AdMiT: Adaptive Multi-Source Tuning in Dynamic Environments

Xiangyu Chang[1], Fahim Faisal Niloy[1], Sk Miraj Ahmed[1,2]*, Srikanth V. Krishnamurthy[1], Basak Guler[1], Ananthram Swami[4], Samet Oymak[3], Amit Roy-Chowdhury[1]
[1]University of California, Riverside [2]Brookhaven National Laboratory
[3]University of Michigan, Ann Arbor [4]DEVCOM Army Research Laboratory
{cxian008@, fnilo001@, sahme047@, krish@cs, basakg@, amitrc@ece.}ucr.edu
ananthram.swami.civ@army.mil, oymak@umich.edu

Abstract

Incorporating transformer models into edge devices poses a significant challenge due to the computational demands of adapting these large models across diverse applications. Parameter-efficient tuning (PET) methods (e.g. LoRA, Adapter, Visual Prompt Tuning, etc.) allow for targeted adaptation by modifying only small parts of the transformer model. However, adapting to dynamic unlabeled target distributions at the test time remains complex. To address this, we introduce AdMiT: Adaptive Multi-Source Tuning in Dynamic Environments. AdMiT innovates by pre-training a set of PET modules, each optimized for different source distributions or tasks, and dynamically selecting and integrating a sparse subset of relevant modules when encountering a new, few-shot, unlabeled target distribution. This integration leverages Kernel Mean Embedding (KME)-based matching to align the target distribution with relevant source knowledge efficiently, without requiring additional routing networks or hyperparameter tuning. AdMiT achieves adaptation with a single inference step, making it particularly suitable for resource-constrained edge deployments. Furthermore, AdMiT preserves privacy by performing an adaptation locally on each edge device, without the need for data exchange. Our theoretical analysis establishes guarantees for AdMiT's generalization, while extensive benchmarks demonstrate that AdMiT consistently outperforms other PET methods across a range of tasks, achieving robust and efficient adaptation.

1. Introduction

Pretrained transformers [1–5] have achieved remarkable success across diverse tasks, but their large parameter counts—often reaching billions [2, 5]—present challenges for deployment, especially on edge devices with limited computational resources. To address these limitations, parameter-efficient tuning (PET) methods, such as prefix/prompt tuning [6–9], adapters [10], and LoRA [11], have been introduced. These methods allow the pretrained model to remain fixed while only adjusting a small set of additional parameters tailored to specific target distributions, significantly reducing memory and computation needs while often achieving performance comparable to that of full fine-tuning.

Most existing PET methods are either single-source—focusing on a single PET module trained for one distribution—or, when incorporating multiple PET modules, require additional computational resources such as routing networks [12] or extensive hyperparameter tuning [13, 14]. These approaches lack the ability to directly integrate knowledge from multiple PET modules, each trained on different source distributions. In *dynamically evolving environments*, adaptation methods benefit from leveraging multiple sources of pre-trained knowledge. Instead of relying on a single PET module trained on a single source, integrating multiple PET modules enables more robust adaptation to shifting distributions by drawing from a diverse set of source-specific knowledge[15]. This multi-source approach is particularly advantageous when access to the original source data used for training each module is restricted due to privacy, storage, or other constraints. In such scenarios, training a unified PET module across combined sources is infeasible, making it both practical and effective to adaptively employ and integrate an array of pre-trained PET modules during test time, resulting in performance improvements often unattainable with single-source PET adaptation.

In this work, we introduce **AdMiT** (Adaptive Multi-Source Tuning in Dynamic Environments), a novel framework designed to efficiently adapt pre-trained PET modules across multiple dynamic distributions. AdMiT pre-trains a structured set of PET modules, each specifically tuned to a different source distribution, providing a versatile foundation for multi-source adaptation. During test time, when faced with new, small-batch target data from a new distri-

*Work done while the author was at University of California, Riverside.

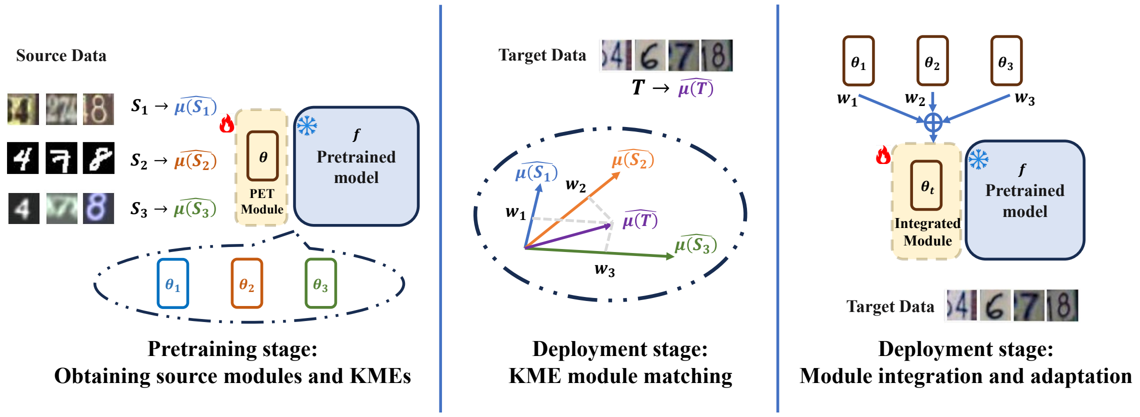


Figure 1. The diagram illustrates the AdMiT workflow, which includes pretraining source modules, matching modules during deployment, and updating the integrated module. In the **pretraining stage**, given a loss function $\mathcal{L}$ and source distributions $\{\mathcal{D}_{\mathcal{S}|}\}_{j=1}^{N}$, we freeze the base model f and fine-tune each module θ_j for its respective distribution. We also map the source data to an embedding space $\mathcal{H}$ as empirical Kernel Mean Embeddings (KMEs) $\{\widehat{\mu(S_j)}\}_{j=1}^{N}$. During the **KME module matching stage (Sec. 3.1)**, AdMiT maps the target data T to an empirical KME $\widehat{\mu(T)}$ and approximates it as a weighted combination of source KMEs, determining the weight coefficients $\hat{w}_j$. In the **module integration and adaptation stage (Sec. 3.2)**, AdMiT integrates the source modules with the highest weights to create an adaptive module $\theta(t)$ for the current target batch. This module $\theta(t)$ can then be directly applied to the target distribution or further adapted using sharpness-aware pseudo-label minimization for enhanced alignment with the target data.

bution, AdMiT (1) selects a relevant subset of PET modules based on their alignment with the target distribution, and (2) integrates these selected modules into a newly composed module optimized for the current target batch. This integrated module can then be directly applied to the target distribution, achieving efficient adaptation with lower computational overhead compared to existing PET methods [12, 14, 16]. Additionally, this integrated module enables further fine-tuning if desired, allowing AdMiT to dynamically enhance its alignment with the target distribution. This dual capability—of zero-shot applicability and optional on-the-fly adaptation—enables AdMiT to adapt robustly to evolving target distributions with low computational cost. Our experiments demonstrate AdMiT's effectiveness in both zero-shot and test-time adaptation settings, highlighting its adaptability and strong performance across dynamic distributions.

AdMiT offers two key advantages. First, by dynamically matching and adapting multiple pre-trained PET modules to the target distribution using small batches of target data, AdMiT achieves superior performance over traditional single-source PET adaptation methods. The ability to integrate multiple PET modules enables AdMiT to more effectively capture complex target distributions by leveraging a diverse set of source-specific knowledge through Kernel Mean Embedding (KME)-based matching (Sec. 3.1). Second, AdMiT bypasses the need for additional hyperparameter tuning or routing network training during deployment by using KME to align the target with relevant source distributions. This approach eliminates the computational burden of full model inference, allowing for fast multi-source PET adaptation. As a result, AdMiT is particularly well-suited for resource-constrained edge deployments, where both efficiency and flexibility are essential. Moreover, this KME-based distribution matching ensures data privacy, as no raw data exchange is required during adaptation.

Main Contributions. We present a new multi-source PET approach, AdMiT, that enables edge devices to selectively integrate a minimal subset of PET modules from a pre-trained collection, adapting in real-time to new, unlabeled data in an unsupervised, few-shot setting. Our contributions include the following:

- **Adaptive Multi-source Module Selection and Integration.** AdMiT efficiently selects and integrates a subset of pre-trained PET modules from a structured collection, based on the distributional characteristics of incoming target data. This adaptive integration avoids the computational burden associated with training routing networks or hyperparameter tuning for each new target distribution. By selectively combining relevant modules, AdMiT achieves performance comparable to that obtained by using all modules simultaneously but with significantly reduced storage and computational requirements, making it practical for edge deployment.
- **Efficient, Privacy-preserving Adaptation.** Unlike existing PET methods that utilize additional routing networks or data-alignment steps for multi-source adaptation [12], AdMiT achieves adaptation by transforming empirical data distributions in a kernel embedding space [17, 18]. This approach avoids the need to exchange raw data, preserving data privacy and reducing computational costs. Moreover,

AdMiT performs efficient module selection and combination without additional inference steps, enabling real-time adaptation on edge devices.

- **Theoretical Guarantees.** We provide theoretical guarantees on AdMiT's generalization performance, showing how it can effectively balance the sample sizes of source modules and the target batch size to ensure reliable multi-source adaptation. This guarantee highlights that well-trained source modules can provide robust adaptation even with limited target data.
- **Comprehensive Empirical Evaluation and Insights.** Extensive evaluations on challenging datasets, including Digit-Five, CIFAR-100C, and ImageNet-C, demonstrate that AdMiT consistently surpasses existing PET methods across various adaptation distributions. AdMiT shows a notable improvement in accuracy when adapting to new distributions (Table 1) and effectively preserves knowledge from source distributions(Table 2), showcasing strong performance in adaptation and retention. Additionally, AdMiT effectively identifies and integrates the most relevant modules(Figure 3) consistently achieving optimal results with minimal computational overhead. Additional results on the segmentation task, using Cityscapes [19] and ACDC [20] datasets, are provided in the Appendix, demonstrating AdMiT 's effectiveness in handling dynamic distributions across different tasks.

2. Related Works

Parameter Efficient Tuning (PET). Large-scale pre-trained models have greatly enhanced performance in natural language processing [1] and computer vision [21] by transferring learned knowledge to downstream tasks. PET methods, such as prompt tuning [21] and adapters [10], allow efficient adaptation by fine-tuning a small subset of parameters. Techniques like CoOp [22] and CoCoOp [23] leverage prompt optimization for out-of-distribution generalization, while CLIP-Adapter [24] and Tip-Adapter [25] fine-tune CLIP using adapters or key-value cache models, improving adaptability to target distributions. However, most PET methods are not designed to handle continuously shifting small-batch target distributions effectively; they adapt independently to each distribution, creating distribution-specific PET modules and often forgetting previously learned information. In contrast, our method enables consistent adaptation to dynamic target distributions while preserving knowledge of the pre-trained source distributions, addressing a gap in existing PET approaches. Our approach is compatible with various PET modules, including LoRA [11], VPT [21], and adapters [10].

Test Time Adaptation (TTA). Unsupervised Domain Adaptation (UDA) requires extensive target distribution data for offline adaptation, whereas TTA operates continuously on incoming test batches [26–28]. Initial TTA approaches [29] used test-batch statistics rather than training data, with methods like TENT [30] updating batch-normalization parameters to reduce entropy on target data. DUA [31] further refines alignment with target distributions by persistently updating batch-norm statistics across test batches. While these single-source TTA methods are effective, they often struggle with forgetting source knowledge over time, particularly in dynamic settings. Approaches like CoTTA and BeCoTTA [32, 33] use stochastic source restoration to mitigate drift, and EATA [34] employs regularization to preserve critical parameters, thus reducing forgetting. However, these methods often adapt to each batch separately, requiring substantial computational resources to balance adaptation and forgetting. In contrast, our multi-source approach dynamically matches relevant source modules to the target distribution, integrating them efficiently and minimizing forgetting with minimal computational overhead.

Ensemble Learning and Multi-source Adaptation. Ensemble learning, a well-known technique, enhances model robustness by combining outputs from various models [35]. Techniques like SESoM [12] and mixture models [14] aim to handle dynamic target distributions by combining multiple pre-trained models or PET modules. However, due to privacy constraints or storage limitations, direct access to source data from pre-trained models or PET modules is often unavailable. This limitation requires ensemble methods to rely on additional hyperparameter tuning [14, 16] or routing networks [12] to match the target and source distributions, leading to substantial computational overhead and frequent forward inference. For CNN-based applications, such adjustments are manageable [15, 36], but in the context of large pre-trained transformers, this tuning becomes prohibitively costly. Our approach bypasses these constraints by performing source-target matching through Kernel Mean Embeddings (KMEs), enabling efficient and privacy-preserving adaptation without requiring raw data or extensive computation, making it well-suited for large-scale pre-trained models in dynamic environments.

3. Proposed Method: AdMiT

Our method, AdMiT, leverages a pretrained transformer model f along with a collection of parameter-efficient tuning (PET) modules, each pretrained on distinct source tasks or domains. These PET modules, represented by parameters $\{\theta_j\}_{j=1}^N$, have significantly fewer parameters than the base model f and can be flexibly integrated into f as needed, thus forming a structured repository of source knowledge. We denote the transformer f combined with a module θ as f_θ. The core idea of AdMiT is to optimize adaptation to a target task or domain by selectively blending these pretrained modules based on their distributional representations in an embedding space and combining their weights to maximize relevance. The adaptation to target distribution is shown in figure 1.

Algorithm 1 AdMiT: Adaptive Multi-Source Tuning in Dynamic Environments.

```
1: Input: Pretrained transformer model $f$, Pretrained source modules $\{\theta_j\}_{j=1}^N$, empirical KME of source modules $\{\widehat{\mu(S_j)}\}_{j=1}^N$, number of modules to be selected $M$ ($M < N$), streaming sequential unlabeled test data $T^{(1)} = \{x_i^{(1)}\}_{i=1}^{|T|} \to T^{(2)} = \{x_i^{(2)}\}_{i=1}^{|T|} \to \dots T^{(t)} = \{x_i^{(t)}\}_{i=1}^{|T|} \to \dots$
2: Output: $M$ scaled weights, finetuned new module $\theta(t)$, size of synthetic dataset $Z$
3: Use Alg. A (in the Appendix) to generate synthetic data and estimate $\{\widehat{\mu(S_j)}\}_{j=1}^N$
4: while $t \geq 1$ do
5:     for Each $x_i$ in the $t$-th batch do
6:         Calculate the empirical KME of the target batch (Eqn. 3)
7:     end for
8:     Obtain mixture weights $\{\hat{w}(t)^j\}_{j=1}^N$ by solving Eqn. 4
9:     Find and select the top $M$ in $\{\hat{w}(t)^j\}_{j=1}^N$
10:    Rescale selected weights to sum up to 1, thus obtain $\{\overline{w}(t)^j\}_{j=1}^M$.
11:    Create a new module $\theta(t)$ by a weighted averaging of the selected pretrained modules
       $\theta(t) = \sum_{j=1}^M \overline{w}(t)^j \theta_j$
12:    Finetune $f_{\theta(t)}$ with Eqn. 8
13: end while
```

- **Pretraining stage: Pretraining and KME calculation.** Given a loss function $\mathcal{L}$ and a set of source tasks or domains $\{\mathcal{D}_{\mathcal{S}j}\}_{j=1}^N$, we freeze the base transformer model f and only update the PET modules, rather than fully fine-tuning f on each domain. For each source $\mathcal{D}_{\mathcal{S}j}$, we optimize the parameters θ_j as $\theta_j = \text{argmin}_\theta \mathcal{L}(f_\theta; \mathcal{D}_{\mathcal{S}j})$. To capture the source distributions, we map the source data to an embedding space $\mathcal{H}$ and represent the empirical distributions using Kernel Mean Embeddings (KMEs) $\{\widehat{\mu(S_j)}\}_{j=1}^N$.
- **Deployment stage: KME Module matching. (Sec. 3.1)** In this stage, AdMiT maps the target data T to an empirical Kernel Mean Embedding (KME) $\widehat{\mu(T)}$ and approximates it as a linear combination of source KMEs $\{\widehat{\mu(S_j)}\}_{j=1}^N$, expressed as $\widehat{\mu(T)} = \sum_{j=1}^N w_j \widehat{\mu(S_j)}$. This approach to multi-source distribution estimation, commonly applied in previous works [15, 16, 36], enables us to interpret the mixture weights $\{w_j\}$ as relevance scores for each source module. These weights guide the selection and integration of source modules, ensuring that the target is adapted efficiently and effectively. We also provide a theoretical bound on the estimation error for this approximation.
- **Deployment stage: Module integration and adaptation. (Sec. 3.2)** Using the computed mixture weights $\hat{w}_j$, AdMiT selects the source modules with the highest weights and integrates them to create a combined module θ. This integrated module θ is subsequently fine-tuned on the target domain $\mathcal{D}_\mathcal{T}$ to refine its alignment with the target distribution. For further enhancement, we apply sharpness-aware pseudo-label minimization [37, 38] to adjust the ensemble module and improve its robustness on the target domain.

Throughout the deployment stage, only a small number of unlabeled target samples are required to identify and integrate the source modules most relevant to the current target distribution. A detailed pseudocode for AdMiT can be found in Algorithm 1. In the following sections, we provide an in-depth explanation of the principles guiding the design of AdMiT.

3.1. Module Matching using KME

In the pretraining stage, given the heterogeneous feature spaces across different models, we assume a unified feature space facilitated by a public feature extractor $G(\cdot)$, which maps the original data $\boldsymbol{x}'$ from both source and target distributions into a shared representation space $\boldsymbol{x} = G(\boldsymbol{x}')$. This setup is practical, as publicly available pre-trained models can serve as feature extractors. In our experiments, we use a DenseNet201 model [39] pre-trained on ImageNet for this purpose. Since source data are inaccessible during the deployment stage, we require a metric to assess the similarity between source and target distributions without exchanging data or performing forward model inference.

Kernel Mean Embedding (KME) [40–43] provides a powerful tool for measuring distribution similarity. KME maps probability distributions into vectors in a high-dimensional Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ using a positive semi-definite bounded kernel $0 \leq k(\cdot,\cdot) \leq K$, simplifying the similarity evaluation of two distributions to inner product calculations in RKHS. Given a distribution $\mathcal{P}$ of an $\mathcal{X}$-valued random variable, its KME is defined as:

$$\mu_k(\mathcal{P}) := \int_{x\in\mathcal{X}} k(x,\cdot)\, dP(x). \quad (1)$$

The norm of the KME in $\mathcal{H}$ can be expressed by the inner product:

$$||\mu_k(\mathcal{P})||_\mathcal{H}^2 := \langle \mu_k(\mathcal{P}), \mu_k(\mathcal{P})\rangle = \mathbf{E}_{x,y\sim\mathcal{P}} k(x,y). \quad (2)$$

Since true distributions $\mathcal{P}$ are typically unknown, we estimate the KME and its norm using a finite batch $X = \{x_n\}_{n=1}^{|X|} \sim \mathcal{P}$:

$$\widehat{\mu(X)} := \frac{1}{|X|}\sum_{n=1}^{|X|} k(x_n,\cdot), \quad (3)$$

$$||\widehat{\mu(X)}||^2_{\mathcal{H}} := \frac{1}{|X|^2} \sum_{x_i, x_j \in X} k(x_i, x_j).$$

These empirical KMEs are computed on source datasets $\{S_j\}_{j=1}^N \sim \mathcal{D}_{\mathcal{S}j}$ during the pretraining stage, and on the target batch $T \sim \mathcal{D}_{\mathcal{T}}$ in the deployment stage.

In the deployment stage, we assume the target distribution can be approximated as a linear combination of source distributions, such that $\mathcal{D}_{\mathcal{T}} \approx \sum_{j=1}^N w_j \mathcal{D}_{\mathcal{S}j}$ for some mixture weights $\{w_j\}_{j=1}^N$. Using the linearity of expectation, we can express the KME of the target as $\mu_{\mathcal{D}_{\mathcal{T}}} \approx \sum_{j=1}^N w_j \mu_{\mathcal{D}_{\mathcal{S}j}}$. Each source KME $\{\widehat{\mu(S_j)}\}_{j=1}^N$ serves as a basis in the Hilbert space $\mathcal{H}$, allowing us to decompose the target empirical KME $\widehat{\mu(T)}$ using these bases. By solving the following optimization, we obtain the mixture weights $\{\hat{w}_j\}_{j=1}^N$ to match source and target distributions:

$$\min_{\{w_j\}_{j=1}^N} \left\| \widehat{\mu(T)} - \sum_{j=1}^N w_j \widehat{\mu(S_j)} \right\|_{\mathcal{H}}. \tag{4}$$

This KME-based approach serves as a reliable metric for distribution similarity (See Section H), enabling efficient and privacy-preserving matching between source and target distributions.

Theorem 3.1. *For a bounded kernel $0 \le k(\cdot,\cdot) \le K$, with probability at least $1-\delta$, the (biased) empirical MMD (obtained by drawing m samples from $p = \mathcal{D}_{\mathcal{T}}$ and n samples from $q = \sum_{j=1}^N w_j \mathcal{D}_{\mathcal{S}j}$, with $\sum_{j=1}^N w_j = 1$) is bounded by: (Proof in the Appendix Corollary H.3.)*

$$\frac{1}{2} \left\| \widehat{\mu(T)} - \sum_{j=1}^N w_j \widehat{\mu(S_j)} \right\|_{\mathcal{H}} < \sqrt{\frac{K}{m}} + \sqrt{\frac{K}{n}} + \sqrt{\frac{K(m+n)\log\frac{1}{\delta}}{2mn}}.$$

The mixture weight solution $\hat{w}_j$ from optimization 4 also implies the similarity of the distribution of source domain $\mathcal{D}_{\mathcal{S}j}$ and the target domain $\mathcal{D}_{\mathcal{T}}$, leading to the module selection strategy in Alg. 1.

Practical Considerations. In real-world applications, not all source modules are closely aligned with the target distribution, and calculating Kernel Mean Embeddings (KMEs) of source data in Eqn. 4 can be computationally intensive, as it involves summing up to $|S_j|$ kernel functions for each source. To enhance efficiency and reduce computational overhead, we employ the following strategies: **(1)** instead of using all source modules during adaptation, we select only the modules with the highest weights $\hat{w}_j$ (as shown in Alg. 1), thus focusing on the most relevant sources, and **(2)** to approximate each source KME $\widehat{\mu(S_j)}$, we generate a smaller synthetic dataset $\{z_m\}_{m=1}^Z$ (where $Z \ll |S_j|$). This synthetic dataset enables efficient computation by reducing the number of kernel functions involved, and it preserves privacy by eliminating the need for raw data exchange during KME decomposition [44]. Details of the synthetic data generation algorithm for source KMEs are provided in the Appendix, Alg. A.

Using synthetic datasets to approximate KMEs offers two major advantages. First, direct access to original source data is often restricted due to privacy or storage constraints, making it necessary to rely on the accessible information from source modules. Generating synthetic data allows us to create accurate KME approximations for each source module in a privacy-preserving way. Second, synthetic datasets provide a computationally efficient alternative to direct KME calculations using source data. By involving fewer data points, synthetic KMEs significantly reduce the computational load for matching the target distribution with source KMEs, making adaptation feasible even in resource-limited settings. We have theoretically demonstrated that (see Appendix Alg. A) the synthetic KMEs closely approximate the one calculated from the raw data. This fidelity ensures that the adaptation performance remains reliable and robust, as shown in our experiments, and enables efficient yet effective alignment of target and source distributions.

3.2. Module Integration and Adaptation

Drawing inspiration from the benefits of a good initialization in test-time adaptation and transfer learning [12, 16, 36, 45], we integrate PET modules trained on distributions related to the target distribution, as these modules are presumed to contain valuable knowledge relevant to these distributions. This integration is achieved through a weighted mixture of the selected modules, aiming to achieve a transfer gain:

$$\theta(t) = \sum_{i=1}^M \bar{w}_i \theta_i,$$

where $\bar{w}_i$ is obtained from Alg. 1, and $\theta(t)$ represents the integrated module. Directly applying the integrated module on the target distribution results in a zero-shot adaptation, whose performance can be bounded by the following theorem.

Theorem 3.2 (Zero-shot adaptation loss bound)**.** *Assume that the source training error is at most ϵ; the loss $L(f_{\hat{\theta}_j}(x), f(x)) \in \mathcal{H}_k$; and the empirical MMD between $\sum_{j=1}^N w_j \mathcal{D}_{\mathcal{S}j}$ and $\mathcal{D}_{\mathcal{T}}$ is from Theorem 3.1. Then, the finite-sample loss is:*

$$\mathcal{L}(\mathcal{D}_{\mathcal{T}}, g, f_{\sum \boldsymbol{w}_j \hat{\theta}_j}) = \sum_{x_i \sim \mathcal{D}_{\mathcal{T}}} \left[L(f_{\sum \boldsymbol{w}_j \hat{\theta}_j}(x_i), g(x_i)) \right]$$
$$\le \epsilon + O(\sqrt{\frac{1}{m}} + \sqrt{\frac{1}{n}})$$

Proof can be found in the Appendix, Theorem H.5.

To further boost performance, we fine-tune $\theta(t)$ for the target distribution $\mathcal{D_T}^{(t)}$ at time t:

$$\theta(t)^* = \arg\min_{\theta(t)} \mathcal{L}(f_{\theta(t)}; \mathcal{D_T}^{(t)}).$$

Starting from the integrated PET module $\theta(t)$ offers an efficient initialization, as the fine-tuning does not incur additional forward inference costs even as the number of candidate modules N or selected modules M increases.

Practical Considerations. Although test-time adaptation (TTA) can stabilize the adapted models, it risks model collapse during the tuning process, where the model may incorrectly classify all inputs as belonging to a single category over time [38]. To mitigate this, we incorporate sharpness-aware techniques [37, 38] to make the model less sensitive to large gradients that may arise from test samples [34].

Once we obtain the new module $\theta(t)$ for the target batch $T^{(t)} = \{x_i^{(t)}\}_{i=1}^{|T|}$ at time step t, we compute the entropy of the pseudo-labels predicted by the model with this module. The entropy of the predictions for the t-th target batch from the model $f_{\theta(t)}$ is:

$$\mathcal{L}^{(t)} = -\mathbf{E}_{\mathcal{D}_T^{(t)}} \sum_{c=1}^{K} \hat{y}_c^{(t)} \log(\hat{y}_c^{(t)}), \tag{5}$$

To properly fine-tune the new module $\theta(t)$ with this pseudo-label entropy minimization, we aim to make the model insensitive to large gradients by encouraging convergence to a flat region of the entropy loss surface. This approach, which seeks a flat minimum, provides good generalization and robustness against large gradients [37, 38]:

$$\min_{\lambda} \mathcal{L}^{SA(t)}(\{x_i^{(t)}\}_{i=1}^{B}; \lambda), \tag{6}$$

$$\text{where } \mathcal{L}^{SA(t)} \triangleq \max_{\|\epsilon\|_2 \leq \rho} \mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B}; \lambda + \epsilon) \tag{7}$$

"SA" denotes sharpness-aware. The gradient for this optimization can be approximated (see Appendix A for details):

$$\nabla_\lambda \mathcal{L}^{SA(t)} \approx \nabla_\lambda \mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B}; \lambda)|_{\lambda+\epsilon^*(\lambda)}. \tag{8}$$

Applying Eqn. 8 instead of standard SGD to update the parameters of $\theta(t)$ based on Eqn. 5 results in a more robust solution for pseudo-label entropy minimization. The effect of sharpness-aware adaptation is discussed further in the ablation study.

4. Evaluations

In our experiments, we evaluate AdMiT 's effectiveness by adapting PET modules pre-trained on source distributions to target data drawn from stationary or dynamically evolving distributions (See experiment setting details in Appendix Sec. F). The target distributions involve the same task as the source but differ due to distribution shifts relative to the source distributions on which the PET modules were trained.

Table 1. **Static Adaptation on ImageNet-C.** Following a similar experiment setup in Fig. 2, we adapt to a target corruption domain by taking the rest $15 - 1 = 14$ domains as source domains, given varying target batch size. Due to space limitations, we report only the averaged accuracy across all target domains.

Source	**Method**	**BS=256**	**BS=128**	**BS=64**	**BS=16**	**BS=1**
Single	TENT-Best [30]	52.2	52.3	52.0	52.4	51.7
	TENT-Worst [30]	34.7	34.5	35.3	34.7	31.6
	BECoTTA-Best [33]	60.4	61.5	62.0	61.1	55.4
	BECoTTA-Worst [33]	35.3	36.4	37.9	37.3	30.4
	SAR-Best [38]	58.1	62.3	61.4	60.3	54.1
	SAR-Worst [38]	37.5	38.6	38.2	38.1	31.1
	GT-Tuning	**67.7**	**68.8**	**69.7**	**65.4**	**60.3**
Multi	π-tuning-PL [16]	61.4	62.4	62.7	61.5	57.1
	SESoM-PL [12]	62.3	62.6	62.5	62.0	56.3
	CONTRAST [15]	63.5	63.1	<u>63.1</u>	61.7	57.9
	Model soup [46]	52.3	53.4	52.2	51.7	49.5
	AdMiT	<u>63.8</u>	<u>63.7</u>	62.4	<u>62.3</u>	<u>58.7</u>
	AdMiT-ZeroShot	60.2	59.6	58.4	55.9	53.3
	AdMiT-Plain	63.5	62.2	62.1	61.0	56.6

We consider two adaptation scenarios: (1) a *static adaptation setting* where the target data are drawn from a stationary distribution, and (2) a *dynamic adaptation setting* where the target data are drawn sequentially from an evolving distribution. The former scenario demonstrates AdMiT 's effectiveness in adapting to a stationary target distribution using pre-trained source modules, while the latter highlights AdMiT 's robustness in adapting to evolving target distributions over time.

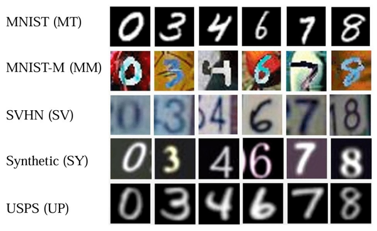
Datasets. For the static adaptation scenario, we evaluate AdMiT on the *Digits-Five* dataset [47], which includes five digit datasets—MNIST (MT), MNIST-M (MM), USPS (UP), SVHN (SV), and Synthetic Digits (SY)—each covering 10 classes (0-9). In these experiments, four distributions are used as sources, with the remaining one reserved for testing. We also use the *ImageNet-C* dataset [48], which applies 15 types of severe corruptions (details in Appendix Sec. G) to ImageNet images [49], following the setup in [38].

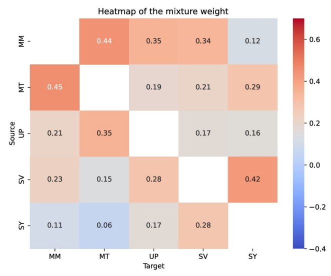
For the dynamic adaptation scenario, we utilize the *CIFAR-100C* benchmark [48], which extends the CIFAR-100 dataset [50] by introducing 15 types of noise at varying levels of severity (1 to 5). This setup results in up to 75 distinct distributions, allowing us to assess AdMiT 's capacity for continuous adaptation as the target distribution evolves.

Finally, although our primary evaluation focuses on image classification tasks, our method is not limited to this setting. It can be extended to other tasks, such as semantic segmentation, with results for segmentation tasks provided in the Appendix E.

4.1. Baseline Methods

Our evaluation includes comparisons with state-of-the-art (SOTA) single-source test-time adaptation (TTA) methods,





Source	**Method**	**MM**	**MT**	**UP**	**SV**	**SY**	**Avg**
Single	TENT-Best [30]	56.1	98.4	84.9	87.0	95.2	84.3
	TENT-Worst [30]	17.6	54.2	59.6	11.4	15.5	31.7
	BECoTTA-Best [33]	52.3	97.6	85.7	86.4	95.6	83.5
	BECoTTA-Worst [33]	22.6	53.8	60.7	24.9	23.6	37.1
	SAR-Best [38]	57.8	97.3	86.2	86.4	92.0	83.9
	SAR-Worst [38]	36.9	57.3	62.9	40.4	35.8	46.6
	GT-Tuning	**70.4**	**99.6**	**93.1**	**90.5**	**97.2**	**90.1**
Multi	π-tuning-PL [16]	61.3	98.2	89.1	86.2	94.6	85.8
	SESoM-PL [12]	62.1	98.3	88.6	87.1	94.5	86.0
	CONTRAST [15]	63.1	97.6	88.5	<u>87.2</u>	94.3	86.1
	Model soup [46]	52.3	83.2	71.5	66.4	75.2	69.7
	AdMiT	<u>63.2</u>	<u>98.9</u>	<u>89.3</u>	86.5	<u>96.2</u>	<u>86.8</u>
	AdMiT-ZeroShot	54.5	97.2	86.3	80.5	92.3	82.2
	AdMiT-Plain	61.3	96.3*	87.4	85.3	93.1	84.7

Figure 2. **Static adaptation on Digits-Five. (Left):** Sample images from the source and target domains used in the adaptation task, which include MNIST (MT), MNIST-M (MM), SVHN (SV), Synthetic (SY), and USPS (UP). **(Center):** Heatmap depicting the mixture weights assigned to various source modules during adaptation to the target domain. Larger mixture weights ($\boldsymbol{w}_j$) indicate a higher similarity between the target and source domains. The weights in each column sum to 1, as the four remaining domains are used to adapt to the target domain. **(Right):** We train the source modules using 4 digits datasets to perform adaptation on the remaining dataset. All the results are the average of 5 runs. **Best** performance is bolded, and <u>second-best</u> performance is underlined. The table clearly demonstrates that the average accuracy of AdMiT outperforms other baselines and is closest to the performance achieved by tuning with ground-truth labels. We also report the module integration (without tuning) results as AdMiT-ZeroShot, and the module adapation using plain SGD tuning results as AdMiT-Plain.

Source	Method	GN	SN	IN	DB	FGB	MB	ZB	**Snow**	**Frost**	**Fog**	**Bright**	Contrast	Elastic	Pixel	JPEG	Avg
Single	TENT-Best [30]	74.2	65.3	51.0	50.7	47.3	45.8	42.0	36.9	35.2	24.3	24.6	14.2	11.9	11.9	8.2	36.2
	SAR-Best [38]	71.2	71.7	69.2	64.4	56.0	54.2	59.8	59.3	56.2	52.7	49.4	46.8	48.2	42.0	44.9	56.4
	BECoTTA-Best [33]	69.7	71.4	65.8	68.7	54.3	53.4	55.7	52.4	55.7	47.2	52.4	46.3	49.6	42.4	41.9	55.1
Multi	π-tuning-PL [16]	**79.8**	**76.3**	**81.1**	<u>75.7</u>	66.2	62.5	68.1	61.6	61.9	58.7	63.4	52.9	50.7	53.1	56.0	64.5
	SESoM-PL [12]	75.0	<u>76.0</u>	<u>74.8</u>	69.0	65.6	60.4	60.3	54.5	55.8	54.5	55.3	49.7	50.8	51.4	53.9	60.5
	Model soup [46]	52.1	55.3	49.6	48.0	48.7	49.1	43.2	<u>73.0</u>	<u>76.5</u>	<u>74.1</u>	74.3	56.3	51.2	47.8	58.3	57.2
	CONTRAST [15]	78.2	75.3	74.4	**76.1**	**72.2**	<u>70.8</u>	<u>71.7</u>	72.1	73.9	71.4	<u>74.6</u>	<u>71.5</u>	<u>69.5</u>	<u>71.3</u>	**65.4**	<u>72.5</u>
	AdMiT	<u>79.1</u>	74.4	74.4	75.2	<u>69.2</u>	**71.8**	**73.7**	**73.1**	**78.9**	**75.2**	**74.9**	**73.2**	**70.5**	**72.8**	<u>65.0</u>	**73.3**

Table 2. **Dynamic adaptation forgetting evaluation on CIFAR-100C**. We take $N = 4$ source modules pretrained on *Snow*, *Frost*, *Fog*, and *Bright* for all the involved methods in the table. The table illustrates the <u>average</u> test accuracy (with all corruption domains of severity level 5) on the <u>4 source domains</u> during a sequential adaptation across different target domains for various methods. AdMiT selects $M = 3$ modules to adapt to new domains. All the results are the average of 5 runs. We employ these models for adaptation on 15 sequential target domains. **Best** performance is bolded, and <u>second-best</u> performance is underlined.

such as TENT [30], BECoTTA [33], and SAR [38]. These methods serve as benchmarks for adapting individual source models to target distributions, providing insight into how well a single-source approach performs in adapting to unseen distributions. As our problem setting involves adapting pre-trained models to new, dynamic distributions during deployment, it is closely related to the objectives of TTA. Therefore, these widely recognized single-source TTA methods are natural baselines for evaluating AdMiT 's ability to adapt effectively. Following a setup similar to that in [36], we apply each source model independently to specific test distribution data, reporting <u>**Best**</u> and <u>**Worst**</u> results, corresponding to the highest and lowest performance achieved across individual source models.

We also compare against leading multi-source ensemble methods in both static adaptation and dynamic adaptation settings, as these approaches simultaneously leverage multiple sources and provide a baseline for assessing the benefits of multi-source adaptation. SESoM [12] trains an attention-based routing network for adaptive weighting across source outputs, while π-tuning [16] fine-tunes hyperparameters based on a weighted mix of source modules, and CONTRAST [15] computes optimal weights for combining multiple source model outputs through gradient descent. These multi-source methods allow us to evaluate the effectiveness of combining knowledge from multiple distributions and highlight the computational trade-offs involved. For fair comparison, we implement pseudo-label (PL) entropy minimization for tuning mixture weights in SESoM and π-tuning, denoting these as SESoM-PL and π-tuning-PL, and apply a greedy model soup approach to minimize PL entropy by averaging module mixtures, following [46]. All baseline methods, including those originally based on CNN architectures, have been reproduced on a transformer architecture for consistency in our comparisons. Additional implementation details are provided in the Appendix.

Lastly, we provide an upper-bound baseline, GT-Tuning, which tunes a new PET module using ground-truth labels, offering insight into the best achievable performance with full label access on the target distribution. Together, these base-

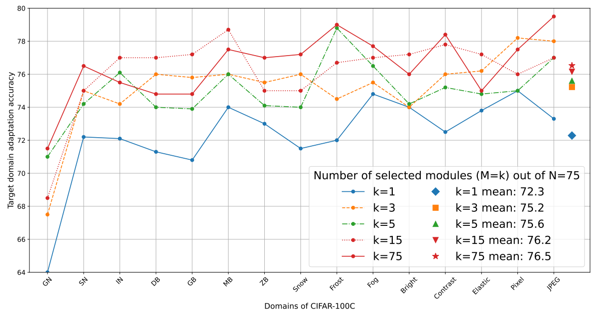


Figure 3. **Module Selection on CIFAR-100C.** Performance of AdMiT on various domains of CIFAR-100C with different numbers of selected source modules. We pretrain a set of 75 modules (each for a corruption domain and severity level) and select top-k modules based on empirical KME weights. $M = k$ indicates the number of selected source modules. Results show that with limited target data (batch size=128), selecting just a few modules ($k > 1$) maintains performance comparable to that from using all modules ($k = 75$). Mean performance across domains (shown as markers) improves with more modules but with diminishing returns, demonstrating our method's efficiency even with significantly fewer modules.

lines capture both single-source and multi-source strategies, illustrating AdMiT's effectiveness in adapting to dynamic target distributions efficiently and without label access.

Module adaptation. We evaluate AdMiT on the Digits-Five [47] dataset for digit classification, with $N = 4$ source modules and all $M = N$ modules used for inference on each target distribution. As shown in Figure 2, GT-Tuning achieves the best performance (serving as an upper bound with labeled data), while AdMiT achieves the second best results (underlined) in most target distributions. The accompanying heatmap illustrates the average weights assigned to each source module, with higher weights corresponding to greater similarity between target and source distributions. We also include results from *AdMiT-Plain*, which uses plain SGD instead of sharpness-aware tuning for adaptation. In some cases (star-marked cells), plain SGD leads to decreased performance, underscoring that sharpness-aware adaptation provides more stable tuning results.

Module integration. We also assess the performance of the integrated module without any tuning to gauge the efficiency of AdMiT in a zero-shot adaptation setting, as shown in Figure 2. *AdMiT-ZeroShot* denotes the accuracy achieved by directly applying the integrated module on target distributions without further adaptation, achieving higher average accuracy than most single-source TTA methods and demonstrating AdMiT 's efficiency and effectiveness in leveraging multi-source knowledge. We further evaluate AdMiT on 15 target distributions of the ImageNet-C dataset, varying the target batch sizes to assess stability. Due to space constraints, we report the average accuracy across target distributions in Table 1. The results show that AdMiT is less sensitive to batch size variations compared to other TTA methods, providing stable performance across different batch sizes.

Module selection. In previous experiments, all source modules were used regardless of their relevance to the target. To investigate selective module integration, we conduct experiments on the CIFAR-100C dataset with a set of $N = 75$ pre-trained modules. For each target batch, AdMiT selects the top $M = k$ modules based on mixture weights. Results in Fig. 3 indicate that AdMiT effectively identifies and uses only the most relevant modules, achieving strong adaptation performance with fewer modules.

Forgetting of source knowledge. To evaluate the resistance of AdMiT to catastrophic forgetting in dynamic test distributions, we use the CIFAR-100C dataset with four source modules pretrained on *Snow*, *Frost*, *Fog*, and *Bright* distributions. AdMiT selects $M = 3$ modules for adaptation to each new target distribution, maintaining higher accuracy on the original source distributions after adaptation. Table 2 shows that AdMiT outperforms other methods, including multi-source approaches like π-tuning and SESoM, as well as anti-forgetting methods like BECoTTA and SAR. Methods like Model Soup and TENT, which adapt solely to the current target, show a faster rate of forgetting.

Computational efficiency. To demonstrate AdMiT's computational advantages, we compare the overhead of our module matching approach against the additional costs incurred by hyperparameter tuning and routing network training. The results show that AdMiT achieves efficient source-target matching with lower computational cost and without any need for forward inference or retraining during deployment. Additionally, we evaluate the compatibility of AdMiT with various PET methods to confirm its minimal overhead and adaptability across different configurations. See the Appendix for the results.

5. Conclusion

We present AdMiT, a novel framework for transformers that dynamically integrates multiple source parameter-efficient tuning (PET) modules to address diverse and evolving target distributions. Unlike traditional PET methods that focus on single-source adaptation or require extra computation for multi-source integration, AdMiT selects and integrates relevant modules without the need for hyperparameter tuning, routing networks, or raw data sharing. This makes it well-suited for settings with privacy or resource constraints. AdMiT performs well in both zero-shot and dynamic adaptation scenarios, using a multi-source approach to handle distribution shifts while keeping source knowledge. Its theoretical guarantees ensure reliable adaptation, and its efficiency makes it practical for use in tasks with changing distributions.

Acknowledgments

This work was partially supported by NSF grants CCF-2046816, CCF-2403075, CCF-2008020, NSF CNS grant 210698, and Office of Naval Research grants N000142412289 and N000141812252. Additionally, research was sponsored by the OUSD (R&E)/RT&L and was accomplished under Cooperative Agreement Number W911NF-20-2-0267. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR, ARL and OUSD(R&E)/RT&L or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. This work was also partially supported by gifts from Open Philanthropy, Amazon Research, and Google Research.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. 1, 3

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020. 1

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[5] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020. 1

[6] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021. 1

[7] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021. 2

[8] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT understands, too," *AI Open*, 2023.

[9] T. Vu, B. Lester, N. Constant, R. Al-Rfou, and D. Cer, "Spot: Better frozen model adaptation through soft prompt transfer," *arXiv preprint arXiv:2110.07904*, 2021. 1

[10] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *International Conference on Machine Learning*, pp. 2790–2799, PMLR, 2019. 1, 3

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021. 1, 3

[12] X. Peng, C. Xing, P. K. Choubey, C.-S. Wu, and C. Xiong, "Model ensemble instead of prompt fusion: a sample-specific knowledge transfer method for few-shot prompt tuning," *arXiv preprint arXiv:2210.12587*, 2022. 1, 2, 3, 5, 6, 7

[13] E. L. Buehler and M. J. Buehler, "X-LoRA: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design," *APL Machine Learning*, vol. 2, no. 2, 2024. 1

[14] X. Wu, S. Huang, and F. Wei, "Mixture of LoRA experts," in *The Twelfth International Conference on Learning Representations*, 2024. 1, 2, 3

[15] S. M. Ahmed, F. F. Niloy, X. Chang, D. S. Raychaudhuri, S. Oymak, and A. K. Roy-Chowdhury, "CONTRAST: Continual multi-source adaptation to dynamic distributions," in *Advances in neural information processing systems*, 2024. 1, 3, 4, 6, 7, 5

[16] C. Wu, T. Wang, Y. Ge, Z. Lu, R. Zhou, Y. Shan, and P. Luo, "pi-tuning: Transferring multimodal foundation models with optimal multi-task interpolation," in *International Conference on Machine Learning*, pp. 37713–37727, PMLR, 2023. 2, 3, 4, 5, 6, 7

[17] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A kernel method for the two-sample-problem," *Advances in neural information processing systems*, vol. 19, 2006. 2

[18] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012. 2, 8, 9

[19] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016. 3, 4

[20] C. Sakaridis, D. Dai, and L. Van Gool, "ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10765–10775, 2021. 3, 4

[21] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim, "Visual prompt tuning," in *European Conference on Computer Vision*, pp. 709–727, Springer, 2022. 3

[22] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, no. 9, pp. 2337–2348, 2022. 3

[23] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Conditional prompt learning for vision-language models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16816–16825, 2022. 3

[24] P. Gao, S. Geng, R. Zhang, T. Ma, R. Fang, Y. Zhang, H. Li, and Y. Qiao, "Clip-adapter: Better vision-language models with feature adapters," *International Journal of Computer Vision*, vol. 132, no. 2, pp. 581–595, 2024. 3

[25] R. Zhang, R. Fang, W. Zhang, P. Gao, K. Li, J. Dai, Y. Qiao, and H. Li, "Tip-adapter: Training-free clip-adapter for better vision-language modeling," *arXiv preprint arXiv:2111.03930*, 2021. 3

[26] J. M. J. Valanarasu, P. Guo, V. VS, and V. M. Patel, "On-the-fly test-time adaptation for medical image segmentation," *arXiv preprint arXiv:2203.05574*, 2022. 3

[27] I. Shin, Y.-H. Tsai, B. Zhuang, S. Schulter, B. Liu, S. Garg, I. S. Kweon, and K.-J. Yoon, "MM-TTA: multi-modal test-time adaptation for 3d semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16928–16937, 2022.

[28] M. Hu, T. Song, Y. Gu, X. Luo, J. Chen, Y. Chen, Y. Zhang, and S. Zhang, "Fully test-time adaptation for image segmentation," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24*, pp. 251–260, Springer, 2021. 3

[29] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," *arXiv preprint arXiv:1603.04779*, 2016. 3

[30] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," *arXiv preprint arXiv:2006.10726*, 2020. 3, 6, 7, 4

[31] M. J. Mirza, J. Micorek, H. Possegger, and H. Bischof, "The norm must go on: Dynamic unsupervised domain adaptation by normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14765–14775, 2022. 3

[32] Q. Wang, O. Fink, L. Van Gool, and D. Dai, "Continual test-time domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022. 3

[33] D. Lee, J. Yoon, and S. J. Hwang, "BECoTTA: Input-dependent online blending of experts for continual test-time adaptation," *arXiv preprint arXiv:2402.08712*, 2024. 3, 6, 7, 5

[34] S. Niu, J. Wu, Y. Zhang, Y. Chen, S. Zheng, P. Zhao, and M. Tan, "Efficient test-time model adaptation without forgetting," in *International conference on machine learning*, pp. 16888–16905, PMLR, 2022. 3, 6

[35] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990. 3

[36] S. M. Ahmed, D. S. Raychaudhuri, S. Paul, S. Oymak, and A. K. Roy-Chowdhury, "Unsupervised multi-source domain adaptation without access to source data," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10103–10112, 2021. 3, 4, 5, 7

[37] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," *arXiv preprint arXiv:2010.01412*, 2020. 4, 6, 1

[38] S. Niu, J. Wu, Y. Zhang, Z. Wen, Y. Chen, P. Zhao, and M. Tan, "Towards stable test-time adaptation in dynamic wild world," *arXiv preprint arXiv:2302.12400*, 2023. 4, 6, 7, 1, 5

[39] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017. 4

[40] J. Mercer, "Functions of positive and negativetypeand their connection with theory ofintegral equations," *Philosophical Trinsoctions of Royal Society*, pp. 4–415, 1909. 4, 6

[41] H. Daumé III, "From Zero to Reproducing Kernel Hilbert Spaces in twelve pages or less," *Online: http://pub. hal3. name/daume04rkhs. ps*, 2004.

[42] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A hilbert space embedding for distributions," in *International conference on algorithmic learning theory*, pp. 13–31, Springer, 2007.

[43] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 4

[44] M. Balog, I. Tolstikhin, and B. Schölkopf, "Differentially private database release via kernel mean embeddings," in *International Conference on Machine Learning*, pp. 414–422, PMLR, 2018. 5, 2

[45] B. Neyshabur, H. Sedghi, and C. Zhang, "What is being transferred in transfer learning?," *Advances in neural information processing systems*, vol. 33, pp. 512–523, 2020. 5

[46] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, *et al.*, "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time," in *International conference on machine learning*, pp. 23965–23998, PMLR, 2022. 6, 7, 5

[47] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019. 6, 8

[48] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019. 6

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009. 6

[50] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," *Scientific Research Publishing*, 2009. 6

[51] A. Baevski and M. Auli, "Adaptive input representations for neural language modeling," *arXiv preprint arXiv:1809.10853*, 2018. 3

[52] C. Sakaridis, D. Dai, S. Hecker, and L. Van Gool, "Model adaptation with synthetic and real data for semantic dense foggy scene understanding," in *Proceedings of the european conference on computer vision (ECCV)*, pp. 687–704, 2018. 4

[53] X. Hu, C.-W. Fu, L. Zhu, and P.-A. Heng, "Depth-attentional features for single-image rain removal," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 8022–8031, 2019. 4

[54] J. Gu, H. Kwon, D. Wang, W. Ye, M. Li, Y.-H. Chen, L. Lai, V. Chandra, and D. Z. Pan, "Multi-scale high-resolution vision transformer for semantic segmentation," in *Proceedings of*

the IEEE/CVF conference on computer vision and pattern recognition, pp. 12094–12103, 2022. 4

[55] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020. 4

[56] C. McDiarmid *et al.*, “On the method of bounded differences,” *Surveys in combinatorics*, vol. 141, no. 1, pp. 148–188, 1989. 8

[57] J. Wellner *et al.*, *Weak convergence and empirical processes: with applications to statistics*. Springer Science & Business Media, 2013. 8

[58] D. Lopez-Paz, K. Muandet, B. Schölkopf, and I. Tolstikhin, “Towards a learning theory of cause-effect inference,” in *International Conference on Machine Learning*, pp. 1452–1461, PMLR, 2015. 9

Appendix Contents

A. Sharpness-aware Optimization

In order to properly finetune the new module $\theta(t)$ with the pseudo-label entropy minimization, we seek to make the model insensitive to large gradients by encouraging the model to converge to a flat area of the entropy loss surface, since a flat minimum leads to good generalization and robustness to large gradients [37, 38]:

$$\min_{\lambda} \mathcal{L}^{SA(t)}(\{x_i^{(t)}\}_{i=1}^{B};\lambda), \tag{A}$$

$$\text{where } \mathcal{L}^{SA(t)} \triangleq \max_{\|\epsilon\|_2\leq\rho} \mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B};\lambda+\epsilon) \tag{B}$$

in which $\mathcal{L}^{(t)}$ is defined in (5) in the main paper. In this context, the inner optimization aims to discover a perturbation ϵ of the module parameter $\theta(t)$ within a Euclidean ball of radius ρ that maximizes entropy. The degree of sharpness is measured by the maximum change in the Euclidean ball neighbourhood $N_\rho(\lambda)$. This bi-level problem incentivizes the optimization process to locate flat minima. Following SAM [37], we can approximately solve the inner optimization via a first-order Taylor expansion:

$$\begin{aligned}\epsilon^*(\lambda) &\triangleq \arg\max_{\|\epsilon\|_2\leq\rho} \mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B};\lambda+\epsilon)\\ &\approx \arg\max_{\|\epsilon\|_2\leq\rho} \mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B};\lambda)+\epsilon^\intercal\nabla_\lambda\mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B};\lambda)\\ &= \arg\max_{\|\epsilon\|_2\leq\rho} \epsilon^\intercal\nabla_\lambda\mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B};\lambda)\end{aligned}$$

Let $\mathbf{v} = \nabla_\lambda\mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B};\lambda)$. Hölder's inequality implies that $\epsilon^\top\mathbf{v} \leq \|\epsilon\|_p\|\mathbf{v}\|_q \leq \rho\|\mathbf{v}\|_q$ $(1/p+1/q=1)$. For $p=q=2$, the linear function achieves that bound $\epsilon^*(\lambda)^\intercal\mathbf{v} = \rho\|\mathbf{v}\|_2$, where $\epsilon^*(\lambda) = \rho\cdot\mathrm{sgn}(\mathbf{v})\cdot\frac{|\mathbf{v}|}{\|\mathbf{v}\|_2}$.
By substituting $\epsilon^*(\lambda)$ back into Eqn. 7 and differentiating both sides, the final gradient approximation is:

$$\nabla_\lambda\mathcal{L}^{SA(t)} \approx \nabla_\lambda\mathcal{L}^{(t)}(\{x_i^{(t)}\}_{i=1}^{B};\lambda)|_{\lambda+\epsilon^*(\lambda)}. \tag{C}$$

B. Synthetic data generation

The procedure is outlined in Alg. A. In the first two steps, synthetic data points $z_1,\ldots,z_M$ are selected independently of the private dataset, relying solely on the database size N. In the main paper, we use $q\sim\mathcal{N}(0,1)$. Steps 3 and 4 involve constructing the linear subspace $\mathcal{H}_M$ of $\mathcal{H}$ spanned by the feature maps of these synthetic points and computing a finite basis for this subspace. The private data is then accessed to calculate the empirical KME $\hat{\mu}_X$ (step 5), which is subsequently projected onto the subspace $\mathcal{H}_M$ and represented using the precomputed basis (step 6-7). The algorithm ensures that the number of synthetic data points M increases to infinity as N approaches infinity (step 1), guaranteeing that Algorithm 1 yields a consistent estimator of the true KME

Algorithm A Synthetic Data Subspace of the RKHS

```
Require: Dataset $\mathcal{D} = \{x_1, \dots, x_N\} \subset \mathcal{X}$, kernel $k$ on $\mathcal{X}$, number of synthetic data points $M$
Ensure: Weighted synthetic dataset (representing an estimate of $\mu_X$ in the RKHS $\mathcal{H}$ of $k$)
1: Initialize $z_1, \dots, z_M$ deterministically or randomly from some distribution $q$ on $\mathcal{X}$
2: $\mathcal{H}_M \leftarrow \mathrm{Span}(\{k(z_1, \cdot), \dots, k(z_M, \cdot)\}) \subseteq \mathcal{H}$
3: $b_1, \dots, b_F$ form an orthonormal basis of $\mathcal{H}_M$ (obtained using Gram-Schmidt process)
4: $\hat{\mu}_X \leftarrow \frac{1}{N}\sum_{n=1}^{N} k(x_n, \cdot)$, empirical KME of $X$ in $\mathcal{H}$
5: $\bar{\mu}_X \leftarrow \sum_{f=1}^{F} \langle b_f, \hat{\mu}_X \rangle b_f = \sum_{f=1}^{F} \alpha_f b_f$, projection of $\hat{\mu}_X$ onto $\mathcal{H}_M$
6: Re-express $\bar{\mu}_X \leftarrow \sum_{f=1}^{F} \beta_f b_f = \sum_{m=1}^{M} w_m k(z_m, \cdot)$ in terms of $k(z_m, \cdot)$
7: return $(z_1, w_1), \dots, (z_M, w_M)$
```

μ_X, provided that the synthetic data points are sampled from a distribution with sufficiently large support.

Lemma B.1. *([44], Lemma 10) Let $\mathcal{X}$ be a compact metric space and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a continuous kernel on $\mathcal{X}$. Suppose that the synthetic data points $z_1, z_2, \dots$ are sampled i.i.d. from a probability distribution q on $\mathcal{X}$. If the support $\mathrm{supp}(\mathcal{X})$ of $\mathcal{X}$ is included in the support of q, then*

$$\|\bar{\mu}_X - \hat{\mu}_X\|_{\mathcal{H}} \xrightarrow{p} 0 \text{ as } N \to \infty. \tag{D}$$

Proof Sketch. Let $\epsilon > 0$. Since k is continuous on the compact space $\mathcal{X} \times \mathcal{X}$, it is uniformly continuous. Therefore, there exists $\delta > 0$ such that

$$|k(x, x') - k(y, y')| < \epsilon^2 \quad \text{whenever} \quad \|x - y\| + \|x' - y'\| < \delta.$$

The compactness of $\mathcal{X}$ implies it is totally bounded; thus, $\mathrm{supp}(\mathcal{X})$ can be covered by finitely many balls $B_1, \dots, B_K$ of radius $\delta/2$. Since $\mathrm{supp}(\mathcal{X}) \subseteq \mathrm{supp}(q)$ and each $q(B_k) > 0$, with high probability, each B_k contains at least one sample point z_m as $M \to \infty$.

For each x_n, select $z_{m(n)}$ within distance δ (possible due to the coverage). Then,

$$\|\hat{\mu}_X - \bar{\mu}_X\|_{\mathcal{H}} \leq \frac{1}{N} \sum_{n=1}^{N} \left\|k(x_n, \cdot) - k(z_{m(n)}, \cdot)\right\|_{\mathcal{H}}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left(k(x_n, x_n) - 2k(x_n, z_{m(n)}) + k(z_{m(n)}, z_{m(n)})\right)^{1/2}$$

$$< \epsilon.$$

The last inequality follows from the uniform continuity of k and the choice of δ. Therefore, as $N \to \infty$, we have $\|\hat{\mu}_X - \bar{\mu}_X\|_{\mathcal{H}} \xrightarrow{p} 0$. □

C. Efficiency of AdMiT

C.1. Parameter Efficiency

Table A. **Number of parameters** for different models' scales and their corresponding PET module size. ViT-T/S/B/L stands for "Tiny, Small, Base, Large", corresponding to different pretrained ViT sizes. The bolded number is the size of the PET modules we applied in our main paper experiments.

	ViT-T	ViT-S	ViT-B	ViT-L
Full model	5,543,716	21,704,164	85,875,556	303,404,132
Adapter	58,564	116,932	233,668	417,984
LoRA	93,028	185,956	**371,812**	888,932
VPT	37,732	75,364	150,628	299,108
Header	19,300	38,500	76,900	102,500

Table B. **Computational cost.** We evaluate the computational cost and adaptation accuracy of multi-source adaptation for the baseline methods compared to AdMiT on ViT-B/16, using a batch size of 128 and LoRA PET modules (as detailed in the main paper), under both static (S) and dynamic (D) adaptation settings.

Tuning Method (PET module: LoRA)		Avg. Tuned Params During Adaptation (M)	GFLOPs (Avg.)	Image Classification Avg. Acc.(BS=128) ImageNet-C (S)	Digits (S)	CIFAR-100 (D)
Full	-	85.87	17.58	61.9	84.1	45.3
Multi	π-tuning	0.25	18.24	62.4	85.8	64.5
	SESoM	1.16	18.32	62.6	86.0	60.5
	CONTRAST	0.25	19.21	63.1	86.1	72.5
	Model soup	0.23	22.14	53.4	69.7	57.2
	AdMiT	0.23	17.62	63.7	86.8	73.3
	AdMiT-ZeroShot	0	17.62	59.6	82.2	59.1

Table C. **Latency from KME matching**. We evaluate the adaptation speed on ViT-B/16 backbone for Static adaptation on ImageNet-C among different PET methods, to demonstrate the latency brought by KME matching. The inference speed is defined by images per second (imgs/sec). All results are the average of 5 runs.

Method	GFLOPs (Avg across BS)	Adaptation speed (img/sec)			Slowdown Percentage (%)		
		BS=1	BS=32	BS=128	BS=1	BS=32	BS=128
Full fine-tuning	17.58	123.4	305.3	308.2	-	-	-
LoRA	17.58	95.7	291.3	283.1	-	-	-
LoRA-AdMiT	17.62	94.5	289.5	282.6	1.26	0.62	0.18
Adapter	17.81	104.2	285.6	296.3	-	-	-
Adapter-AdMiT	17.85	102.5	285.1	295.9	1.63	0.18	0.13
VPT	18.32	117.3	248.3	251.4	-	-	-
VPT-AdMiT	18.38	116.6	247.5	250.7	0.60	0.32	0.28

Parameter-efficient tuning (PET) methods align naturally with model ensemble techniques[7], particularly in terms of parameter efficiency. In contrast to other models where an ensemble of N models results in N times more module parameters, the additional module parameters introduced by the module integration in Alg. 1 is only of the size of one PET module. This represents less than 0.4% of a pretrained ViT-base model (≈ 86M, w.r.t. Table. A).

C.2. Computational Efficiency

In Table B, Table C, and Table D, we present a comprehensive comparison of inference speeds and adaptation performance across various benchmarks. As shown in Table D, we

Table D. **Edge device performance evaluation:** Adaptation performance and inference speed comparison on *Raspberry Pi 5* using Digit-5 dataset ($N = 4$) and ImageNet-C ($N = 14$), with the same experiment setting as the Figure 2 of the main paper. Existing PET methods exhibit slower adaptation speed than full fine-tuning despite tuning fewer parameters, as they require extra gradient propagation through the PET modules. Different source-target matching approaches further introduce varying computational overhead.

Tuning method	Num of	Adaptation Speed	Avg. Acc. (BS=128)	
PET module: LoRA	Tuned Prams (M)	(img/100 second)	Digit-5	ImageNet-C
Full model	85.87	42	83.5	61.4
π-tuning	0.25	34	83.2	62.1
SESoM	1.16	25	84.1	62.3
CONTRAST	0.25	37	84.4	62.8
AdMiT	0.23	40	**85.3**	**63.5**
AdMiT-ZeroShot	**0**	**145**	81.5	58.2

evaluate the computational efficiency of different methods on edge devices (Raspberry Pi 5) using Digit-5 ($N = 4$) and ImageNet-C ($N = 14$) datasets.

The results reveal that existing PET methods exhibit slower adaptation speeds than full fine-tuning despite tuning fewer parameters, as they require extra gradient propagation through the PET modules. Our method, AdMiT, demonstrates (Table C) superior computational efficiency during deployment with only a minimal slowdown (less than 2%) compared to standard PET methods due to the empirical KME calculation for module matching.

Notably, AdMiT achieves the highest accuracy on both benchmarks (85.3% on Digit-5 and 63.5% on ImageNet-C) while maintaining competitive adaptation speed (40 img/100 second). For scenarios where fine-tuning is not feasible, AdMiT-ZeroShot enables remarkably efficient adaptation through direct weighted combination of stored modules, achieving 81.5% and 58.2% accuracy on Digit-5 and ImageNet-C respectively, while offering the fastest inference speed (145 img/100 second) with zero tunable parameters.

Unlike existing methods that require extra tuning parameters or routing network optimization, AdMiT's matching and weight assignment rely solely on empirical KME calculations, making our method more computationally efficient during deployment while achieving superior adaptation performance.

D. Additional results

As shown in table. A, we use LoRA in the main paper as PET modules. We provide the results for other PET modules in this sections.

D.1. Details on different PET methods

Visual-Prompt Tuning (VPT). With a pre-trained Transformer (ViT) model as our starting point, we introduce a set of p continuous embeddings in the input space, each of dimension d, referred to as "prompts." During fine-tuning, only the prompts specific to the task are updated, while the

Table E. **Results on Digits**, same setup as for Fig 2.We train the source modules using 4 digits datasets to perform adaptation on the remaining dataset. All the results are the average of 5 runs. **Best** performance is bolded, and second-best performance is underlined. ($M = N = 4$).

PET Module	Source	Method	MM	MT	UP	SV	SY	Avg
LoRA	Single	GT-Tuning	**70.4**	**99.6**	**93.1**	**90.5**	**97.2**	**90.2**
	Multi	π-tuning-PL	61.3	98.2	89.1	86.2	94.6	85.9
		SESoM-PL	62.1	98.3	88.6	87.1	94.5	86.1
		Model soup	52.3	83.2	71.5	66.4	75.2	69.7
		AdMiT	63.2	98.9	89.3	86.5	96.2	86.8
Adapter	Single	GT-Tuning	**71.3**	**99.8**	**94.1**	**89.9**	**97.0**	**90.4**
	Multi	π-tuning-PL	62.0	97.5	88.8	86.0	94.5	85.8
		SESoM-PL	61.2	97.5	89.4	85.3	94.7	85.6
		Model soup	53.3	83.2	70.0	65.1	74.9	69.3
		AdMiT	63.9	97.8	88.9	85.0	94.4	86.0
VPT	Single	GT-Tuning	**71.6**	**99.7**	**92.5**	**92.6**	**97.9**	**90.9**
	Multi	π-tuning-PL	63.5	98.8	88.6	85.4	96.1	86.5
		SESoM-PL	64.2	98.3	88.3	88.7	93.8	86.7
		Model soup	51.8	82.5	71.9	66.4	75.4	69.6
		AdMiT	65.7	99.4	89.9	88.5	97.2	88.1

Transformer backbone remains frozen. We applied VPT-Shallow[21] as follows:

VPT-Shallow. Prompts are inserted into the first Transformer layer only. Each prompt token is a learnable $d-$dimensional vector. A module, which is a collection of p (which is the prompt length) prompts is denoted as $\mathbf{P} = \{\mathbf{v} \in \mathbb{R}^d | k \in \mathbf{N}, 1 \le k \le p\}$, the shallow-prompted ViT is:

$$[x_1, \mathbf{Z}_1, \mathbf{E}_1] = L_1([x_0, \mathbf{P}, \mathbf{E}_0]) \tag{E}$$

$$[x_i, \mathbf{Z}_i, \mathbf{E}_i] = L_i([x_{i-1}, \mathbf{Z}_{i-1}, \mathbf{E}_{i-1}]), i = 2, \cdots, l \tag{F}$$

$$\mathbf{y} = \text{Head}(x_l), \tag{G}$$

where $\mathbf{Z}_i \in \mathbb{R}^{p \times d}$ represents the features computed by the i-th Transformer layer, and $[x_i, \mathbf{Z}_i, \mathbf{E}_i] \in \mathbb{R}^{(1+p+P) \times d}$ (P is the number of patches that a 2D image input is divided into). The colors above indicate learnable and frozen parameters. For ViTs, x_i is invariant to the location of prompts since they are inserted after positional encoding. The overall parameter count for Adapters in an l-layer Transformer can be calculated as $|\theta| = p \times d$.

Adapter. In the conventional configuration, a transformer model incorporates two Adapters per layer [51]. Each Adapter layer is composed of $2 \times k \times d$ parameters, accounting for both the down and up-projection matrices. Here, k represents the input dimension size, while d refers to the bottleneck dimension of the Adapter. The overall parameter count for Adapters in an l-layer Transformer can be calculated as $|\theta| = 2 \times l \times 2 \times k \times d$.

Results for different PET modules. Following the setup in main paper, we replace LoRA with Adapters,VPTs as the PET module, and demonstrate the stationary distribution adaptation results in Table. E. In can be concluded that our method, AdMiT, retains good performance across different PET modules.

E. Semantic Segmentation

AdMiT is not limited to image classification tasks and can be seamlessly extended to tasks like semantic segmentation. In this setting, we assume access to a collection of pre-trained PET modules $\{\theta^j\}_{j=1}^N$, where each module is fine-tuned on a distinct source distribution for pixel-wise classification. Specifically, each module outputs per-pixel probabilities for K classes, formulated as $f_{\theta^j} : \mathbb{R}^{H\times W} \rightarrow \mathbb{R}^{H\times W\times K}$. To adapt AdMiT for semantic segmentation, the entropy term in Eqn. 5 of the main paper is updated as follows:

$$\mathcal{L}_w^{(t)}(\mathbf{w}) = -\mathbf{E}_{\mathcal{D}_T^{(t)}} \sum_{h=1}^{H} \sum_{w=1}^{W} \sum_{c=1}^{K} \hat{y}_{\text{hwc}}^{(t)} \log(\hat{y}_{\text{hwc}}^{(t)}) \qquad \text{(H)}$$

Here, $\hat{y}_{\text{hwc}}^{(t)}$ represents the weighted probability output corresponding to class c for the pixel at location (h, w) at time-step t. The rest of the framework remains unchanged, ensuring consistency across tasks.

E.1. Datasets

Our experiments involve the following datasets:

- **Cityscapes:** The Cityscapes dataset [19] provides large-scale, densely annotated pixel-level data for 30 classes, grouped into 8 categories: flat surfaces, humans, vehicles, constructions, objects, nature, sky, and void. Simulated variants of this dataset include fog and rain conditions [52, 53].
- **ACDC:** The Adverse Conditions Dataset (ACDC) [20] includes pixel-level annotations for images captured under adverse conditions such as fog, nighttime, rain, and snow. The class structure aligns with the 19 semantic labels used in the Cityscapes evaluation, excluding the void class.

E.2. Experimental Setup

For all experiments, we use HRViT-b1 [54] as the segmentation model. We evaluate performance on 19 semantic labels, excluding the void label.

We consider a static target distribution setting for evaluation. Specifically, we train three source PET modules using the **clean**, **fog**, and **rain** splits of the Cityscapes dataset. After training, these modules are tested on the respective weather condition splits of the ACDC dataset. Using AdMiT, we dynamically integrate the source modules for each target condition and compare the results with baseline methods.

E.3. Results

Results on Cityscapes to ACDC: Table F shows the performance of AdMiT and baseline methods on ACDC test data under different weather conditions (static target distributions). The source modules are trained on Cityscapes and its simulated noisy variants. AdMiT significantly outperforms baseline adaptation methods, with results reported in terms of % mIoU, highlighting its effectiveness in leveraging multi-source knowledge for target adaptation.

Table F. **Semantic segmentation results.**

Source	Method	Fog	Rain	Snow	Night	Avg
Single	TENT-Best	25.4	21.7	19.7	13.5	20.0
	BECoTTA-Best	26.3	22.4	21.3	14.5	21.1
	SAR-Best	25.8	22.2	20.1	15.5	20.9
Multi	π-tuning-PL	28.3	23.0	24.2	17.4	23.2
	SESoM-PL	29.2	25.3	25.4	18.2	24.5
	CONTRAST	32.4	29.4	25.2	18.7	26.4
	Model soup	25.4	25.5	21.4	14.7	21.8
	AdMiT	**32.5**	**29.9**	**25.4**	**19.1**	**26.7**
	AdMiT-ZeroShot	27.5	22.3	19.9	14.7	21.1

F. Implementation details

We perform all the experiment on a single A100 GPU. We use ViT-Base-16 [55] model in all our experiments. For all experiments without extra clarification, we use a target batch size of $|T| = 128$, as used by TENT [30]. The experimental setup for tuning the integrated module is listed as in Table. G summarizing the optimization configurations we used. Implementation details for each tuning method apply to both source and target distributions.

In this problem setting, we propose to adaptively combine multiple pre-trained parameter-efficient tuning (PET) modules during deployment through suitable combination weights, which are determined based on a limited number of target samples. Consider the scenario where we have a collection of N pre-trained PET modules, denoted as $\{\theta^j\}_{j=1}^N$, which are fine-tuned on distinct source distributions. During deployment, target data arrives in an online fashion as a sequence of batches $\{x_i^{(1)}\}_{i=1}^B \rightarrow \{x_i^{(2)}\}_{i=1}^B \rightarrow \ldots \{x_i^{(t)}\}_{i=1}^B \rightarrow \ldots$, where t represents the time-stamp and B is the number of samples in each target batch. The target distribution at time-stamp t is denoted as $\mathcal{D}_T^{(t)}$, implying $\{x_i^{(t)}\}_{i=1}^B \sim \mathcal{D}_T^{(t)}$.

Motivated by the multi-source adaptation framework, we model the target distribution at each time-stamp t as a linear combination of source distributions, with combination weights denoted as $\{w_j^{(t)}\}_{j=1}^N$. Using these weights, AdMiT integrates the pre-trained PET modules to form an adaptive module for the current target batch. Thus, the inference model for test batch t can be expressed as $f_T^{(t)} = f_{\theta(t)}$, where $\theta(t) = \sum_{j=1}^N w_j^{(t)} \theta^j$ is the dynamically integrated PET module for time-stamp t.

We implement the baselines as follows:

- **TENT.** TENT [30] adapts transformers by modifying only the LayerNorm statistics during test-time adaptation while keeping the PET modules and backbone weights unchanged. It minimizes the entropy of predictions for target batches, encouraging confident predictions. The LayerNorm parameters (mean and variance) are updated

Table G. **Hyperparameters** for tuning AdMiT

	Full, Adapter, LoRA	VPT
Optimizer	AdamW	SGD
Optimizer momentum	N/A	0.9
$base\_lr$ search range	{0.001, 0.0001, 0.0005, 0.005}	{50., 25., 10., 5., 2.5, 1.,0.5, 0.25, 0.1, 0.05}
Weight decay range	{0.01, 0.001, 0.0001, 0.0}	
LR schedule	cosine decay	
Warm up epochs	10	
Total epochs	100	

iteratively using gradients computed from the entropy loss. Key hyperparameters include the learning rate for updating LayerNorm statistics (1×10^{-4}) and the batch size for target adaptation ($B = 64$).

- **BECoTTA.** BECoTTA(-M) [33] integrates multiple PET modules using a MoDE (Mixture of Domain Experts) module, which applies a Top-K routing strategy to select the $K = 2$ most relevant PET modules based on input features. During pretraining, BECoTTA initializes with $D = 3$ proxy domains (source domain, darkness, and brightness) and trains the MoDE module alongside a domain discriminator and a synergy loss, freezing the backbone. In deployment, PET modules are activated for online adaptation, with entropy-based filtering used to refine pseudo-labels. The primary hyperparameters include the number of proxy domains ($D = 3$) and the Top-K selection parameter ($K = 2$).
- **SAR.** SAR [38] adapts transformers by restricting updates to LayerNorm statistics during test-time adaptation while freezing PET modules and backbone weights. Unlike TENT, SAR selectively filters high-entropy (low-confidence) samples, focusing on reliable predictions. Sharpness-aware optimization is applied to smooth the entropy loss, improving robustness against noisy target distributions. The key hyperparameters include the entropy threshold for filtering ($E_0 = 0.4 \times \ln(K)$, where K is the number of classes) and the sharpness radius ($\rho = 0.05$) for optimization.
- **π-Tuning.** π-Tuning [16] combines knowledge from multiple pre-trained PET modules by interpolating their outputs based on task similarity. Task embeddings are computed using the Fisher Information Matrix (FIM), with similarity calculated as cosine similarity between the embeddings of target and source tasks. The top k PET modules are selected for interpolation, and weights are optimized via pseudo-label entropy minimization. Key hyperparameters include the number of selected PET modules ($k = 3$), learning rate for fine-tuning (1×10^{-4}).
- **SESoM.** SESoM [12] integrates the outputs of multiple source models through the utilization of an additional attention-based routing network. Within our experimental framework, each PET module operates as a source-specific unit. The logits derived from these modules are transmitted to an attention-based routing network tasked with computing sample-specific weights. The routing network undergoes fine-tuning via pseudo-label entropy minimization, employing few-shot pseudo-labeled target data while maintaining the PET modules and backbone architecture in a fixed state. Key hyperparameters comprise the learning rate for the attention module (3×10^{-4}) and a dropout rate set at 0.1. An attentionn-based routing network of approximately $d \times d'_x + d'_x \times d' + v \times d'_l + d'_l \times d' + 4d' = 0.85M$ size (as defined in [12]) is employed to derive the attention weights.
- **CONTRAST.** CONTRAST [15] adapts to evolving target distributions by dynamically combining pre-trained source models and selectively updating the most relevant model. For ViT-based architectures, CONTRAST computes weights for PET modules based on LayerNorm statistics or feature embeddings and updates the PET module with the highest weight for each target batch. Key hyperparameters include learning rate for weight updates (1×10^{-4}), and test batch size ($B = 128$).
- **Model Soup.** Model Soup [46] improves performance by averaging the weights of multiple fine-tuned PET modules without additional inference cost. We adapt Model Soup by sequentially adding PET modules to the soup using the Greedy Soup strategy, retaining only modules that improve validation accuracy. Hyperparameters include learning rates ($\{10^{-4}, 10^{-5}\}$) and using 10% of training data for validation.

G. Details of dataset corruptions

We summarize these corruptions types by example in Fig. A. The order of these corruptions is the same as the order in Table. 2 and Figure. 3.

H. Theoretical Insights

H.1. Maximum Mean Discrepancy as distribution similarity metric

The objective of module selection in AdMiT is to quantify the similarity between the source distribution $\mathcal{D}_S$ and

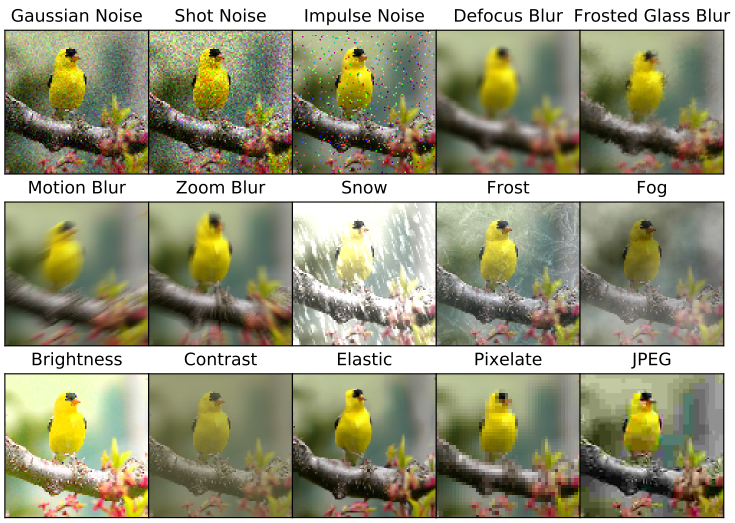


Figure A. Examples of each corruption type in the image corruptions benchmark. While synthetic, this set of corruptions aims to represent natural factors of variation like noise, blur, weather, and digital imaging effects.

the target distribution $\mathcal{D}_{\mathcal{T}}$ without access to the raw data. Rather than comparing moments of different orders, such as $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}}(\boldsymbol{x}^n)$, we adopt a more comprehensive metric, $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}}(f(\boldsymbol{x}))$, to universally characterize the properties of distributions.

The discrepancy between the expected function values across two distributions, $\mathcal{D}_{\mathcal{T}}$ and $\mathcal{D}_{\mathcal{S}}$, is captured by the Maximum Mean Discrepancy (MMD). Mathematically, MMD is defined in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ as:

$$\mathrm{MMD}(\mathcal{F},\mathcal{D}_{\mathcal{T}},\mathcal{D}_{\mathcal{S}}) = \sup_{f\in\mathcal{F}} \left(\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_{\mathcal{S}}}(f(\boldsymbol{x})) - \mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_{\mathcal{T}}}(f(\boldsymbol{x}))\right).$$

Without loss of generality, f is assumed to reside within a unit ball , i.e., $||f||_{\mathcal{H}} \le 1$. The RKHS is structured using an orthogonal basis derived from the decomposition of a symmetric and positive semi-definite kernel function $k(\boldsymbol{x},\boldsymbol{y})$. In the main paper, we employ a Gaussian radial basis kernel $k(\boldsymbol{x},\boldsymbol{y}) = \exp(-||\boldsymbol{x}-\boldsymbol{y}||^2)$.

A symmetric and positive semi-definite kernel function $k(\boldsymbol{x},\boldsymbol{y})$ can be decomposed [40] into a set of eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$ and corresponding orthogonal eigenfunctions $\{\psi_i(\cdot)\}_{i=1}^{\infty}$:

$$k(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{\infty} \lambda_i\psi_i(\boldsymbol{x})\psi_i(\boldsymbol{y}).$$

These eigenfunctions form an orthogonal basis $\{\sqrt{\lambda_i}\psi_i(\cdot)\}$ used to construct the Hilbert space $\mathcal{H}$. Any function f within this space can be expressed either as a linear combination of these basis functions:

$$f(\cdot) = \sum_{i=1}^{\infty} f_i\sqrt{\lambda_i}\psi_i(\cdot),$$

or represented as an infinite-dimensional vector in $\mathcal{H}$: $f = (f_1, f_2, \dots)_{\mathcal{H}}^{\top}$. When one parameter of the kernel function is fixed to $\boldsymbol{x}$, it behaves like a function with a single variable or an infinite vector:

$$k(\boldsymbol{x},\cdot) = \sum_{i=1}^{\infty} \lambda_i\psi_i(\boldsymbol{x})\psi_i(\cdot) = (\sqrt{\lambda_1}\psi_1(\boldsymbol{x}), \sqrt{\lambda_2}\psi_2(\boldsymbol{x}), \dots)_{\mathcal{H}}^{\top}.$$

This leads to the following computation for the inner product of these two functions, illustrating the reproducing property of RKHS:

$$\begin{aligned}\langle f, k(\boldsymbol{x},\cdot)\rangle_{\mathcal{H}} &= (f_1, f_2, \dots)_{\mathcal{H}} \cdot (\sqrt{\lambda_1}\psi_1(\boldsymbol{x}), \sqrt{\lambda_2}\psi_2(\boldsymbol{x}), \dots)_{\mathcal{H}}^{\top} \\ &= \sum_{i=1}^{\infty} f_i\sqrt{\lambda_i}\psi_i(\boldsymbol{x}) = f(\boldsymbol{x}),\end{aligned}$$

which effectively captures the essence of the reproducing property within the RKHS framework.

Furthermore, for a given distribution $\mathcal{D}$, we introduce the kernel mean embedding (KME), defined as:

$$\mu_{\mathcal{D}} = \mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}}[k(\boldsymbol{x},\cdot)].$$

This allows the expressions within the Maximum Mean Discrepancy (MMD) to be rewritten in terms of inner products in the RKHS:

$$\begin{aligned}\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}}(f(\boldsymbol{x})) &= \mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}}\langle f, k(\boldsymbol{x},\cdot)\rangle_{\mathcal{H}} = \langle f, \mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}}k(\boldsymbol{x},\cdot)\rangle_{\mathcal{H}} \\ &= \langle f, \mu_{\mathcal{D}}\rangle_{\mathcal{H}}.\end{aligned}$$

In the context of MMD, we assess the supremum with these inner products:

$$\mathrm{MMD}(\mathcal{F}, \mathcal{D}_\mathcal{T}, \mathcal{D}_\mathcal{S}) = \sup_{||f||_\mathcal{H} \leq 1} (\mathbf{E}_{\boldsymbol{x} \sim \mathcal{D}_\mathcal{S}}(f(\boldsymbol{x})) - \mathbf{E}_{\boldsymbol{x} \sim \mathcal{D}_\mathcal{T}}(f(\boldsymbol{x})))$$
$$= \sup_{||f||_\mathcal{H} \leq 1} \langle \mu_{\mathcal{D}_\mathcal{T}}, f \rangle_\mathcal{H} - \langle \mu_{\mathcal{D}_\mathcal{S}}, f \rangle_\mathcal{H}$$
$$= \sup_{||f||_\mathcal{H} \leq 1} \langle \mu_{\mathcal{D}_\mathcal{T}} - \mu_{\mathcal{D}_\mathcal{S}}, f \rangle_\mathcal{H} \leq \sup_{||f||_\mathcal{H} \leq 1} ||\mu_{\mathcal{D}_\mathcal{T}} - \mu_{\mathcal{D}_\mathcal{S}}||_\mathcal{H} \cdot ||f||_\mathcal{H}$$
$$= ||\mu_{\mathcal{D}_\mathcal{T}} - \mu_{\mathcal{D}_\mathcal{S}}||_\mathcal{H}.$$

We work with several source datasets $S_j = \{(x_i, y_i)\}_{i=1}^{|S_j|} \sim \mathcal{D}_{\mathcal{S}j}$, $j \in [N]$ in the pretraining stage, and an unlabeled target batch $T = \{(x_i, \cdot)\}_{i=1}^{|T|} \sim \mathcal{D}_\mathcal{T}$ in the deployment stage. The empirical KME at these phases can be estimated as:

$$\widehat{\mu(T)} = \frac{1}{|T|} \sum_{x_n \in T} k(x_n, \cdot), \quad \widehat{\mu(S_j)} = \frac{1}{|S_j|} \sum_{x_n \in S_j} k(x_n, \cdot). \tag{I}$$

We can also compute the squared Maximum Mean Discrepancy (MMD^2) by expanding the definition:

$$\mathrm{MMD}^2(\mathcal{D}_\mathcal{T}, \mathcal{D}_{\mathcal{S}j}) = ||\mu_{\mathcal{D}_\mathcal{T}} - \mu_{\mathcal{D}_{\mathcal{S}j}}||_\mathcal{H}^2$$
$$= ||\mu_{\mathcal{D}_\mathcal{T}}||_\mathcal{H}^2 - 2\langle \mu_{\mathcal{D}_\mathcal{T}}, \mu_{\mathcal{D}_{\mathcal{S}j}} \rangle_\mathcal{H} + ||\mu_{\mathcal{D}_{\mathcal{S}j}}||_\mathcal{H}^2$$
$$= \mathbf{E}_{\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{D}_\mathcal{T}} k(\boldsymbol{x}, \boldsymbol{y}) - 2\mathbf{E}_{\boldsymbol{x} \sim \mathcal{D}_\mathcal{T}, \boldsymbol{y} \sim \mathcal{D}_{\mathcal{S}j}} k(\boldsymbol{x}, \boldsymbol{y})$$
$$+ \mathbf{E}_{\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{D}_{\mathcal{S}j}} k(\boldsymbol{x}, \boldsymbol{y}),$$

which results in the following empirical estimate:

$$\widehat{\mathrm{MMD}}^2(T, S_j)$$
$$= \frac{1}{|T|^2} \sum_{x_n, x_m \in T} k(x_n, x_m) - \frac{2}{|T||S_j|} \sum_{x_n \in T, x_m \in S_j} k(x_n, x_m)$$
$$+ \frac{1}{|S_j|^2} \sum_{x_n, x_m \in S_j} k(x_n, x_m). \tag{J}$$

We provide proofs for the following properties:
- If the kernel $k(\cdot, \cdot)$ is universal, the mapping from $\mathcal{D}$ to $\mu(\mathcal{D})$ via KME is injective.
- $\mathrm{MMD}(\mathcal{D}_\mathcal{T}, \mathcal{D}_\mathcal{S}) = 0$ if and only if $\mathcal{D}_\mathcal{T} = \mathcal{D}_\mathcal{S}$.

These properties enable the measurement of distribution similarity through MMD and KME.

Theorem H.1. *If the kernel k is universal, then the mean map $\mu : \mathcal{P}_X \to \mathcal{H}$ is injective.*

Proof. We will use a proof by contradiction to establish the theorem.

Assume that $\mu : \mathcal{P}_X \to \mathcal{H}$ is not injective. Then, there exist two different probability measures p and q such that $\mu[p] = \mu[q]$, i.e.,

$$\mu[p] = \mu[q].$$

The mean map $\mu[p]$ is represented as:

$$\mu[p](\cdot) = \int_X k(x, \cdot)\, dp(x),$$

and similarly,

$$\mu[q](\cdot) = \int_X k(x, \cdot)\, dq(x).$$

For any $f \in \mathcal{H}$, we have:

$$\langle f, \mu[p] \rangle_\mathcal{H} = \langle f, \mu[q] \rangle_\mathcal{H}.$$

The inner product $\langle f, \mu[p] \rangle_\mathcal{H}$ can be written as:

$$\langle f, \mu[p] \rangle_\mathcal{H} = \mathbb{E}_{X \sim p}[f(X)],$$

and similarly,

$$\langle f, \mu[q] \rangle_\mathcal{H} = \mathbb{E}_{X \sim q}[f(X)].$$

Since $\mu[p] = \mu[q]$, it follows that:

$$\mathbb{E}_{X \sim p}[f(X)] = \mathbb{E}_{X \sim q}[f(X)] \quad \forall f \in \mathcal{H}.$$

By the universality of the kernel k, the RKHS $\mathcal{H}$ is dense in $C(X)$. This means that the above equality holds for all continuous functions $f \in C(X)$.

By the uniqueness theorem for measures, if two measures p and q agree on all continuous functions, then $p = q$. This contradicts our assumption that p and q are different.

Therefore, the assumption that μ is not injective must be false, and hence μ is injective. □

Theorem H.2. *Let $\mathcal{F}$ be a unit ball in a universal RKHS $\mathcal{H}$, defined on the compact metric space $\mathcal{X}$, with associated continuous kernel $k(\cdot, \cdot)$. Then MMD $\{\mathcal{F}, p, q\} = 0$ if and only if $p = q$.*

Proof. First, it is clear that $p = q$ implies MMD $\{\mathcal{F}, p, q\}$ is zero. We now prove the converse.

By the universality of $\mathcal{H}$, for any given $\epsilon > 0$ and $f \in C(\mathcal{X})$, there exists a $g \in \mathcal{H}$ such that

$$\|f - g\|_\infty \leq \epsilon.$$

We next make the expansion

$$|\mathbb{E}_x f(x) - \mathbb{E}_y f(y)| \leq |\mathbb{E}_x f(x) - \mathbb{E}_x g(x)| +$$
$$|\mathbb{E}_x g(x) - \mathbb{E}_y g(y)| + |\mathbb{E}_y g(y) - \mathbb{E}_y f(y)|.$$

The first and third terms satisfy

$$|\mathbb{E}_x f(x) - \mathbb{E}_x g(x)| \leq \mathbb{E}_x |f(x) - g(x)| \leq \epsilon.$$

Next, write

$$\mathbb{E}_x g(x) - \mathbb{E}_y g(y) = \langle g, \mu_p - \mu_q \rangle_\mathcal{H} = 0,$$

since MMD $\{\mathcal{F}, p, q\} = 0$ implies $\mu_p = \mu_q$. Hence

$$|\mathbb{E}_x f(x) - \mathbb{E}_y f(y)| \leq 2\epsilon$$

for all $f \in C(\mathcal{X})$ and $\epsilon > 0$, which implies $p = q$. □

H.2. w-convergence bound

For a bounded kernel $0 \leq k(\cdot,\cdot) \leq K$, the biased empirical estimator of MMD is

$$\mathrm{MMD}_b[\mathcal{F}, X, Y] = \sup_{f\in\mathcal{F}} \left(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(y_i) \right),$$

where X, Y are random variables from distribution p, q, and $\{x_i\}_{i=1}^m$, $\{y_i\}_{i=1}^n$ are samples drawn from these two distributions.

We want to show that the absolute difference between $\mathrm{MMD}(\mathcal{F}, p, q)$ and $\mathrm{MMD}_b(\mathcal{F}, X, Y)$ is close to its expected value, independent of the distributions p and q. To this end, we prove three intermediate results, which we then combine. The first result we need is an upper bound on the absolute difference between $\mathrm{MMD}(\mathcal{F}, p, q)$ and $\mathrm{MMD}_b(\mathcal{F}, X, Y)$. We have

$$\begin{aligned}
&|\mathrm{MMD}(\mathcal{F}, p, q) - \mathrm{MMD}_b(\mathcal{F}, X, Y)| \\
&= |\sup_{f\in\mathcal{F}} \left(\mathbb{E}_x(f) - \mathbb{E}_y(f)\right) \\
&- \sup_{f\in\mathcal{F}} \left(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(y_i) \right)| \\
&\leq \sup_{f\in\mathcal{F}} \underbrace{\left| \mathbb{E}_x(f) - \mathbb{E}_y(f) - \frac{1}{m}\sum_{i=1}^{m} f(x_i) + \frac{1}{n}\sum_{i=1}^{n} f(y_i) \right|}_{\Delta(p,q,X,Y)}
\end{aligned} \tag{K}$$

Then, we provide an upper bound on the difference between $\Delta(p, q, X, Y)$ and its expectation. Changing either of x_i or y_i in $\Delta(p, q, X, Y)$ results in changes in magnitude of at most $2K^{1/2}/m$ or $2K^{1/2}/n$, respectively. We can then apply McDiarmid's theorem [56], given a denominator in the exponent of

$$\begin{aligned}
&m\left(\frac{2K^{1/2}}{m}\right)^2 + n\left(\frac{2K^{1/2}}{n}\right)^2 \\
&= 4K\left(\frac{1}{m} + \frac{1}{n}\right) = \frac{4K(m+n)}{mn},
\end{aligned}$$

to obtain

$$\Pr_{X,Y}\left(\Delta(p, q, X, Y) - \mathbb{E}_{X,Y}[\Delta(p, q, X, Y)] > \epsilon\right) \leq \exp\left(-\frac{\epsilon^2 mn}{2K(m+n)}\right). \tag{L}$$

Next, we exploit symmetrisation, following [57], to upper bound the expectation of $\Delta(p, q, X, Y)$. Denoting by X' an i.i.d sample of size m drawn independently of X (and likewise for Y'), we have

$$\begin{aligned}
&\mathbb{E}_{X,Y}[\Delta(p, q, X, Y)] \leq \\
&\mathbb{E}_{X,Y} \sup_{f\in\mathcal{F}} \left| \mathbb{E}_x(f) - \frac{1}{m}\sum_{i=1}^{m} f(x_i) - \mathbb{E}_y(f) + \frac{1}{n}\sum_{i=1}^{n} f(y_i) \right| \\
&= \mathbb{E}_{X,Y} \sup_{f\in\mathcal{F}} |\mathbb{E}_{X'}\left(\frac{1}{m}\sum_{i=1}^{m} f(x_i')\right) - \frac{1}{m}\sum_{i=1}^{m} f(x_i) \\
&- \mathbb{E}_y(f) + \frac{1}{n}\sum_{i=1}^{n} f(y_i)| \\
&= \mathbb{E}_{X,Y} \sup_{f\in\mathcal{F}} \left| \mathbb{E}_{X'}\left(\frac{1}{m}\sum_{i=1}^{m} f(x_i') - \frac{1}{m}\sum_{i=1}^{m} f(x_i)\right) \right. \\
&\left. + \frac{1}{n}\sum_{i=1}^{n} f(y_i) - \mathbb{E}_y(f) \right| \\
&\leq \quad \mathbb{E}_{X,Y,X',Y'} \sup_{f\in\mathcal{F}} \left| \frac{1}{m}\sum_{i=1}^{m} f(x_i') - \frac{1}{m}\sum_{i=1}^{m} f(x_i) \right| \\
&+ \mathbb{E}_{Y,Y'} \sup_{f\in\mathcal{F}} \left| \frac{1}{n}\sum_{i=1}^{n} f(y_i') - \mathbb{E}_y(f) \right| \\
&= \quad \mathbb{E}_{X,Y,X',Y'} \sup_{f\in\mathcal{F}} \left| \frac{1}{m}\sum_{i=1}^{m} \sigma_i\left(f(x_i') - f(x_i)\right) \right| + \\
&\mathbb{E}_{Y,Y',\sigma'} \sup_{f\in\mathcal{F}} \left| \frac{1}{n}\sum_{i=1}^{n} \sigma_i'\left(f(y_i') - f(y_i)\right) \right| \\
&\leq \quad \mathbb{E}_{X,X',\sigma} \sup_{f\in\mathcal{F}} \left| \frac{1}{m}\sum_{i=1}^{m} \sigma_i\left(f(x_i') - f(x_i)\right) \right| + \\
&\mathbb{E}_{Y,Y',\sigma'} \sup_{f\in\mathcal{F}} \left| \frac{1}{n}\sum_{i=1}^{n} \sigma_i'\left(f(y_i') - f(y_i)\right) \right| \\
&\leq \quad 2\left[R_m(\mathcal{F}, p) + R_n(\mathcal{F}, q)\right] \\
&\leq 2\left[\left(\frac{K}{m}\right)^{1/2} + \left(\frac{K}{n}\right)^{1/2}\right]
\end{aligned} \tag{M}$$

The first step applies Jensen's inequality to simplify the expression. Next, the triangle inequality is used to separate the terms, followed by substituting the Rademacher average to connect the empirical and expected values. Finally, the Rademacher averages are bounded as defined in Definition 30 of [18]. Combining Eqn. L with the bounded term

$\mathbb{E}_{X,Y}[\Delta(p,q,X,Y)]$ derived from Eqn. M, we can obtain

$$\Pr_{X,Y}\left(\Delta(p,q,X,Y) - 2\left[\left(\frac{K}{m}\right)^{1/2} + \left(\frac{K}{n}\right)^{1/2}\right] > \epsilon\right) \leq \exp\left(-\frac{\epsilon^2 mn}{2K(m+n)}\right). \tag{N}$$

Combining Eqn. N with Theorem 7 in [18], leads to the following Corollary:

Corollary H.3. *If $p = q$, then with probability at least $1-\delta$, the (biased) empirical MMD (obtained by drawing m samples from p and n samples from q) is bounded by:*

$$\frac{1}{2}MMD_b(\hat{p},\hat{q}) < \sqrt{\frac{K}{m}} + \sqrt{\frac{K}{n}} + \sqrt{\frac{K(m+n)\log\frac{1}{\delta}}{2mn}}$$

for any arbitrarily small $\delta > 0$.

Setting $p = \mathcal{D}_\mathcal{T}, q = \sum_{j=1}^N w_j \mathcal{D}_{\mathcal{S}j}, \sum_{j=1}^N w_j = 1$ leads to:

$$\boxed{\frac{1}{2}\left\|\widehat{\mu(T)} - \sum_{j=1}^N w_j \widehat{\mu(S_j)}\right\|_\mathcal{H} < \sqrt{\frac{K}{m}} + \sqrt{\frac{K}{n}} + \sqrt{\frac{K(m+n)\log\frac{1}{\delta}}{2mn}}.}$$

The above result upper-bounds the error of estimating empirical target KME with weighted average of empirical source KME in the optimization in 4 of the main paper. Note that n is the number of samples from q, i.e. the number of samples from the **involved** source datasets, thus as the number of modules N increases, n also increases.

Similarly, we can derive the following estimation error bound for KME and empirical KME:

Corollary H.4. *([58], Theorem 1) With probability at least $1-\delta$ we have*

$$\|\mu(p) - \widehat{\mu(p)}\|_\mathcal{H} \leq 2\sqrt{\frac{K}{n}} + \sqrt{\frac{2\log\frac{1}{\delta}}{n}}.$$

Proof. Proof can be found in [58], section B.1. ☐

H.3. Loss bound

H.3.1. Assumptions

Suppose we are given a pretrained frozen backbone transformer model f, and there are N source domains in the upload phase. We build PET modules on their own domains and upload them to the module store for future users. Each source domain has a corresponding dataset $\{S_j\}_{j=1}^N$, reflecting the distribution $\mathcal{D}_{\mathcal{S}j}$. The source datasets will be inaccessible after the pretraining stage.

We also assume that a global optimal rule function $g : \mathcal{X} \to \mathcal{Y}$ exists for the domain adaptation problem,

$$\forall j \in [N], \forall(x,y) \in S_j, g(x) = y.$$

We assume that all sources are competent, and the source datasets are sufficient to solve their domains. Formally speaking, the modules $\hat{\theta}_j$ can help the pretrained model to reach a small error rate $\epsilon > 0$ with respect to a certain loss function L (upper-bounded by $|L|$) on their domain distribution $\mathcal{D}_{\mathcal{S}j}$ when applied with the pretrained backbone model f:

$$\forall j \in [N], \mathcal{L}(\mathcal{D}_{\mathcal{S}j}, f_{\hat{\theta}_j}) = \mathbb{E}_{x\sim\mathcal{D}_{\mathcal{S}j}}[L(f_{\hat{\theta}_j}(x), y)] \leq \epsilon. \tag{O}$$

In this context, the loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ can be either a regression loss or classification loss. Since the domains $\{S_j\}_{j=1}^N$ are equipped with low-error pre-trained modules, they are referred to as solved domains.

In the deployment phase, a new user wants the model to solve the current domain with only unlabeled testing data $x \sim \mathcal{D}_\mathcal{T}$. Thus the target is to learn a good model $\hat{f}_t$ which minimizes $\mathcal{L}(\mathcal{D}_\mathcal{T}, \hat{f}_t)$, utilizing the information contained in pre-trained modules $\{\hat{\theta}_j\}_{j=1}^N$.

Theorem H.5 (Zero-shot adaptation loss bound)**.** *Assume that the assumptions in corollary H.3 hold. The module θ_j from each source dataset satisfies $\forall j \in [N], \mathcal{L}(\mathcal{D}_{\mathcal{S}j}, f_{\hat{\theta}_j}) = \mathbb{E}_{x\sim\mathcal{D}_{\mathcal{S}j}}[L(f_{\hat{\theta}_j}(x), y)] \leq \epsilon$. Assume that the loss function $L(f_{\hat{\theta}_j}(x), f(x)) \in \mathcal{H}_k$. The empirical MMD between distribution mixture $\sum_{j=1}^N w_j \mathcal{D}_{\mathcal{S}j}$ and current distribution $\mathcal{D}_\mathcal{T}$ can be estimated from*

$$MMD_b = \left\|\widehat{\mu(T)} - \sum_{j=1}^N w_j \widehat{\mu(S_j)}\right\|_\mathcal{H}$$

Then the finite sample loss satisfies:

$$\mathcal{L}(\mathcal{D}_\mathcal{T}, g, f_{\sum \boldsymbol{w}_j\hat{\theta}_j}) = \sum_{x_i\sim\mathcal{D}_\mathcal{T}} \left[L(f_{\sum \boldsymbol{w}_j\hat{\theta}_j}(x_i), g(x_i))\right] \leq \epsilon + O(\sqrt{\frac{1}{m}} + \sqrt{\frac{1}{n}})$$

Proof. By the reproducing property, the loss function can be written as:

$$L(f_{\sum \boldsymbol{w}_j\hat{\theta}_j}, g(x)) = L_{\sum \boldsymbol{w}_j\hat{\theta}_j}(x) = \langle L_{\sum \boldsymbol{w}_j\hat{\theta}_j}, k(x,\cdot)\rangle.$$

We then can represent the error of each model in the form of KME. For the current mixture:

$$\mathbb{E}_{x\sim\mathcal{D}_\mathcal{T}}[L_{\sum \boldsymbol{w}_j\hat{\theta}_j}(x)] = \langle L_{\sum \boldsymbol{w}_j\hat{\theta}_j}, \mu_{\mathcal{D}_\mathcal{T}}\rangle$$

The empirical loss can also be represented by:

$$\left|\left\langle L_{\sum w_j\hat{\theta}_j}, \widehat{\mu(T)}\right\rangle\right| \leq$$
$$\underbrace{\left|\left\langle L_{\sum w_j\hat{\theta}_j}, \widehat{\mu(T)}\right\rangle - \left\langle L_{\sum w_j\hat{\theta}_j}, \sum_j w_j\widehat{\mu(S_j)}\right\rangle\right|}_{(A)} +$$
$$\underbrace{\left|\left\langle L_{\sum w_j\hat{\theta}_j}, \sum_j w_j\widehat{\mu(S_j)}\right\rangle\right|}_{(B)} \tag{P}$$

We then bound (A) and (B) separately.

By the convergence rate for empirical MMD,

$$\begin{aligned}(A) &= \left|\langle L_{\sum \boldsymbol{w}_j\hat{\theta}_j}, \widehat{\mu(T)} - \sum_{j=1}^{N} w_j\widehat{\mu(S_j)}\rangle\right| \\ &\leq \|L_{\sum \boldsymbol{w}_j\hat{\theta}_j}\|\|\widehat{\mu(T)} - \sum_{j=1}^{N} w_j\widehat{\mu(S_j)}\| \\ &\leq O(m,n) \end{aligned}\tag{Q}$$

$$\begin{aligned}(B) \leq& \left|\langle L_{\sum \boldsymbol{w}_j\hat{\theta}_j}, \sum w_j\mu_{\mathcal{D}_{\mathcal{S}j}}\rangle\right| \\ &+ \left|\langle L_{\sum \boldsymbol{w}_j\hat{\theta}_j}, \sum_j w_j\widehat{\mu(S_j)} - \sum w_j\mu_{\mathcal{D}_{\mathcal{S}j}}\rangle\right| \\ &\leq \sum_{j=1}^{N} w_j\left|\langle L_{\hat{\theta}_j}, \mu_{\mathcal{D}_{\mathcal{S}j}}\rangle\right| \\ &+ |L|\,\|\sum_j w_j\widehat{\mu(S_j)} - \sum w_j\mu_{\mathcal{D}_{\mathcal{S}j}}\| \\ &\leq \epsilon + 2\sqrt{\frac{K}{m}} + \sqrt{\frac{2\log\frac{1}{\delta}}{m}} \end{aligned}\tag{R}$$

The last steps of the estimation (A) and (B) can be obtained directly from corollary H.4 and corollary H.3, and this completes the proof.

□

Theorem H.5 estimates the loss in directly applying the integrated module $\theta(t)$ to the target domain without any tuning on the target dataset, i.e. the zero-shot performance. It ensures a good initialization of the tuning in Sec. 3.2, leading to a good adaptation performance.