for knowledge transfer, our method is more flexible in designing the student architecture for different scenarios. We also report multiply-Accumulate operations (MACS) for inference and time complexity on adaptation. As ARM needs to make adaptations before inference on every example, its adaptation cost scales linearly with the number of examples. Our proposed method performs better in accuracy and requires much less computational cost for adaptation, as reported in Table 5.

**Constraint on data privacy.** On top of computational limitations, privacy-regulated scenarios are common in the real world. It introduces new challenges as the raw data is inaccessible. Our method does not need to access the raw data but the trained models, which greatly mitigates such regulation. Thus, as shown in Table 6, our method does not suffer from much performance degradation compared to other methods that require access to the private raw data.

Table 5: Adaptation efficiency evaluated on iWildCam using MobileNet V2. Our method not only outperforms prior methods but also keeps constant time complexity in test-time adaptation.

| Method | Acc / Macro-F1 | MACS | Complexity |
|---|---|---|---|
| ERM | 56.7 / 17.5 | $7.18 \times 10^7$ | N/A |
| ARM-CML | 58.2 / 15.8 | $7.73 \times 10^7$ | O(n) |
| ARM-LL | 57.5 / 12.6 | $7.18 \times 10^7$ | O(n) |
| Ours | 59.5 / 19.7 | $7.18 \times 10^7$ | O(1) |

Table 6: Results on privacy-related regulation setting evaluated on iWildCam and FMoW using MobileNet V2. Without privacy considered in the design, prior methods can only exploit public data and thus achieve far worse performance.

| | iWildCam | | FMoW | |
|---|---|---|---|---|
| Method | Acc | Macro-F1 | WC Acc | Acc |
| ERM | 51.2 | 11.2 | 22.5 | **35.4** |
| CORAL | 50.2 | 11.1 | 18.1 | 25.4 |
| ARM-CML | 42.7 | 7.5 | 16.8 | 24.1 |
| ARM-BN | 46.9 | 8.7 | 14.2 | 22.2 |
| ARM-LL | 46.8 | 9.3 | 13.7 | 22.6 |
| Ours | **54.7** | **14.2** | **24.4** | 33.8 |

### 5.4 Ablation studies

In this section, we conduct ablation studies on iWildCam to analyze various components of the proposed method. We also seek to answer the two key questions: 1) Does the number of experts affect the capability of capturing knowledge from multi-source domains? 2) Is meta-learning superior to standard supervised learning under the knowledge distillation framework?

**Number of domain-specific experts.** We investigate the impact of exploiting multiple experts to store domain-specific knowledge separately. Specifically, we keep the total number of data for experts pretraining fixed and report the results using a various number of expert models. The experiments in Table 7 validate the benefits of using more domain-specific experts.

Table 7: Results on the number of domain-specific experts. More experts increase the learning capacity to better explore each source domain, thus, improving generalization.

| # of experts | 2 | 5 | 7 | 10 |
|---|---|---|---|---|
| Accuracy | 70.4 | 74.1 | 76.4 | 77.2 |
| Macro-F1 | 30.6 | 32.3 | 33.7 | 34.0 |

**Training scheme.** To verify the effectiveness of meta-learning, we investigate three training schemes: random initialization, pre-train, and meta-train. To pre-train the aggregator, we add a classifier layer to its aggregated output and follow the standard supervised training scheme. For fair comparisons, we use the same testing scheme, including the number of updates and images for adaptation. Table 8 reports the results of different training scheme combinations. We observe that the randomly initialized student model

struggles to learn with only a few-shot data. And the pre-trained aggregator brings weaker adaptation guidance to the student network as the aggregator is not learned to distill. In contrast, our bi-level optimization-based training scheme enforces the aggregator to choose more correlated knowledge from multiple experts to improve the adaptation of the student model. Therefore, the meta-learned aggregator is more optimal (row 1 vs. row 2). Furthermore, our meta-distillation training process simulates the adaptation in testing scenarios, which aligns with the training objective and evaluation protocol. Hence, for both meta-trained aggregator and student models, it gains additional improvement (row 3 vs. row 4).

**Aggregator and distillation methods.** Table 9 reports the effects of various aggregators including two hand-designed operators: Max and Average pooling, and two MLP-based methods: Weighted sum (MLP-WS) and Projector (MLP-P) (details are provided in the supplement). We found that the fully learned transformer-based aggregator is crucial for mixing domain-specific features. Another important design choice in our proposed framework is in the form of knowledge: distilling the teacher model's logits, intermediate features, or both. We show evaluation results of those three forms of knowledge in Table 10.

Table 8: Evaluation of training schemes. Using both meta-learned aggregator and student model improves generalization as they are learned towards test-time adaptation.

| Train Scheme | | Metrics | |
|---|---|---|---|
| Aggregator | Student | Acc | Macro-F1 |
| Pretrain | Random | 6.2 | 0.1 |
| Meta | Random | 32.7 | 0.5 |
| Pretrain | Meta | 74.8 | 32.9 |
| Meta | Meta | 77.2 | 34.0 |

Table 9: Comparison between different aggregator methods. The transformer explores interconnection, which gives the best result.

| | Max | Ave. | MLP-WS | MLP-P | Trans.(ours) |
|---|---|---|---|---|---|
| Acc. | 69.2 | 69.7 | 70.7 | 73.7 | 77.2 |
| M-F1 | 29.2 | 25.0 | 32.8 | 32.7 | 34.0 |

Table 10: Comparison between different distillation methods. Distilling only the feature extractor yields the best generalization.

| | Logits | Logits + Feat. | Feat. (Ours) |
|---|---|---|---|
| Accuracy | 72.1 | 73.1 | 77.2 |
| Marco-F1 | 26.4 | 26.9 | 34.0 |

## 6 Discussion

We present Meta-DMoE, a framework for adaptation towards domain shift using unlabeled examples at test time. We formulate the adaptation as a knowledge distillation process and devise a meta-learning algorithm to guide the student network to fast adapt to unseen target domains via transferring the aggregated knowledge from multiple sources domain-specific models. We demonstrate that Meta-DMoE is state-of-the-art on four benchmarks. And it is competitive under two constrained real-world settings, including limited computational budget and data privacy consideration.

**Limitations.** As discussed in Section 5.4, Meta-DMoE can improve the capacity to capture complex knowledge from multi-source domains by increasing the number of experts. However, to compute the aggregated knowledge from domain-specific experts, every expert model needs to have one feed-forward pass. As a result, the computational cost of adaptation scales linearly with the number of experts. Furthermore, to add or remove any domain-specific expert, both the aggregator and the student network need to be re-trained. Thus, enabling a sparse-gated Meta-DMoE to encourage efficiency and scalability could be a valuable future direction, where a gating module determines a sparse combination of domain-specific experts to be used for each target domain.

**Social impact.** Tackling domain shift problems can have positive social impacts as it helps to elevate the model accuracy in real-world scenarios (e.g., healthcare and self-driving cars). In healthcare, domain shift occurs when a trained model is applied to patients in different hospitals. In this case, model performance might dramatically decrease, which leads to severe consequences. Tackling domain shifts helps to ensure that models can work well on new data, which can ultimately lead to better patient care. We believe our work is a small step toward the goal of adapting to domain shift.

## References

[1] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision*, 2016. 3

[2] Sk Miraj Ahmed, Dripta S Raychaudhuri, Sujoy Paul, Samet Oymak, and Amit K Roy-Chowdhury. Unsupervised multi-source domain adaptation without access to source data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 4, 5, 6

[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 3, 7, 8, 18

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 6, 19

[5] Mahsa Baktashmotlagh, Mehrtash Harandi, and Mathieu Salzmann. Distribution-matching embedding for visual domain adaptation. *Journal of Machine Learning Research*, 17:Article–number, 2016. 3

[6] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, 2018. 2

[7] Peter Bandi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, et al. From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge. *IEEE Transactions on Medical Imaging*, 2018. 6

[8] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 3

[9] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. In *European Conference on Artificial Intelligence*, 2020. 6

[10] Sara Beery, Elijah Cole, and Arvi Gjoka. The iwildcam 2020 competition dataset. *arXiv preprint arXiv:2004.10340*, 2020. 6

[11] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 2011. 8

[12] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *European Conference on Computer Vision*, 2020. 2

[13] Can Chen, Xi Chen, Chen Ma, Zixuan Liu, and Xue Liu. Gradient-based bi-level optimization for deep learning: A survey. *arXiv preprint arXiv:2207.11719*, 2022. 3

[14] Can Sam Chen, Yingxue Zhang, Jie Fu, Xue Liu, and Mark Coates. Bidirectional learning for offline infinite-width model-based optimization. *arXiv preprint arXiv:2209.07507*, 2022. 3

[15] Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. Meta-baseline: exploring simple meta-learning for few-shot learning. In *IEEE International Conference on Computer Vision*, 2021. 7

[16] Zhixiang Chi, Li Gu, Huan Liu, Yang Wang, Yuanhao Yu, and Jin Tang. Metafscil: A meta-learning approach for few-shot class incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 3

[17] Zhixiang Chi, Yang Wang, Yuanhao Yu, and Jin Tang. Test-time fast adaptation for dynamic scene deblurring via meta-auxiliary learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2, 3

[18] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 6

[19] Jeff Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, 2019. 6

[20] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *International Joint Conference on Neural Networks*, 2017. 3

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 7

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 6, 19

[23] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021. 3

[24] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *IEEE International Conference on Computer Vision*, 2013. 1, 18

[25] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Lachine Learning*, 2017. 2, 3, 4, 5

[26] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 1, 3, 8

[27] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, 2018. 4

[28] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *IEEE International Conference on Computer Vision*, 2015. 2, 3

[29] Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3

[30] Li Gu, Zhixiang Chi, Huan Liu, Yuanhao Yu, and Yang Wang. Improving protonet for few-shot video object recognition: Winner of orbit challenge 2022. *arXiv preprint arXiv:2210.00174*, 2022. 3

[31] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2020. 7, 15

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 7, 20

[33] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. 3

[34] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems*, 2015. 2, 5

[35] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, 2018. 3

[36] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 7, 20

[37] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. 3

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014. 7

[39] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, 2021. 1, 2, 3, 4, 6, 7, 15, 19, 20, 21

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 1

[41] Vinod K Kurmi, Venkatesh K Subramanian, and Vinay P Namboodiri. Domain impression: A source data free domain adaptation method. In *IEEE Winter Conference on Applications of Computer Vision*, 2021. 2

[42] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2020. 3

[43] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 8

[44] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *IEEE International Conference on Computer Vision*, 2017. 7

[45] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *European Conference on Computer Vision*, 2018. 3

[46] Yizhuo Li, Miao Hao, Zonglin Di, Nitesh Bharadwaj Gundavarapu, and Xiaolong Wang. Test-time personalization with a transformer for human pose estimation. *Advances in Neural Information Processing Systems*, 2021. 3

[47] Hanwen Liang, Niamul Quader, Zhixiang Chi, Lizhe Chen, Peng Dai, Juwei Lu, and Yang Wang. Self-supervised spatiotemporal representation learning by exploiting video continuity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 1

[48] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, 2020. 2

[49] Huan Liu, Li Gu, Zhixiang Chi, Yang Wang, Yuanhao Yu, Jun Chen, and Jin Tang. Few-shot class-incremental learning via entropy-regularized data-free replay. *arXiv preprint arXiv:2207.11213*, 2022. 3

[50] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, 2015. 1

[51] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, 2018. 1, 3, 18

[52] Daniela Massiceti, Luisa Zintgraf, John Bronskill, Lida Theodorou, Matthew Tobias Harris, Edward Cutrell, Cecily Morrison, Katja Hofmann, and Simone Stumpf. Orbit: A real-world few-shot dataset for teachable object recognition. In *IEEE International Conference on Computer Vision*, 2021. 7

[53] Toshihiko Matsuura and Tatsuya Harada. Domain generalization using a mixture of multiple latent domains. In *AAAI Conference on Artificial Intelligence*, 2020. 3, 4

[54] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, 2013. 2

[55] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap via style-agnostic networks. *arXiv preprint arXiv:1910.11645*, 2019. 8

[56] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. Boil: Towards representation change for few-shot learning. In *International Conference on Learning Representations*, 2021. 5

[57] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *AAAI Conference on Artificial Intelligence*, 2018. 3

[58] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *IEEE International Conference on Computer Vision*, 2019. 3, 7

[59] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. *Advances in Neural Information Processing Systems*, 2019. 3, 4

[60] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020. 7, 8

[61] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 6, 7

[62] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016. 3

[63] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018. 3

[64] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. 4

[65] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017. 3, 5

[66] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, 2020. 4

[67] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, 2016. 3, 7, 8, 18

[68] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, 2020. 2, 3

[69] J. Taylor, B. Earnshaw, B. Mabey, M. Victors, and J. Yosinski. Rxrx1: An image set for cellular morphological variation across many experimental batches. In *International Conference on Learning Representations*, 2019. 6

[70] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 3

[71] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008. 8

[72] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 1999. 7, 8

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 6, 7, 19

[74] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016. 5

[75] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems*, 2018. 3

[76] Junfeng Wen, Russell Greiner, and Dale Schuurmans. Domain aggregation networks for multi-source domain adaptation. In *International Conference on Machine Learning*, 2020. 3, 4

[77] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 8

[78] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3

[79] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *IEEE International Conference on Computer Vision*, 2019. 1

[80] Luyu Yang, Yogesh Balaji, Ser-Nam Lim, and Abhinav Shrivastava. Curriculum manager for source selection in multi-source domain adaptation. In *European Conference on Computer Vision*, 2020. 4

[81] Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. Generalized source-free domain adaptation. In *IEEE International Conference on Computer Vision*, 2021. 2

[82] Huaxiu Yao, Linjun Zhang, and Chelsea Finn. Meta-learning with fewer tasks through task interpolation. *arXiv preprint arXiv:2106.02695*, 2021. 8

[83] Christopher Yeh, Anthony Perez, Anne Driscoll, George Azzari, Zhongyi Tang, David Lobell, Stefano Ermon, and Marshall Burke. Using publicly available satellite imagery and deep learning to understand economic well-being in africa. *Nature Communications*, 2020. 6

[84] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 5

[85] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 3

[86] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 5

[87] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: Learning to adapt to domain shift. In *Advances in Neural Information Processing Systems*, 2021. 1, 2, 3, 6, 7, 8

[88] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*, 2018. 4

[89] Sicheng Zhao, Guangzhi Wang, Shanghang Zhang, Yang Gu, Yaxian Li, Zhichao Song, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. Multi-source distilling domain adaptation. In *AAAI Conference on Artificial Intelligence*, 2020. 3, 6

## 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] See Section 1.
   (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 supplemental material
   (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] See supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See 5.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5.2
   (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 supplemental material
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] See [39] [31]
   (b) Did you mention the license of the assets? [Yes] See supplemental material
   (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]

# A Additional Ablation Studies

In this section, we provide three additional ablation studies and discussions to further analyze our proposed method. These ablation studies are conducted on the iWildCam dataset.

## A.1 Aggregator Methods

In Table 9, we include several hand-designed aggregation operators: max-pooling, average-pooling, and two MLP-based learnable architectures. The two MLP-based learnable architectures work as follows.

MLP weighted sum (MLP-WS) takes the output features from the MoE models as input and produces the score for each expert. Then, we weigh those output features using the scores and sum them to obtain the final output for knowledge distillation.

For the MLP projector (MLP-P), the output features from the MoE are flattened at first $(N \times D \rightarrow ND \times 1)$ and then fed into an MLP architecture $(ND \times D, D \times D)$ to obtain the final output $(D \times 1)$ for knowledge distillation.

## A.2 Excluding Overlapping Expert

As discussed in Section 4.1, we simulate the test-time out-of-distribution by excluding the corresponding expert model in each episode since the training domains overlap for the MoE and meta-training. If the corresponding expert model is not excluded during meta-training, the aggregator output might be dominated by the corresponding expert output, or even collapse into a domain classification problem from the perspective of the aggregator. This might hinder the generalization on OOD domains. The experiments in Table 11 also validate the benefits of using such an operation.

## A.3 Expert Architecture

In this section, we analyze the effects of using a different expert architecture. Table 12 validates the benefits of using the knowledge aggregator and our proposed training algorithm. Our proposed method could perform robustly across different expert architectures.

Table 11: Comparison using ID test split in iWildCam. The ID test split contains images from the same domains as the training set but on different days from the training images. The model trained without masks performs better than the model trained with masks under the ID test split but has lower accuracy and a comparable Macro-F1 than the model trained with masks in the OOD test split.

| MoE Mask | ID Acc | ID Macro-F1 | OOD Acc | OOD Macro-F1 |
|---|---|---|---|---|
| Mask all except overlap | 75.5 | 46.8 | —- | —- |
| Without mask | 76.4 | 48.0 | 74.1 | 35.1 |
| With mask | 72.9 | 44.4 | 77.2 | 34.0 |

Table 12: Comparison with different expert architectures. Our proposed method is robust to different expert architectures with different capacities.

| Expert architecture | Student architecture | Acc | Macro-F1 |
|---|---|---|---|
| MobileNet V2 | MobileNet V2 | 59.5 | 19.7 |
| ResNet-50 | MobileNet V2 | 58.8 | 21.0 |

## A.4 Number of Images Used for Test-Time Adaptation

During deployment, our method uses a small number of unlabelled images to adapt the student prediction network to the target domain. Increasing the number of images used for adaptation might give a better approximation of the marginal of the target domain. Thus, the performance in the target domains is also enhanced. The experiments in Table 13 validate the benefits of using more images for adaptation.

Table 13: Results on the number of images for adaptation. Adaptation using more images leads to better approximations of the marginal and improves generalization.

| # of images for adaptation | 2 | 4 | 8 | 16 | 24 |
|---|---|---|---|---|---|
| Accuracy | 76.5 | 76.9 | 77.0 | 77.2 | 77.2 |
| Macro-F1 | 31.5 | 31.2 | 31.7 | 33.0 | 34.0 |

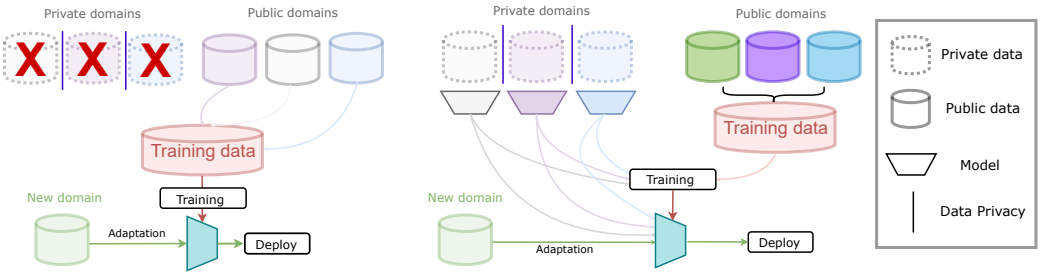



Figure 3: Left: Standard methods require sampling mini-batched data across domains and thus cannot utilize the locally-stored private data within each private domain. Right: Privacy-related algorithms can improve the adaptation results by transferring knowledge from the private data without access to the raw data.

# B Details on Privacy Constrained Setting

## B.1 Problem Definition

In this section, we discuss a problem setting where data privacy regulation is imposed. To achieve data diversity, large-scale labeled training data are normally collected from public venues (internet or among institutes) and stored in a server where i.i.d conditions can be satisfied to train a generic model by sampling mini-batches. However, in real-world applications, due to privacy-related regulations, some datasets cannot be shared among users or distributed edges. Such data can only be processed locally. Thus, they cannot be directly used for training a generalized model in most existing approaches [24, 51]. In this work, we consider a more realistic deployment problem with privacy constraints imposed.

We illustrate the privacy-regulated test-time adaptation setting in Fig. 3. To simulate the privacy-regulated scenario, we explicitly separate the distributed training source domains into two non-overlapping sets of domains: ${\mathcal{D}_\mathcal{S}}^{pri}$ for private domains and ${\mathcal{D}_\mathcal{S}}^{pub}$ for public domains. Each domain within ${\mathcal{D}_\mathcal{S}}^{pri}$ contains private data that can only be shared and accessed within that domain. Therefore, the data within ${\mathcal{D}_\mathcal{S}}^{pri}$ can only be accessed locally in a distributed manner during training, and cannot be seen at test time. ${\mathcal{D}_\mathcal{S}}^{pub}$ contains domains with only public data that has fewer restrictions and can be accessed from a centralized platform. Such splitting allows the use of ${\mathcal{D}_\mathcal{S}}^{pub}$ to simulate $\mathcal{D}_\mathcal{T}$ at training to learn the interaction with ${\mathcal{D}_\mathcal{S}}^{pri}$. It is also possible for some algorithms to mix all ${\mathcal{D}_\mathcal{S}}^{pub}$ and store them in a server to draw a mini-batch for every training iterations [67, 3], but such operation is not allowed for private data.

The ultimate goal under this privacy-regulated setting is to train a recognition model on domains ${\mathcal{D}_\mathcal{S}}^{pri}$ and ${\mathcal{D}_\mathcal{S}}^{pub}$ with the above privacy regulations applied. The model should perform well in the target domains $\mathcal{D}_\mathcal{T}$ without accessing either ${\mathcal{D}_\mathcal{S}}^{pri}$ or ${\mathcal{D}_\mathcal{S}}^{pub}$.

## B.2 Applying Meta-DMoE to Privacy Constrained Setting

Our proposed Meta-DMoE method is a natural solution to this setting. Concretely, for each private domain ${\mathcal{D}_\mathcal{S}}^{i,pri}$, we train an expert model $\mathcal{M}_e^i$ using only data from ${\mathcal{D}_\mathcal{S}}^{i,pri}$. After obtaining the domain-specific experts $\{\mathcal{M}_e^i\}$, we perform the subsequent meta-training on ${\mathcal{D}_\mathcal{S}}^{pub}$ to simulation OOD test-time adaptation. The training algorithm is identical to Alg. 1, except we don't mask any experts' output since the training domains for the MoEs and meta-training do not overlap. In this

way, we can leverage the knowledge residing in ${\mathcal{D}_{\mathcal{S}}}^{pri}$ without accessing the raw data but only the trained model on each domain during centralized meta-training. We also include the details of the experiments under this setting in Appendix D.1.

# C Details on Knowledge Aggregator

In this section, we discuss the detailed architecture and computation of the knowledge aggregator. We use a naive single-layer transformer encoder [73, 22] to implement the aggregator. The transformer encoder consists of multi-head self-attention blocks (MSA) and multi-layer perceptron blocks (MLP) with layernorm (LN) [4] applied before each block, and residual connection applied after each block. Formally, given the concatenated output features from the MoE models,

$$\mathbf{z}_0 = Concat[\mathcal{M}_e^1(\mathbf{x}), \mathcal{M}_e^2(\mathbf{x}), ..., \mathcal{M}_e^N(\mathbf{x})] \in \mathbb{R}^{N \times d}, \tag{4}$$

$$\mathbf{z}_0' = MSA_k(LN(\mathbf{z}_0)) + \mathbf{z}_0, \tag{5}$$

$$\mathbf{z}_{out} = MLP(LN(\mathbf{z}_0')) + \mathbf{z}_0', \tag{6}$$

where $MSA_k(\cdot)$ is the MSA block with $k$ heads and a head dimension of $d_k$ (typically set to $d/k$),

$$[\mathbf{q}, \mathbf{k}, \mathbf{v}] = \mathbf{z}\mathbf{W}_{qkv} \quad \mathbf{W}_{qkv} \in \mathbb{R}^{d \times 3 \cdot d_k}, \tag{7}$$

$$SA(\mathbf{z}) = Softmax(\frac{\mathbf{q}\mathbf{k}^T}{\sqrt{d_k}})\mathbf{v}, \tag{8}$$

$$MSA_k(\mathbf{z}) = Concat[SA_1(\mathbf{z}), ..., SA_k(\mathbf{z})]\mathbf{W}_o \quad \mathbf{W}_o \in \mathbb{R}^{k \cdot D_k \times D} . \tag{9}$$

We finally average-pool the transformer encoder output $\mathbf{z}_{out} \in \mathbb{R}^{N \times d}$ along the first dimension to obtain the final output. In the case when the dimensions of the features outputted by the aggregator and the student are different, we apply an additional MLP layer with layernorm on $\mathbf{z}_{out}$ to reduce the dimensionality as desired.

# D Additional Experimental Details

We run all the experiments using a single NVIDIA V100 GPU. The official WILDS dataset contains training, validation, and testing domains which we use as source, validation target, and test target domains. The validation set in WILDS [39] contains held-out domains with labeled data that are non-overlapping with training and testing domains. To be specific, we first use the training domains to pre-train expert models and meta-train the aggregator and the student prediction model and then use the validation set to tune the hyperparameters of meta-learning. At last, we evaluate our method with the test set. We include the official train/val/test domain split in the following subsections. We run each experiment and report the average as well as the unbiased standard deviation across three random seeds unless otherwise noted. In the following subsections, we provide the hyperparameters and training details for each dataset below. For all experiments, we select the hyperparameters settings using the validation split on the default evaluation metrics from WILDS. For both meta-training and testing, we perform one gradient update for adaptation on the unseen target domain.

## D.1 Details for Privacy Constrained Evaluation

We mainly perform experiments under privacy constrained setting on two subsets of WILDS for image recognition tasks, iWildCam and FMoW. To simulate the privacy constrained scenarios, we randomly select 100 domains from iWildCam training split as ${\mathcal{D}_{\mathcal{S}}}^{pri}$ to train $\{\mathcal{M}_e^i\}_{i=1}^M$ and the rest as ${\mathcal{D}_{\mathcal{S}}}^{pub}$ to meta-train the knowledge aggregator and student network. As for FMoW, we randomly select data from 6 years as ${\mathcal{D}_{\mathcal{S}}}^{pri}$ and the rest as ${\mathcal{D}_{\mathcal{S}}}^{pub}$. The domains are merged into 10 and 3 super-domains, respectively, as discussed in Section 5.1. Since ARM and other methods only utilize the data as input, we train them on only ${\mathcal{D}_{\mathcal{S}}}^{pub}$.

### D.2 IWildCam Details

IWildCam is a multi-class species classification dataset, where the input $x$ is an RGB photo taken by a camera trap, the label $y$ indicates one of 182 animal species, and the domain $z$ is the ID of the camera trap. During training and testing, the input $x$ is resized to 448 $\times$ 448. The train/val/test set contains 243/32/48 domains, respectively.

**Evaluation.** Models are evaluated on the Macro-F1 score, which is the F1 score across all classes. According to [39], Macro-F1 score might better describe the performance on this dataset as the classes are highly imbalanced. We also report the average accuracy across all test images.

**Training domain-specific model.** For this dataset, we train 10 expert models where each expert is trained on a super-domain formed by 24-25 domains. The expert model is trained using a ResNet-50 model pretrained on ImageNet. We train the expert models for 12 epochs with a batch size of 16. We use Adam optimizer with a learning rate of 3e-5.

**Meta-training and testing.** We train the knowledge aggregator using a single-layer transformer encoder with 16 heads. The transformer encoder has an input and output dimension of 2048, and the inner layer has a dimension of 4096. We use ResNet-50 [32] model for producing the results in Table 1. We first train the aggregator and student network with ERM until convergence for faster convergence speed during meta-training. After that, the models are meta-trained using Alg. 1 with a learning rate of 3e-4 for $\alpha$, 3e-5 for $\beta_s$, 1e-6 for $\beta_a$ using Adam optimizer, and decay of 0.98 per epoch. Note that we use a different meta learning rate, $\beta_a$ and $\beta_s$ respectively, for the knowledge aggregator and the student network as we found it more stable during meta training. In each episode, we first uniformly sample a domain, and then use 24 images in this domain for adaptation and use 16 images to query the loss for meta-update. We train the models for 15 epochs with early stopping on validation Macro-F1. During testing, we use 24 images to adapt the student model to each domain.

### D.3 Camelyon Details

This dataset contains 450,000 lymph node scan patches extracted from 50 whole-slide images (WSIs) with 10 WSIs from each of 5 hospitals. The task is to perform binary classification to predict whether a region of tissue contains tumor tissue. Under this task specification, the input $x$ is a 96 by 96 scan patch, the label $y$ indicates whether the central region of a patch contains tumor tissue, and the domain $z$ identifies the hospital. The train/val/test set contains 30/10/10 WSIs, respectively.

**Evaluation.** Models are evaluated on the average accuracy across all test images.

**Training domain-specific model.** For this dataset, we train 5 expert models where each expert is trained on a super-domain formed by 6 WSIs since there are only 3 hospitals in the training split. The expert model is trained using a DenseNet-121 model from scratch. We train the expert models for 5 epochs with a batch size of 32. We use an Adam optimizer with a learning rate of 1e-3 and an $L2$ regularization of 1e-2.

**Meta-training and testing.** We train the knowledge aggregator using a single-layer transformer encoder with 16 heads. The knowledge aggregator has an input and output dimension of 1024, and the inner layer has a dimension of 2048. We use DenseNet-121 [36] model for producing the results in Table 1. We first train the aggregator until convergence, and the student network is trained from ImageNet pretrained. After that, the models are meta-trained using Alg. 1 with a learning rate of 1e-3 for $\alpha$, 1e-4 for $\beta_s$, 1e-3 for $\beta_a$ using Adam optimizer and a decay of 0.98 per epoch for 10 epochs. In each episode, we first uniformly sample a WSI, and then use 64 images in this WSI for adaptation and use 32 images to query the loss for meta-update. The model is trained for 10 epochs with early stopping. During testing, we use 64 images to adapt the student model to each WSI.

### D.4 RxRx1 Details

The task is to predict 1 of 1,139 genetic treatments that cells received using fluorescent microscopy images of human cells. The input $x$ is a 3-channel fluorescent microscopy image, the label $y$ indicates which of the treatments the cells received, and the domain $z$ identifies the experimental batch of the image. The train/val/test set contains 33/4/14 domains, respectively.

**Evaluation.** Models are evaluated on the average accuracy across all test images.

**Training domain-specific model.** For this dataset, we train 3 expert models where each expert is trained on a super-domain formed by 11 experiments. The expert model is trained using a ResNet-50 model pretrained from ImageNet. We train the expert models for 90 epochs with a batch size of 75. We use an Adam optimizer with a learning rate of 1e-4 and an $L2$ regularization of 1e-5. We follow [39] to linearly increase the learning rate for the first 10 epochs and then decrease it using a cosine learning rate scheduler.

**Meta-training and testing.** We train the knowledge aggregator using a single-layer transformer encoder with 16 heads. The knowledge aggregator has an input and output dimension of 2048, and the inner layer has a dimension of 4096. We use the ResNet-50 model to produce the results in Table 1. We first train the aggregator and student network with ERM until convergence. After that, the models are meta-trained using Alg. 1 with a learning rate of 1e-4 for $\alpha$, 1e-6 for $\beta_s$, 3e-6 for $\beta_a$ using Adam optimizer and following the cosine learning rate schedule for 10 epochs. In each episode, we use 75 images from the same domain for adaptation and use 48 images to query the loss for meta-update. During testing, we use 75 images to adapt the student model to each domain.

### D.5 FMoW Details

FMoW is comprised of high-definition satellite images from over 200 countries based on the functional purpose of land in the image. The task is to predict the functional purpose of the land captured in the image out of 62 categories. The input $x$ is an RBD satellite image resized to 224 $\times$ 224, the label $y$ indicates which of the categories that the land belongs to, and the domain $z$ identifies both the continent and the year that the image was taken. The train/val/test set contains 55/15/15 domains, respectively.

**Evaluation.** Models are evaluated by the average accuracy and worst-case (WC) accuracy based on geographical regions.

**Training domain-specific model.** For this dataset, we train 4 expert models where each expert is trained on a super-domain formed by all the images in 2-3 years. The expert model is trained using a DenseNet-121 model pretrained from ImageNet. We train the expert models for 20 epochs with a batch size of 64. We use an Adam optimizer with a learning rate of 1e-4 and a decay of 0.96 per epoch.

**Meta-training and testing.** We train the knowledge aggregator using a single-layer transformer encoder with 16 heads. The knowledge aggregator has an input and output dimension of 1024, and the inner layer has a dimension of 2048. We use the DenseNet-121 model to produce the results in Table 1. We first train the aggregator and student network with ERM until convergence. After that, the models are meta-trained using Alg. 1 with a learning rate of 1e-4 for $\alpha$, 1e-5 for $\beta_s$, 1e-6 for $\beta_a$ using Adam optimizer and a decay of 0.96 per epoch. In each episode, we first uniformly sample a domain from $\{\text{continent} \times \text{year}\}$, and then use 64 images from this domain for adaptation and use 48 images to query the loss for meta-update. We train the models for 30 epochs with early stopping on validation WC accuracy. During testing, we use 64 images to adapt the student model to each domain.

### D.6 Poverty Details

The task is to predict the real-valued asset wealth index using a multispectral satellite image. The input $x$ is an 8-channel satellite image resized to 224 $\times$ 224, the label $y$ is a real-valued asset wealth index of the captured location, and the domain $z$ identifies both the country that the image was taken and whether the area is urban or rural. For this dataset, we use MSE Loss for training the domain-specific experts and meta-training. The train/val/test set contains 26-28/8-10/8-10 domains, respectively. The number of domains varies slightly from the fold to the fold for Poverty.

**Evaluation.** Models are evaluated by the Pearson correlation (r) and worst-case (WC) r based on urban/rural sub-populations. This dataset is split into 5 folds where each fold defines a different set of Out-of-Distribution (OOD) countries. The results are aggregated over 5 folds.

**Training domain-specific model.** For this dataset, we train 3 expert models where each expert is trained on a super-domain formed by 4-5 countries. The expert model is trained using a ResNet-18 model from scratch. We train the expert models for 70 epochs with a batch size of 64. We use an Adam optimizer with a learning rate of 1e-3 and a decay of 0.96 per epoch.

**Meta-training and testing.** We train the knowledge aggregator using a single-layer transformer encoder with 16 heads. The knowledge aggregator has an input and output dimension of 512, and the inner layer has a dimension of 1024. We use the ResNet-18 model to produce the results in Table 1. We first train the aggregator and student network with ERM until convergence. After that, the models are meta-trained using Alg. 1 with a learning rate of 1e-3 for $\alpha$, 1e-4 for $\beta_s$, 1e-4 for $\beta_a$ using Adam optimizer and a decay of 0.96 per epoch. In each episode, we first uniformly sample a domain from $\{\text{country} \times \text{urban/rural}\}$, and then use 64 images from this domain for adaptation and use 64 images to query the loss for meta-update. We train the models for 100 epochs with early stopping on validation Pearson r. During testing, we use 64 images to adapt the student model to each domain.