Unsupervised Domain Adaption Harnessing Vision-Language Pre-training

Wenlve Zhou, Zhiheng Zhou *

***Abstract*—This paper addresses two vital challenges in Unsupervised Domain Adaptation (UDA) with a focus on harnessing the power of Vision-Language Pre-training (VLP) models. Firstly, UDA has primarily relied on ImageNet pre-trained models. However, the potential of VLP models in UDA remains largely unexplored. The rich representation of VLP models holds significant promise for enhancing UDA tasks. To address this, we propose a novel method called Cross-Modal Knowledge Distillation (CMKD), leveraging VLP models as teacher models to guide the learning process in the target domain, resulting in state-of-the-art performance. Secondly, current UDA paradigms involve training separate models for each task, leading to significant storage overhead and impractical model deployment as the number of transfer tasks grows. To overcome this challenge, we introduce Residual Sparse Training (RST) exploiting the benefits conferred by VLP's extensive pre-training, a technique that requires minimal adjustment (approximately 0.1%∼0.5%) of VLP model parameters to achieve performance comparable to fine-tuning. Combining CMKD and RST, we present a comprehensive solution that effectively leverages VLP models for UDA tasks while reducing storage overhead for model deployment. Furthermore, CMKD can serve as a baseline in conjunction with other methods like FixMatch, enhancing the performance of UDA. Our proposed method outperforms existing techniques on standard benchmarks. Our code will be available at: https://github.com/Wenlve-Zhou/VLP-UDA.**

***Index Terms*—Unsupervised domain adaptation, vision-language pre-training, cross-modal knowledge distillation, residual sparse training, model deployment.**

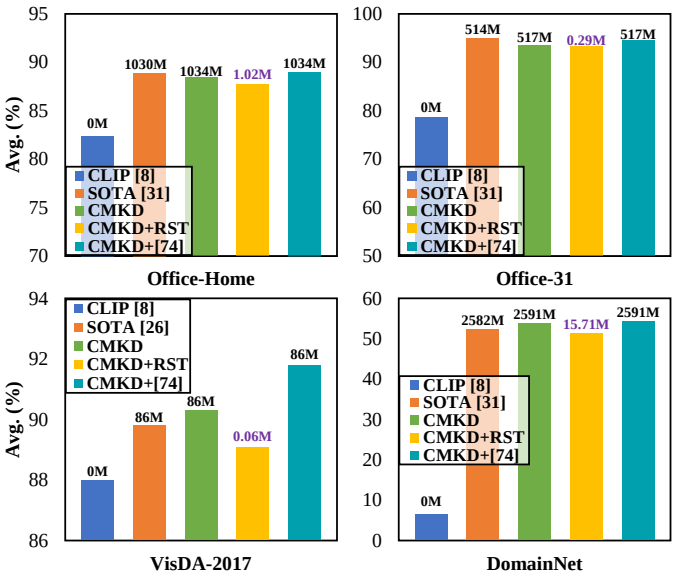


Fig. 1. The bar chart displays the average accuracy of our method and previous State-Of-The-Art (SoTA) on popular benchmarks. The statics on each bar represent the Downstream Parameters (DSP) of the respective method (The definition of DSP is detailed in Section IV). Combining CMKD with RST leads to a substantial reduction in model storage overhead, while its combination with FixMatch [74] results in further enhanced model performance. CLIP [8] represents the zero-shot inference performance on target domain.

I. Introduction

DEEP learning has been witnessed remarkable progress recently [1]–[3], yet it has not performed as well when applied to the actual target dataset as neural networks are sensitive to domain gaps. Fortunately, Unsupervised Domain Adaptation (UDA) has emerged as a critical research area that transferring knowledge from labeled source domains to unlabeled target domains, offering potential solutions to address the limitations of deep learning [4]–[7]. While considerable advancements have been made in UDA, two crucial challenges have received insufficient attention. Firstly, the considerable potential of VLP's rich feature representation for UDA tasks remains largely unexplored, such as CLIP [8], FLIP [9], etc. Secondly, current UDA methods predominantly rely on training a distinct full-parameter model for each target domain, which not only poses inconveniences during model deployment but also demands substantial storage capacity for the parameters.

UDA approaches capitalize on prior knowledge from pre-trained models to adapt to target domains. Previous studies have primarily employed models pre-trained on ImageNet-1k [10] or ImageNet-21k [11]. In the era of large-scale vision-language pre-training, VLP models have gained significant traction. The extensive training provides rich representations that offer tremendous potential for boosting UDA tasks. Surprisingly, despite extensive pre-training on vision-language datasets, VLP model still fall short of the state-of-the-art methods in terms of zero-shot performance on relevant benchmarks (see Figure 1). It suggests that the VLP model hasn't achieved

This work is supported by the National Key Research and Development Program of China(2022YFF0607001), Guangdong Basic and Applied Basic Research Foundation (2023A1515010993), Guangdong Provincial Key Laboratory of Human Digital Twin (2022B1212010004), Guangzhou City Science and Technology Research Projects (2023B01J0011), Jiangmen Science and Technology Research Projects (2021080200070009151), Shaoguan Science and Technology Research Project(230316116276286), Foshan Science and Technology Research Project(2220001018608). (* Corresponding author: Zhiheng Zhou.)

Wenlve Zhou, Zhiheng Zhou are with School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, Guangdong, China, and also with Key Laboratory of Big Data and Intelligent Robot, Ministry of Education, South China University of Technology, Guangzhou 510641, China (email: wenlvezhou@163.com; zhouzh@scut.edu.cn).

comprehensive coverage across various data distributions. Moreover, applying previous methods to VLP has the potential to distort semantic feature structures and compromise class discriminability [12]. Hence, effectively leveraging the pre-training features becomes a critical aspect. The two common tunning approaches for VLP models are "Visual Encoder Only" and "Prompt Tuning", as shown in Figure 2(a). The former involves excluding the text encoder and solely fine-tuning the visual encoder [13], which inadvertently sacrifices the rich general knowledge encapsulated in the text encoder—a pivotal aspect for acquiring domain-specific knowledge. Similar to how humans rely on common-sense to expedite learning in specialized subjects, models can benefit from this general knowledge. The latter approach, "Prompt Tuning," freezes the vision-language encoder and exclusively fine-tunes newly added learnable tokens via Image-Text Contrastive learning (ITC) [14]. Although this approach preserves the model's general knowledge and its efficiency has been validated in other fields [15], considering the limited model capacity in the UDA task, the representation of the frozen visual encoder may not necessarily cover the target domain in UDA. Additionally, this approach incurs additional computational costs during model inference due to the inclusion of the text encoder.

Another significant challenge manifests during the model deployment phase. As business requirements evolve, numerous diverse target domains may emerge. Traditional methods typically necessitate training a full-parameters model for each new target domain. In real-world deployment scenarios, preserving models for both source and target domains demands substantial parameter storage, resulting in inefficiencies. Therefore, it becomes crucial to explore strategies that facilitate model adaptation with minimal additional storage parameters for target domains. One promising avenue is the exploration of methods such as Parameter-Efficient Fine-Tuning (PEFT) [16] in Large Language Models (LLMs) [17], where a few extra parameters are trained in the training phase, and inference without additional burden via re-parameterization tricks [18]. The visual encoder in VLP models can act like LLMs, and the heavy storage and deployment problems can be solved by training a small number of additional parameters for each domain along the lines of PEFT, as shown in Figure 2(b). However, the effectiveness of such methods on relatively lightweight models remains uncertain, such as ResNet50 [19] and ViT-B [20].

In this paper, we address these challenges by proposing novel methods to integrate visual-language models into UDA tasks and explore efficient model deployment strategies. To address Challenge 1, we propose a simple but effective method called Cross-Modal Knowledge Distillation (CMKD). By leveraging general knowledge from the text encoder as prior, this method assists the model in self-training on the target domain data. The CMKD can be easily implemented with a simple loss function. For Challenge 2, we introduce a novel parameter-efficient training method called Residual Sparse Training (RST), which does not require modifying the network architecture. With RST, model deployment can be achieved using a highly sparse weights with almost no performance degradation. Both of these techniques can be applied to the CNNs or Transformers architectures within VLP models. We conduct extensive experiments and evaluations on various benchmarks to demonstrate the effectiveness and efficiency of our proposed approaches. Our contributions are summarized as follows:

(1) We incorporate VLP into UDA tasks and propose Cross-Modal Knowledge Distillation (CMKD) to efficiently utilize the prior knowledge from the text encoder.

(2) With the Residual Sparse Training (RST), we achieve the goal of domain adaptation, while significantly reducing the parameter storage and improving the deployment efficiency.

(3) In extensive experimental evaluations, our method has demonstrated exceptional performance on multiple benchmarks, while also exhibiting efficient computational and memory requirements. This validates the feasibility and practicality of our approach.

The subsequent sections of this paper are organized as follows: Section II presents a comprehensive literature review. In Section III, we introduce our proposed methodologies. Section IV provides detailed information regarding the experimental setup and presents the evaluation results. Finally, Section V concludes the paper and outlines potential directions for future research.

II. Related Work

In this section, we present a brief survey on the most related fields of our works in four aspects: visual-language pre-training, unsupervised domain adaption, knowledge distillation, and parameter efficient fine-tuning.

A. Visual-Language Pre-training

Visual-language pre-training have been witnessed significant advancements in recent research, with two prominent paradigms: single-tower models and dual-tower models. Single-tower models aim to jointly encode images and text using a shared visual-linguistic encoder, such as the Transformer architecture, mapping them into a common embedding space [21]–[23]. Through training on alignment tasks for image-text pairs, single-tower models learn semantic correlations between images and text. On the other hand, dual-tower models employ separate visual and language encoders to process image and text inputs independently [8], [9], [24]. The visual encoder transforms images into visual feature representations, while the language encoder generates semantic vectors from text. By training on alignment tasks, dual-tower models learn to establish correspondences between visual and linguistic modalities.

To address the feature entanglement issue, the dual-tower model is utilized in this paper, as it emphasizes the characterization of individual modes instead of solely focusing on feature interactions between them. This enables the text encoder's knowledge to serve as common knowledge, facilitating the training of downstream tasks.

B. Unsupervised Domain Adaption

Unsupervised domain adaptation has emerged as a prominent research area in computer vision, aiming to address the

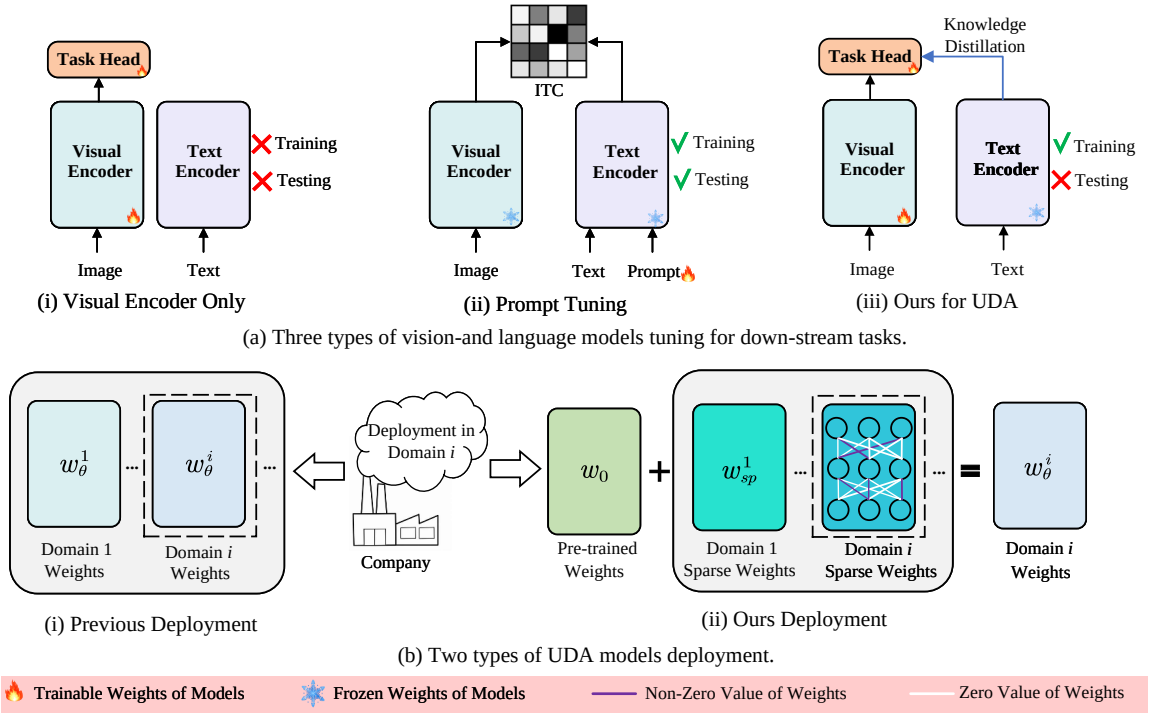


Fig. 2. Overview of VLP models tuning and UDA deployment. (a) VLP models tuning. The previous methods struggled to balance the general knowledge utilization and domain-specific representation. We introduce cross-modal knowledge distillation enabling reconciliation and remove text encoder when inference. (b) UDA deployment. Conventional UDA deployment selects domain weights with full parameters based on the application scenario. Our method learns a highly sparse weight (approximately 0.1%∼0.5% of the downstream models' parameters) that can be added to the pre-trained model for deployment.

domain shift problem. UDA methods seek to leverage labeled data from a source domain to improve the performance of a model on a target domain where labeled data is unavailable. Various approaches have been proposed to tackle UDA, including domain adversarial learning [25], [26], self-training [27], and consistency regularization [28]. These methods seek to align the feature distributions between the source and target domains, thereby facilitating knowledge transfer. In recent years, due to Transformers' [29] powerful contextual modeling capabilities, it has swept the major fields of computer vision, including unsupervised domain adaptation. These methods are based on Transformer's inductive bias, designing cross-domain attention [7], [30], patch-mix mechanism [31]. Thanks to the effectiveness of the Transformer framework, UDA tasks have seen significant performance improvements in recent years.

However, prior studies have seldom taken into account the utilization of pre-training on visual-language data, and although Ge et al. [12] introduced DAPL into UDA, its performance improvement is limited compared to the State-Of-The-Art (SoTA), and its application scenario may be constrained by the requirement for precise domain descriptions. Moreover, each of these methods necessitates "one-domain-one-model" paradigm, creating significant obstacles for the deployment of UDA.

C. Knowledge Distillation

Knowledge distillation has gained significant attention in machine learning as a powerful technique for transferring knowledge from a complex teacher model to a simpler student model. The primary objective is to enhance the performance of the student model by leveraging the knowledge encapsulated in the teacher model. By mimicking the output behaviour of the teacher model, the student model can benefit from the rich information provided by the teacher. Various approaches have been proposed to effectively carry out knowledge distillation. One common method is to minimize the discrepancy between the predictions of the student model and the soft targets generated by the teacher model. This can be achieved through techniques such as KL divergence, which encourages the student model to produce similar output probabilities as the teacher model [32]. Additionally, intermediate representations [33], structural knowledge [34] from the teacher model can be utilized as additional supervision signals to guide the learning process of the student model.

In this paper, drawing inspiration from knowledge distillation, the proposed CMKD effectively transfers common knowledge from the VLP models of text encoders to assist the model in unlabelled target data adaption.

D. Parameter Efficient Fine-tuning

Parameter efficient fine-tuning techniques aim to adapt a frozen pre-trained model to a specific target task or domain using minimal additional parameters. These methods [16], [35]–[41] are particularly valuable in resource-constrained environments or situations where storing multiple models is impractical. Approaches like prefix tuning [37] and prompt tuning [38], [40]–[42] explore strategies to excavate base model with a few learnable tokens. Adapter [39] adds new network layers or modules between network layers inside the pre-trained model to adapt to downstream tasks. More recently,

LoRA [16] has gained popularity for tuning large models due to its effectiveness and negligible additional inference overhead, which has achieved performance comparable to fine-tuning with the full parameters. Besides, for any downstream task, it is only necessary to save the relevant additional parameters instead of the entire model.

LoRA [16] offers a solution to the "one-domain-one-model" problem. However, its effectiveness significantly diminishes when transitioning from LLMs to models with less parameters in VLP models. To address this limitation, we propose RST, a technique that achieves superior performance while requiring less storage for additional parameters.

III. Methods

In the unsupervised domain adaption, data from two different distributions will be sampled to form the source labeled domain $D_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and unlabeled target domain $D_t = \{x_j^t\}_{j=1}^{n_t}$ datasets, where n_s and n_t denote the quantity of data in each domain. $y^s \in \mathbb{R}^{n_s \times c}$ and c represent the number of classes in the UDA datasets. The primary objective of UDA is to learn a domain-invariant network that can adapt to the target domain using source data and labels, even in the absence of labeled target domain data during training.

In Section III.A, We will first give an introduction to the dual-tower model CLIP [8] as preliminary. Based on the analysis of Section I and the inspiration from the knowledge distillation, we propose cross-modal knowledge distillation $\mathcal{L}_{\text{cmkd}}$ to assists the model in self-training on the target domain data, as described in Section III.B. Furthermore, in Section III.C, we introduce residual sparse training to improve the flexibility of deployment in response to the current heavy overhead of UDA deployment.

For UDA, the objective is formulated as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{cmkd}} \tag{1}$$

where $\mathcal{L}_{\text{cls}}$ is the standard classification loss of source domain implemented with Cross-Entropy.

A. Preliminary

Our backbone architecture is based on CLIP [8], consisting of an image encoder, denoted as $f(\cdot)$, and a text encoder, denoted as $\mathrm{g}(\cdot)$. The image encoder can be either ResNet [19] or Vision Transformer [20], while the text encoder adopts a Transformer [29]. These encoders transform the high-dimensional image and text inputs into a lower-dimensional feature space.

CLIP [8] is trained in a contrastive manner using image-text pairs. Each input text describes a category using the format "a photo of a [CLASS]" (where [CLASS] represents the class token). A positive pair consists of an image x_i and its corresponding text t_i, which describes the category of x_i. On the other hand, a negative pair pairs an image x_i with an irrelevant description t_j, where $i \neq j$. The training objective aims to maximize the cosine similarity of positive pairs and minimize the cosine similarity of negative pairs. This contrastive learning objective aligns the image and text representations in the same feature space.

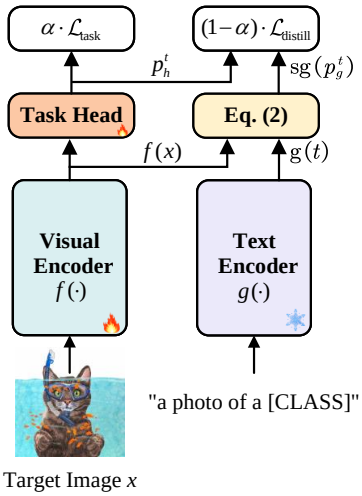


Fig. 3. The pipeline of cross-modal knowledge distillation. CMKD is exclusively utilized for target data.

By leveraging these aligned features, CLIP [8] enables zero-shot inference. Given K category descriptions, when forwarding an image x, the model predicts that it belongs to the category y_i can be described as:

$$p_{\text{g}}(y=i|x) = \frac{\exp(\langle \text{g}(t_i), f(x) \rangle)}{\sum_{k=1}^{K} \exp(\langle \text{g}(t_k), f(x) \rangle)} \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the cosine similarity.

Both a text encoder and a visual encoder are utilized to facilitate unsupervised domain adaptation. Throughout the training process, we introduce an auxiliary task head $h(\cdot)$ in conjunction with the visual encoder to establish a downstream model, while maintaining the text encoder in a frozen state. Due to the extensive pre-training of CLIP on a dataset consisting of 400 million image-text pairs [8], we posit that the p_{g} arises from the common-sense reasoning of model. This general knowledge bolsters the downstream model in adaption, equipping the model to execute UDA. Additionally, during the inference phase, the text encoder is omitted to alleviate the burden on the inference process.

B. Cross-Modal Knowledge Distillation

Based on the previous analysis, we anticipate that the knowledge distillation framework, incorporating the common knowledge of the CLIP [8] model, could facilitate adaptation for unlabeled target data. The classical knowledge distillation framework [32] employs KL divergence to minimize the disparity between the distributions of the student and teacher models, facilitating effective knowledge transfer, which can be formulated as:

$$\mathcal{L}_{\text{kd}} = \alpha \cdot \text{KL}(y || p_{\text{student}}) + (1-a) \cdot \text{KL}(sg(p_{\text{teacher}}) || p_{\text{student}}) \tag{3}$$

where y represents the ground truth, p_{student} denotes the output of the student model, p_{teacher} signifies the output of the teacher model, and $sg(\cdot)$ is the operation of stop-gradient. For convenience, the first term quantifies the discrepancy between the predictions of the student model and the ground-truth, referred to as the task term $\mathcal{L}_{\text{task}}$. The second term assesses

the divergence between the predictions of the student model and those of the teacher model, known as the distillation term $\mathcal{L}_{\text{distill}}$. The parameter a represents the trade-off for balancing these two tasks.

However, due to the absence of corresponding labels in the target domains, executing the task term becomes challenging. To achieve a model capable of approximating learning in task-term, we adopt self-training [43]. A common practice is to minimize the uncertainty of predictions on unlabeled data with Gibbs entropy. Yet the Gibbs entropy is verified that make the predictions over-confident [27], we use weaker penalization Gini impurity (GI) for self-training, where the objective and gradient function can be formulated as:

$$\mathcal{L}_{\text{task}} = \text{GI}(p_h^t) = 1 - \sum_{i=1}^{c} [p_h^t(y=i|x^t)]^2 \tag{4}$$

$$\frac{\partial \mathcal{L}_{\text{task}}}{\partial \theta} = -\sum_{i=1}^{c} [2 \cdot p_h^t(y=i|x^t) \cdot \frac{\partial p_h^t(y=i|x^t)}{\partial \theta}] \tag{5}$$

where $p_h^t = h(f(x^t))$ denotes the output from the task-head, and θ refer to the trainable components, namely the task-head and visual encoder parameters.

For the distillation term, a straightforward approach is to directly utilize $\mathcal{L}_{\text{distill}}$ in the vanilla KD [32], written as:

$$\mathcal{L}_{\text{distill}} = \text{KL}(sg(p_{\text{g}}^t)||p_h^t) \tag{6}$$

Yet Figure 1 reveals that CLIP's zero-shot performance lags behind that of SoTA. Optimizing Eq. (6) would punish the student perspective p_h^t closer to the teacher perspective p_{g}^t, potentially adversely affecting model performance instead. Hence, the primary role of p_{g}^t should be as a common-sense auxiliary for the expertise learning of the model instead of ground truth. Although the probability distribution of the task-head p_h^t may differ from p_{g}^t, p_h^t should remain dominant, while p_{g}^t serves only as a constraining factor. The distillation term can be rewritten as:

$$\mathcal{L}_{\text{distill}} = \text{GI}\left(p_m^t\right) = 1 - \sum_{i=1}^{c} \left[p_m^t\left(y=i \mid x^t\right)\right]^2 \tag{7}$$

$$p_m^t = 0.5 \cdot \left(p_h^t + sg\left(p_{\text{g}}^t\right)\right) \tag{8}$$

In the above objective, we employ self-training to train the mixed distribution of p_h^t and p_{g}^t, with p_{g}^t being held constant throughout the training process. To better understand the role, the gradient of the Eq. (7) is first given as:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{distill}}}{\partial \theta} = &-\sum_{i=1}^{c} p_h^t\left(y=i \mid x^t\right) \cdot \frac{\partial p_h^t\left(y=i \mid x^t\right)}{\partial \theta} \\ &-\sum_{i=1}^{c} p_{\text{g}}^t\left(y=i \mid x^t\right) \cdot \frac{\partial p_h^t\left(y=i \mid x^t\right)}{\partial \theta}\end{aligned} \tag{9}$$

We observe that p_{g}^t serves as a constraint on the gradient compared to Eq. (5). In cases where there is a significant disparity between p_h^t and p_{g}^t, the model tends to gravitate towards the mixed distribution rather than attempting to force the constraint to match p_{g}^t. The pipeline is shown in Figure 3 and the overall objectives of CMKD are:

$$\mathcal{L}_{\text{cmkd}} = \alpha \cdot \mathcal{L}_{\text{task}} + (1-a) \cdot \mathcal{L}_{\text{distill}} \tag{10}$$

After exploring the fundamental framework of CMKD, we observed that the hyper-parameter α in KD [32] is typically determined through experience to strike a balance between distillation loss and task loss. However, in unsupervised domain adaptation, where labels for knowledge transfer are unavailable, we suggest employing a dynamic mechanism to produce distinct trade-offs for each sample. We opted to use the discrepancy between p_h^t and p_{g}^t as the trade-off α, dynamically determining the weighting of the two tasks based on evaluating the consistency between the students' and the teachers' perspective, written as:

$$\alpha = sg(\exp(-\text{KL}(p_h^t \parallel p_{\text{g}}^t))) \tag{11}$$

where $\alpha \in \mathbb{R}^{n_t}$, range from 0 to 1.

Furthermore, since the visual encoder is trainable, the distribution of p_{g}^t will gradually drift. To ensure the stability of training, we incorporate a regularization term $\mathcal{L}_{\text{reg}}$ that constrains the parameter space based on the CLIP [8] fine-tuning paradigm. As a result, Eq. (10) is reformulated as follows:

$$\mathcal{L}_{\text{cmkd}} = \lambda_1 \cdot (\alpha \cdot \mathcal{L}_{\text{task}} + (1-a) \cdot \mathcal{L}_{\text{distill}}) + \mathcal{L}_{\text{reg}} \tag{12}$$

$$\mathcal{L}_{\text{reg}} = \lambda_2 \cdot \text{KL}\left(y||p_g^s\right) + \lambda_3 \cdot \text{GI}\left(p_{\text{g}}^t\right) \tag{13}$$

The trade-off parameters λ_1, λ_2 and λ_3 are introduced to balance the multi-loss

Algorithm 1 RST Training.

Input: task *i*, task weight w_θ^i, pre-trained weight w_0, pre-defined threshold τ
for t = 0 to MaxIter **do**
  Obtain w_θ^i with UDA training.
  With w_0 and τ, update w_θ^i by Eq. (17)
end for
With w_0 gain w_{sp}^i by Eq. (16)
Preserve w_{sp}^i in sparse form.
Return w_{sp}^i

Algorithm 2 RST Inference.

Input: task *i*, pre-trained weight w_0, sparse series $\{w_{\text{sp}}^k\}_{k=1}^n$
Retrieve w_{sp}^i from the series $\{w_{\text{sp}}^k\}_{k=1}^n$
Transform to dense form.
Obtain task weight with addition:

$$w_\theta^i = w_0 + w_{sp}^i$$

Return w_θ^i

C. Residual Sparse Training

During the adaptation of LLM to downstream tasks, LoRA [16] successfully achieves remarkable generalization with minimal parameter storage and negligible performance degradation. Due to the low intrinsic dimension of LLM [44], Hu et

al. [16] put forward the hypothesis that updates to the weights during adaptation also exhibit a low intrinsic rank and propose LoRA, formulated as:

$$h = w_0 \cdot x + \Delta w \cdot x = w_0 \cdot x + B \cdot A \cdot x \tag{14}$$

where $w_0 \in \mathbb{R}^{d\times k}$ is a pre-trained weight matrix. The update of the weight is constrained and update by representing the latter with a low-rank decomposition $w_0 + \Delta w = w_0 + B \cdot A$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d, k)$.

However, the assumption that low intrinsic dimension of updated weights may not be applicable to VLP models due to the relatively small size of the models, such as ResNet50 [19] and ViT-B [20]. The theory of structural neuroplasticity in brain neuroscience [45] provides valuable insight into the dynamic nature of neuronal connections. While neurons in the brain are predominantly hard-wired, localized regions have the capacity to remodel when learning or adapting to new environments. This phenomenon allows frequently used neurons to strengthen their connections, facilitating the brain's adaptation to changing circumstances. Inspired by this theory, we draw an analogy between the neural network fine-tuning process in downstream tasks and the brain's adaptation to new environments. During fine-tuning, weight connections exhibiting significant variations are deemed beneficial for learning in the downstream task, resembling the remodeling process of neuronal links in the brain. Conversely, weights with minor variations are considered irrelevant to the downstream task, akin to the hardwired regions of the brain that remain fixed as pre-training weights. This approach, termed Residual Sparse Training (RST), entails modifying only a minimal fraction of weights in the pre-trained model to fine-tune the parameters, mirroring the selective adaptation observed in the brain's structural neuroplasticity. Moreover, the index of the weights to be altered should be determined by the requirements of the UDA tasks. The RST is given as:

$$h = w_\theta \cdot x = w_0 \cdot x + w_{\text{sp}} \cdot x \tag{15}$$

where $w_\theta \in \mathbb{R}^{d\times k}$ is a task weight during UDA training. Upon completion of the training process, the sparse weights w_{sp} can be obtained using the residual paradigm, as expressed by the following equation:

$$w_{\text{sp}} = w_\theta - w_0 \tag{16}$$

Therefore, the focus shifts from learning a sparse weight w_{sp} to learning a weight that closely resembles w_0. We employ a straightforward method to determine whether to retain the pre-trained weights by comparing the magnitude of the weight change with a predefined threshold τ, written as:

$$w_\theta = \begin{cases} w_\theta, |w_0 - w_\theta| > \tau \\ w_0, |w_0 - w_\theta| \leq \tau \end{cases} \tag{17}$$

When considering n downstream tasks, after conducting RST, a sparse series $\{w_{\text{sp}}^k\}_{k=1}^n$ is generated. This series is then summed with the pre-training weights w_0 to derive the task-specific sequence $\{w_\theta^k\}_{k=1}^n$. The entire process of RST is remarkably simple, requiring no modification to the network structure. As a result, it can be seamlessly applied to both CNNs and Transformers. We illustrate RST in Figure 2(b) and propose a pseudo-code implementation of training and inference in Algorithm 1 and Algorithm 2 respectively.

IV. Experiments

We evaluate CMKD and RST against competitive UDA baseline on popular benchmarks. These datasets include **Office-Home**, **Office-31**, **Visda-2017**, **ImageCLEF-DA**, **DomainNet**. Apart from the datasets, digits classification constructed from **MNIST**, **USPS** and **SVHN** is taken into account. Besides, RST will be compared with the popular PEFT to explore the superiority of our method in VLP. In ablation studies, the various settings of our methods are investigated.

A. Setup

Office-Home [46] contains 15,588 images, which consists of images from 4 different domains: Artistic images (Ar), Clip Art (Cl), Product images (Pr) and Real-World images (Re). Collected in office and home settings, the dataset contains images of 65 object categories for each domain. We use all domain combinations and construct 12 transfer tasks.

Office-31 [47] is a benchmark dataset for domain adaptation which collected from three distinct domains: Amazon (A), Webcam (W), and DSLR (D). The dataset comprises 4,110 images in 31 classes, taken by web camera and digital SLR camera with different photographical settings, respectively. 6 transfer tasks are performed to enable unbiased evaluation.

Digits is a common UDA benchmark for digit recognition, utilizing three subsets: SVHN (S) [48], MNIST (M) [49], and USPS (U) [50]. We use the training sets from each domain to train our model, and publish the recognition results using the standard test set from the target domain.

ImageCLEF-DA [51] is a benchmark dataset for the ImageCLEF 2014 domain adaptation challenge, constructed by picking 12 common categories shared by the three public datasets, each of which is designated a domain: Caltech-256 (C), ImageNet ILSVRC 2012 (I), and Pascal VOC 2012 (P). Each category has 50 photographs, and each domain has 600 images. We develop six transfer tasks using all domain combinations.

VisDA-2017 [52] is a difficult simulation-to-real dataset with two very separate domains: Synthetic, renderings of 3D models from various perspectives and under various lighting conditions; Real, natural images. Over 280K photos from 12 different classes make up its training, validation, and test domains.

DomainNet [53] is a dataset that encompasses a wide range of common objects across six distinct domains. Each domain consists of 345 categories of objects. These domains encompass Clipart (clp), Real (rel), Sketch (skt), Infograph (inf), painting (pnt), and quickdraw (qdr). We develop 30 transfer tasks using all domain combinations.

Baseline Methods for UDA. We compare our method with state-of-the-art methods on UDA tasks, including MinEnt [54], DANN [5], DSAN [6], CDAN+E [55], CDAN+BSP [56], CDAN+TN [57], rRGrad+CAT [28], SWD [58], MSTN+DSBN [59], BNM [60], DCAN [61], SHOT [62], ATDOC-NA [63], CGDM [64], TVT [30], CDTrans

TABLE I
COMPARISON WITH SOTA UDA METHODS ON OFFICE-HOME. ^{o} IMPLIES ITS PRE-TRAINED FROM ON IMAGENET-21K INSTEAD OF IMAGENET-1K. * IS PRE-TRAINED FROM CLIP. THE BEST PERFORMANCE IS MARKED AS BOLD.

Method	Ar→Cl	Ar→Pr	Ar→Re	Cl→Ar	Cl→Pr	Cl→Re	Pr→Ar	Pr→Cl	Pr→Re	Re→Ar	Re→Cl	Re→Pr	Avg.	DSP(M)
ResNet50:														
DCAN	54.5	75.7	81.2	67.4	74.0	76.3	67.4	52.7	80.6	74.1	59.1	83.5	70.5	283.60
BNM	52.3	73.9	80.0	63.3	72.9	74.9	61.7	49.5	79.7	70.5	53.6	82.2	67.9	283.60
ATDOC-NA	58.3	78.8	82.3	69.4	78.2	78.2	67.1	56.0	82.7	72.0	58.2	85.5	72.2	283.60
DSAN	54.4	70.8	75.4	60.4	67.8	68,0	62.6	55.9	78.5	73.8	60.6	83.1	67.6	283.60
SHOT	57.1	78.1	81.5	68.0	78.2	78.1	67.4	54.9	82.2	73.3	58.8	84.3	71.8	283.60
DAPL*	54.1	84.3	84.8	74.4	83.7	85.0	74.5	54.6	84.8	75.2	54.7	83.8	74.5	**0.00**
CLIP*	51.6	81.9	82.6	71.9	81.9	82.6	71.9	51.6	82.6	71.9	51.6	81.9	72.0	**0.00**
Baseline*	52.1	69.7	75.2	52.5	64.3	63.9	56.2	48.4	74.9	68.9	54.4	82.7	63.7	460.40
CMKD*	65.9	86.6	87.3	74.4	87.7	85.8	75.9	64.4	87.9	79.1	67.2	90.0	79.3	460.40
CMKD+RST*	64.9	**87.3**	87.1	74.2	87.5	85.4	74.5	63.0	88.0	78.1	66.5	89.6	78.8	**1.23**
CMKD+FixMatch*	**67.2**	**87.3**	**87.7**	**76.7**	**88.2**	**86.0**	**76.8**	**66.8**	**88.4**	**79.5**	**68.4**	**90.4**	**80.3**	460.40
ViT-B-16:														
CDTrans	68.8	85.0	86.9	81.5	87.1	87.3	79.6	63.3	88.2	82.0	66.0	90.6	80.5	1030.20
TVT^{o}	74.9	86.8	89.4	82.8	88.0	88.3	79.8	71.9	90.1	85.5	74.6	90.6	83.6	1030.20
SDAT^{o}	70.8	87.0	90.5	85.2	87.3	89.7	84.1	70.7	90.6	88.3	75.5	92.1	84.3	1030.20
SSRT^{o}	75.2	89.0	91.1	85.1	88.3	90.0	85.0	74.2	91.3	85.7	78.6	91.8	85.5	1030.20
PMTrans^{o}	**81.2**	91.6	92.4	**88.9**	91.6	**93.0**	**88.5**	80.0	**93.4**	**89.5**	**82.4**	94.5	88.9	1030.20
CLIP*	67.8	89.0	89.8	82.9	89.0	89.8	82.9	67.8	89.8	82.9	67.8	89.0	82.4	**0.00**
Baseline*	71.2	83.6	88.5	78.9	85.8	85.5	71.9	71.2	86.3	81.8	73.2	90.7	80.7	1034.80
CMKD*	79.4	94.2	92.7	86.3	93.4	92.2	86.7	79.5	92.1	88.2	81.2	94.5	88.4	1034.80
CMKD+RST*	78.9	93.8	92.7	85.3	93.0	91.7	85.3	78.4	92.0	87.2	80.3	94.1	87.7	**1.02**
CMKD+FixMatch*	80.9	**94.6**	**92.7**	87.1	**93.7**	92.5	87.8	**80.2**	92.3	89.2	82.1	**94.6**	**89.0**	1034.80

[7], ADDA [65], ADR [66], CyCADA [67], SSRT [68], CDAN+MCC [69], DAN [4], D-CORAL [70], MADA [71], SDAT [26], DAPL [12], and PMTrans [31].

Baseline Methods for PEFT. We compare our method with SoTA PEFT methods on UDA tasks, including LoRA [16], BitFit [72], AdaLoRA [73].

Implementation Details. We use a ResNet50 [19] backbone for Office-Home, Office-31, Digits, ImageCLEF experiments and a ResNet-101 [19] backbone for VisDA-2017. Additionally, we also report the performance of ViT-B-16 [20] backbone on all 6 benchmarks. The models used in our experiments are pre-trained from CLIP [8]. The "Baseline" indicates directly training a backbone on the source domain and testing on the target domain. For all tasks, mini-batch stochastic gradient descent (SGD) with momentum of 0.9 and the weight decay ratio 5e-4 are adopted to optimize the training process. During training, the text encoder and "Batch Normalization" in ResNet [19] is frozen, and for ViT-B-16, we set the learning rate of the visual encoder to 3e-6 for most of our tasks, while the learning rate for VisDA is set to 3e-7. The learning rate of the task head is set to 1,000 times the visual encoder. For ResNet, we set the initial learning rate to 3e-9 on the VisDA dataset and 3e-7 for the remaining datasets. Additionally, the learning rate for the task head is configured to be 10,000 times that of the visual encoder. The tradeoff λ_2 is set to 0.1, while λ_1 and λ_3 adopt dynamic mechanism like DSAN to suppress noisy activations at the early stages of training: $\lambda = \frac{2 \cdot \beta}{\exp(-1.0 \cdot \mu)} - 1$. μ is the ratio of the current training iteration discourse to the maximum number of rounds set to 1e4. a of λ_1 is set to 0.25 and of λ_3 to 0.025. The batch size is set to 32 while Digits task is set to 64, total epoch set to 20, the random seed set to 42, the factor of label smoothing set to 0.1. During CMKD training, the prompt we applied in most benchmarks is "a photo of [CLASS]", while in digits benchmark, the prompt is designed as "a photo of the number: [CLASS]". When utilizing RST, the value of τ is set to 1e-6 and 1e-5 for ViT and ResNet respectively. All the experiments are conducted with 1x NVIDIA GTX 3090 with 24G RAM and the PyTorch framework is implemented to perform our experiments. For a more comprehensive implementation, please refer to Appendix.A.

Evaluation. In addition to the commonly used Accuracy metrics for UDA, we propose the inclusion of **D**own**S**tream **P**arameters (**DSP**) as a measure to assess the model's storage overhead during deployment for downstream tasks. The calculation of DSP involves multiplying the number of migration tasks by the total number of parameters to be stored. To illustrate, let's consider the Office-Home benchmark as an example. In the case of training the ViT-B model with all parameters, the visual encoder comprises 86.2 million parameters, the task header includes 0.03 million parameters, and the downstream task model contains 86.23 million parameters. Given that there are 12 sub-tasks on Office-Home, the corresponding DSP equals 1034.80 ($86.23 * 12 \approx 1034.80$). When applying RST, we calculate the corresponding DSP by sum up the number of non-zero elements in the visual encoder and the classifier parameters. For the other PEFT methods, the corresponding DSP is calculated by sum up the number of additional parameters and classifier parameters. See Appendix.C for details on the parameters of the backbones.

B. Comparison with SoTA UDA Methods

We compared our approach with the state-of-the-art unsupervised domain adaptation (UDA) method on both ResNet and Transformer models to validate its effectiveness and flexibility. Additionally, we discovered that combining CMKD with FixMatch [74] can yield further performance improvements, and the implementation details are described in Appendix.B.

TABLE II
COMPARISON WITH SOTA UDA METHODS ON OFFICE-31.

Method	A→D	A→W	D→A	D→W	W→A	W→D	Avg.	DSP(M)
ResNet50:								
CDAN+E	92.9	94.1	71.0	98.6	69.3	**100.0**	87.7	141.40
rRGrad+CAT	90.8	94.4	72.2	98.0	70.2	**100.0**	87.6	141.40
CDAN+BSP	93.0	93.3	73.6	98.2	72.6	**100.0**	88.5	141.40
CDAN+TN	94.0	**95.7**	73.4	98.7	74.2	**100.0**	89.3	141.40
DSAN	90.2	93.6	73.5	98.3	74.8	**100.0**	88.4	141.40
BNM	90.3	91.5	70.9	98.5	71.6	**100.0**	86.8	141.40
SHOT	94.0	90.1	74.7	98.7	74.3	99.8	88.6	141.40
CLIP⋆	74.3	67.8	72.6	67.8	72.6	74.3	71.6	**0.00**
Baseline⋆	85.1	77.1	70.1	97.1	67.7	99.6	82.8	229.99
CMKD⋆	95.0	91.8	79.9	98.6	80.5	**100.0**	90.9	229.99
CMKD+RST⋆	95.0	90.9	80.7	97.4	81.0	**100.0**	90.8	**0.31**
CMKD+FixMatch⋆	**95.8**	93.6	**81.3**	**99.1**	**81.6**	**100.0**	**91.9**	229.99
ViT-B-16:								
CDTrans	97.0	96.7	81.1	99.0	81.9	**100.0**	92.6	514.94
TVT^o	96.4	96.4	84.9	99.4	86.1	**100.0**	93.8	514.94
SSRT^o	98.6	97.7	83.5	99.2	82.2	**100.0**	93.5	514.94
PMTrans^o	**99.4**	**99.1**	**85.7**	**99.6**	**86.3**	**100.0**	**95.0**	514.94
CLIP⋆	79.9	76.9	78.9	76.9	78.9	79.9	78.6	**0.00**
Baseline⋆	87.3	85.8	76.3	98.7	75.9	**100.0**	87.3	517.30
CMKD⋆	96.6	97.4	84.4	99.1	84.5	**100.0**	93.4	517.30
CMKD+RST⋆	95.2	96.9	84.0	99.1	84.2	**100.0**	93.2	**0.29**
CMKD+FixMatch⋆	98.0	98.4	85.3	99.4	85.4	**100.0**	94.4	517.30

TABLE III
COMPARISON WITH SOTA UDA METHODS ON DIGITS.

Method	S→M	U→M	M→U	Avg.	DSP(M)
LetNet5:					
ADDA	76.0	90.1	89.4	85.2	**0.03**
ADR	95.0	93.1	93.2	93.8	**0.03**
CDAN+E	89.2	98.0	95.6	94.3	**0.03**
CyCADA	90.4	96.5	95.6	94.2	**0.03**
rRGrad+CAT	98.8	96.0	94.0	96.3	**0.03**
DSAN	90.1	95.3	96.9	94.1	**0.03**
SWD	98.9	97.1	**98.1**	98.0	**0.03**
SHOT	**98.9**	**98.0**	97.9	**98.3**	**0.03**
ResNet50:					
CLIP⋆	58.4	58.4	47.6	54.8	**0.00**
Baseline⋆	81.8	93.1	94.6	89.8	114.93
CMKD⋆	96.4	**97.8**	**97.1**	**97.1**	114.93
CMKD+RST⋆	95.6	97.1	95.7	96.1	**0.23**
CMKD+Fixmatch⋆	97.0	97.2	96.1	96.8	114.93
ViT-B-16:					
CLIP⋆	55.5	55.5	62.9	58.0	**0.00**
Baseline⋆	85.7	96.8	**97.6**	93.4	258.62
CMKD⋆	98.3	98.9	97.8	98.3	258.62
CMKD+RST⋆	97.3	96.8	96.1	96.7	**0.72**
CMKD+Fixmatch⋆	**98.6**	**99.1**	**97.8**	**98.5**	258.62

TABLE IV
COMPARISON WITH SOTA UDA METHODS ON IMAGECLEF-DA.

Method	I→P	I→C	P→I	P→C	C→I	C→P	Avg.	DSP(M)
ResNet50:								
DAN	75.0	93.3	86.2	91.3	84.1	69.8	83.3	141.14
DANN	75.0	96.2	86.0	91.5	87.0	74.3	85.0	141.14
D-CORAL	76.9	93.6	88.5	91.6	86.8	74.0	85.2	141.14
MADA	75.0	96.0	87.9	92.2	88.8	75.2	85.8	141.14
CDAN+E	77.7	97.7	90.7	94.3	91.3	74.2	87.7	141.14
DSAN	80.2	97.2	93.3	95.9	93.8	80.8	90.2	141.14
CLIP⋆	81.3	96.5	94.8	96.5	94.8	81.3	90.9	**0.00**
Baseline⋆	78.2	95.8	90.3	93.5	91.0	75.8	87.4	229.87
CMKD⋆	81.5	97.8	94.5	**98.6**	96.0	**81.5**	91.7	229.87
CMKD+RST⋆	**83.3**	**98.8**	**97.0**	97.0	**96.7**	80.5	**92.2**	**0.21**
CMKD+Fixmatch⋆	81.0	98.7	95.8	97.8	95.7	81.3	91.7	229.87
ViT-B-16:								
CLIP⋆	80.7	93.3	96.3	93.3	96.3	80.7	90.1	**0.00**
Baseline⋆	82.3	97.5	93.8	94.5	90.5	75.6	89.0	517.23
CMKD⋆	85.3	99.5	98.0	99.2	**98.3**	**85.0**	94.2	517.23
CMKD+RST⋆	85.0	99.5	98.0	99.0	98.2	83.8	93.9	**0.08**
CMKD+Fixmatch⋆	**85.5**	**99.7**	**98.5**	**99.3**	98.0	**85.0**	**94.3**	517.23

Moreover, when CMKD is integrated with RST, it significantly reduces the number of parameters required for downstream tasks while only experiencing a slight decrease in performance.

Results on Office-Home. Table I shows the quantitative results of methods. As expected, CMKD achieves comparable performance **(88.4%)** with SoTA PMTrans **(88.9%)**. By incorporating CMKD with FixMatch, our approach surpasses PMTrans and emerges as the new SoTA **(89.0%)**. Our method demonstrates remarkable results on the CNN architecture as well, surpassing DAPL by a significant margin of **+5.8%** and outperforming zero-shot performance of CLIP with a large gap of **+8.3%** in terms of average accuracy. Although the DSP of CMKD significantly surpasses that of CLIP and DAPL, when integrated with RST, a mere **1.23M** parameters count is sufficient to yield remarkable outcomes across 12 downstream tasks.

Results on Office-31. Table II shows the quantitative comparison with various UDA algorithms. The adaptability of CMKD and the parameter-efficient learning capability of RST have been duly confirmed. In the case of CMKD, it achieved comparable performance to the SoTA on both CNN and Transformer architectures, with average accuracy of **90.9%** and **93.4%** respectively. With the utilization of RST (**90.8%**, **93.2%**), a mere **0.31M** and **0.29M** parameters are needed to uphold comparable performance to full parametric training.

Results on Digits. In this benchmark, the previous methods have all employed LeNet-5 as the backbone and achieved favorable outcomes for digits adaptation tasks. Intriguingly, as shown in Table III, despite undergoing large-scale visual-language pre-training, the zero-shot inference of CLIP falls significantly short of the previous methods and even lags behind the baseline performance. Interestingly, when employing CMKD based on ViT-B, the model attains state-of-the-art performance (**98.3%**). Moreover, when combined with FixMatch, the performance is further enhanced to **98.5%**, which represents a remarkable improvement of **+40.5%** compared to CLIP. This clearly demonstrates the superiority of CMKD, even in cases where the guidance from the teacher's perspective is insufficient. CMKD solely considers the supervised signals from the teacher as a reference rather than a learning target.

Results on ImageCLEF-DA. Table IV shows the quantitative results with the CNN-based and ViT-based methods. Overall, CMKD achieves the best performance on each task with **94.3%** accuracy and outperforms the SoTA methods. Numerically, under the ResNet50, CMKD surpasses the SoTA methods with an increase of **+0.8%** accuracy over CLIP, **+1.5%** accuracy over DSAN, respectively.

Results on VisDA-2017. As shown in Table V, due to the similar distribution between the pre-training data of CLIP and the target domain (real) of VisDA, CLIP's zero-shot inference capability under the ViT-B architecture reached **88.0%**, significantly surpassing the baseline (**78.6%**). In contrast to the digits benchmark, representing a scenario where teacher

TABLE V
COMPARISON WITH SOTA UDA METHODS ON VISDA-2017.

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Avg.	DSP(M)
ResNet101:														
CDAN+E	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.9	42.52
DSAN	90.9	66.9	75.7	62.4	88.9	77.0	93.7	75.1	92.8	67.6	89.1	39.4	75.1	42.52
BNM	89.6	61.5	76.9	55.0	89.3	69.1	81.3	65.5	90.0	47.3	89.1	30.1	70.4	42.52
MSTN+DSBN	94.7	86.7	76.0	72.0	95.2	75.1	87.9	81.3	91.1	68.9	88.3	45.5	80.2	42.52
CGDM	92.8	85.1	76.3	64.5	91.0	93.2	81.3	79.3	92.4	83.0	85.6	44.8	80.8	42.52
SHOT	95.5	87.5	80.1	54.5	93.6	94.2	80.2	80.9	90.0	89.9	87.1	58.4	82.7	42.52
CDAN+MCC	94.5	80.8	78.4	65.3	90.6	79.4	87.5	**82.2**	**94.7**	81.0	86.0	44.6	80.4	42.52
DAPL*	97.8	83.1	88.8	77.9	97.4	91.5	94.2	79.7	88.6	89.3	92.5	62.0	86.9	**0.00**
CLIP*	**98.2**	83.9	**90.5**	73.5	97.2	84.0	**95.3**	65.7	79.4	89.9	91.8	63.3	84.4	**0.00**
Baseline*	92.6	39.1	73.3	78.5	83.1	54.1	90.6	29.0	78.8	67.3	**97.5**	14.4	66.5	56.31
CMKD*	97.8	87.6	87.8	75.4	96.8	95.7	87.4	79.3	91.2	89.8	92.0	**63.4**	87.0	56.31
CMKD+RST*	97.5	86.1	87.1	78.1	96.8	95.1	90.2	**82.2**	87.3	91.8	94.5	53.0	86.6	**0.03**
CMKD+FixMatch*	**98.2**	**88.0**	89.1	**79.3**	**98.3**	**96.5**	92.8	78.8	91.9	**94.7**	94.7	50.9	**87.8**	56.31
ViT-B-16:														
CDTrans	97.1	90.5	82.4	77.5	96.6	96.1	93.6	**88.6**	**97.9**	86.9	90.3	62.8	88.4	85.81
TVT^o	92.9	85.6	77.5	60.5	93.6	98.2	89.4	76.4	93.6	92.0	91.7	55.7	83.9	85.81
SDAT^o	98.4	90.9	85.4	82.1	98.5	97.6	**96.3**	86.1	96.2	96.7	92.9	56.8	89.8	85.81
SSRT-B^o	98.9	87.6	89.1	**84.7**	98.3	**98.7**	96.2	81.0	94.8	97.9	94.5	43.1	88.7	85.81
PMTrans^o	98.9	93.7	84.5	73.3	99.0	98.0	96.2	67.8	94.2	98.4	96.6	49.0	87.5	85.81
CLIP*	99.3	91.7	93.9	74.3	98.4	94.3	90.3	78.2	78.3	97.3	95.2	64.8	88.0	**0.00**
Baseline*	**99.5**	83.0	84.4	76.3	97.4	78.1	95.9	20.8	97.2	94.0	**98.6**	17.6	78.6	86.21
CMKD*	99.4	94.6	**91.5**	78.9	98.7	97.3	93.3	81.3	91.8	97.9	96.9	61.7	90.3	86.21
CMKD+RST*	99.3	91.3	91.0	77.7	99.0	96.8	94.0	81.8	91.1	97.1	96.8	52.8	89.1	**0.06**
CMKD+FixMatch*	**99.5**	**95.2**	91.3	81.5	**99.4**	98.0	95.2	83.5	95.5	**98.5**	96.8	**67.0**	**91.8**	86.21

vision is superior, CMKD enables the student perspective to achieve further performance improvement beyond the limits set by the teacher vision. It achieves a performance of **90.3%**, outperforming the baseline and CLIP by **+11.5%** and **+2.3%**, respectively. When combined with FixMatch, the performance even reached **91.8%**. The validity is also demonstrated in the CNN structure, which achieved a score of **87.8%**, surpassing the baseline and CLIP by **+21.3%** and **+4.4%** respectively. When CMKD is combined with RST, only need **0.06 M** parameters can reach **89.1%**, achieving approximate results with SDAT (**89.9%**) while the latter requires **85.81 M**.

Results on DomainNet. CMKD and CMKD+FixMatch emerged as the new state-of-the-art models, achieving respective accuracies of **53.9%** and **54.3%**, surpassing the previous SoTA model, PMTrans, which achieved **52.4%**, as shown in Table VI. While CLIP does not rely on downstream task parameters, its average accuracy of **6.6%** falls significantly behind other methods. By combining CMKD with RST, a mere **15.71M** DSP is needed to achieve a remarkable accuracy of **52.2%**, yielding results on par with PMTrans.

Analysis. The experimental results provide several insightful observations. Firstly, despite the extensive visual-language pre-training of CLIP, its zero-shot performance still significantly lags behind that of previous SoTA. Secondly, CMKD demonstrates remarkable performance, achieving SoTA on five benchmarks when combined with FixMatch. This indicates that future research could leverage CMKD as a baseline to explore more effective techniques for enhancing the potential of VLP models on UDA tasks. Thirdly, both CMKD and RST exhibit great flexibility and applicability across ResNet and ViT structures. Fourthly, despite a slight performance degradation with RST, the deployment cost, as measured by DSP, drops by more than 99%, demonstrating its efficiency in reducing resource requirements. Fifthly, regardless of the sufficiency of guidelines from the teacher model (e.g., based on the ViT-B model, CLIP's zero-shot inference result on Office-Home is 82.4%, while it is only 6.6% on DomainNet), CMKD effectively balances the perspectives of both the student and the teacher, leading to improved model performance.

C. Comparison with SoTA PEFT Methods

RST can be regarded as a type of PEFT designed specifically for unsupervised domain adaptation. Therefore, in this subsection, we will conduct a fair comparison with advanced PEFT methods. In our experiments, we consistently utilize the CMKD approach proposed in this paper as the UDA training method, combined with various PEFT techniques. We compare the average accuracy and DSP of these approaches. This section of the experiments is performed on the Office-Home benchmark. All configurations will remain the same except for PEFT-related hyperparameters. To ensure that the DSPs of the related methods are comparable, we set the ranks of our LoRAs [16] as 1 and 4, represented as $\text{LoRA}_{r=1}$ and $\text{LoRA}_{r=4}$ respectively. For AdaLoRA [73], the initial ranks are set as 8 and 2, and the corresponding target ranks as 4 and 1, denoted as $\text{AdaLoRA}_{r=2\to1}$ and $\text{AdaLoRA}_{r=8\to4}$. Furthermore, we also compare "Fine-Tuning" with "Linear-Probe". The former refers to training the entire backbone and task-head, whereas the latter only trains the task-head. As AdaLoRA [73] and Adapter do not offer an implementation of CNNs, comparisons can only be conducted within the framework of the ViT architecture.

As shown in Table VII, our method demonstrates outstanding performance. With the ResNet50 architecture, we achieve an average accuracy of **78.8%** using only **1.23M** DSP, resulting in a mere **-0.7%** accuracy drop compared to Fine-Tuning. Moreover, the DSP employed is only **0.2%** of that used in

TABLE VI
COMPARISON WITH SOTA UDA METHODS ON DOMAINNET.

SVD	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	14.7	31.9	10.1	45.3	36.5	27.7
inf	22.9	-	24.2	2.5	33.2	21.3	20.0
pnt	33.6	1530	-	4.4	46.1	30.7	26.0
qdr	15.5	2.2	6.4	-	11.1	10.2	9.1
rel	41.2	18.1	44.0	4.6	-	31.6	27.9
skt	44.2	15.2	37.0	10.3	44.7	-	30.3
Avg	31.5	13.1	29.0	6.4	36.1	26.1	23.6
DSP(M)	726.20						

BNM	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	12.1	33.1	6.2	50.8	40.2	28.5
inf	26.6	-	28.5	2.4	38.5	18.1	22.8
pnt	39.9	12.2	-	3.4	54.5	36.2	29.2
qdr	17.8	1.0	3.6	-	9.2	8.3	8.0
rel	48.6	13.2	49.7	3.6	-	33.9	29.8
skt	54.9	12.8	42.3	5.4	51.3	-	33.3
Avg	37.6	10.3	31.4	4.2	40.9	27.3	25.3
DSP(M)	726.20						

CGDM	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	16.9	35.3	10.8	53.5	36.9	30.7
inf	27.8	-	28.2	4.4	48.2	22.5	26.2
pnt	37.7	14.5	-	4.6	59.4	33.5	30.0
qdr	14.9	1.5	6.2	-	10.9	10.2	8.7
rel	49.4	20.8	47.2	4.8	-	38.2	32.0
skt	50.1	16.5	43.7	11.1	55.6	-	35.4
Avg	36.0	14.0	32.1	7.1	45.5	28.3	27.2
DSP(M)	726.20						

MDD	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	20.5	40.7	6.2	52.5	42.1	32.4
inf	33.0	-	33.8	2.6	46.2	24.5	28.0
pnt	43.7	20.4	-	2.8	51.2	41.7	32.0
qdr	18.4	3.0	8.1	-	12.9	11.8	10.8
rel	52.8	21.6	47.8	4.2	-	41.2	33.5
skt	54.3	17.5	43.1	5.7	54.2	-	35.0
Avg	40.4	16.6	34.7	4.3	43.4	32.3	28.6
DSP(M)	726.20						

SCDA	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	18.6	39.3	5.1	55.0	44.1	32.4
inf	29.6	-	34.0	1.4	46.3	25.4	27.3
pnt	44.1	19.0	-	2.6	56.2	42.0	32.8
qdr	30.0	4.9	15.0	-	25.4	19.8	19.0
rel	54.0	22.5	51.9	2.3	-	42.5	34.6
skt	55.6	18.5	44.7	6.4	53.2	-	35.7
Avg	42.6	16.7	37.0	3.6	47.2	34.8	30.3
DSP(M)	726.20						

CDTrans	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	29.4	57.2	26.0	72.6	58.1	48.7
inf	57.0	-	54.4	12.8	69.5	48.4	48.4
pnt	62.9	27.4	-	15.8	72.1	53.9	46.4
qdr	44.6	8.9	29.0	-	42.6	28.5	30.7
rel	66.2	31.0	61.5	16.2	-	52.9	45.6
skt	69.0	29.6	59.0	27.2	72.5	-	51.5
Avg	59.0	25.3	52.2	19.6	65.9	48.4	45.2
DSP(M)	2581.94						

SSRT^o	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	33.8	60.2	19.4	75.8	59.8	49.8
inf	55.5	-	54.0	9.0	68.2	44.7	46.3
pnt	61.7	28.5	-	8.4	71.4	55.2	45.0
qdr	42.5	8.8	24.2	-	37.6	33.6	29.3
rel	69.9	37.1	66.0	10.1	-	58.9	48.4
skt	70.6	32.8	62.2	21.7	73.2	-	52.1
Avg	60.0	28.2	53.3	13.7	65.3	50.4	45.2
DSP(M)	2581.94						

PMTRans^o	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	34.2	62.7	**32.5**	**79.3**	63.7	54.5
inf	67.4	-	61.1	**22.2**	**78.0**	57.6	57.3
pnt	69.7	33.5	-	**23.9**	**79.8**	61.2	53.6
qdr	54.6	17.4	**38.9**	-	**49.5**	41.0	40.3
rel	74.1	35.3	**70.0**	**25.4**	-	61.1	53.2
skt	73.8	33.0	62.6	**30.9**	77.5	-	55.6
Avg	67.9	30.7	59.1	**27.0**	**72.8**	56.9	52.4
DSP(M)	2581.94						

CLIP*	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	9.6	5.4	1.7	8.0	6.9	6.3
inf	9.8	-	5.4	1.7	8.0	6.9	6.3
pnt	9.8	9.6	-	1.7	8.0	6.9	6.3
qdr	9.8	9.6	5.4	-	8.0	6.9	6.3
rel	9.8	9.6	5.4	1.7	-	6.9	6.3
skt	9.8	9.6	5.4	1.7	8.0	-	6.3
Avg	9.8	9.6	5.4	1.7	8.0	6.9	6.6
DSP(M)	**0.00**						

CMKD*	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	43.5	65.9	22.8	78.2	66.7	55.4
inf	71.3	-	64.5	19.8	78.0	61.0	58.9
pnt	**70.6**	42.3	-	19.1	76.8	65.0	54.8
qdr	50.3	27.2	37.1	-	46.4	41.5	40.5
rel	77.0	48.0	69.2	18.5	-	66.0	55.7
skt	76.1	44.7	67.5	22.9	78.0	-	57.8
Avg	69.1	41.1	60.8	20.6	71.5	60.0	53.9
DSP(M)	2591.30						

+RST*	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	43.2	64.3	20.7	77.5	65.5	54.2
inf	70.6	-	62.9	17.7	77.2	60.3	57.7
pnt	68.4	40.5	-	18.1	75.9	63.8	53.3
qdr	48.5	22.5	34.3	-	41.2	37.6	36.8
rel	75.8	47.2	67.9	16.8	-	64.8	54.5
skt	74.8	43.1	65.9	21.6	77.4	-	56.6
Avg	67.7	39.3	59.1	19.0	69.8	58.4	52.2
DSP(M)	**15.71**						

+FixMatch*	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	**44.3**	**66.2**	22.6	78.4	**66.9**	**55.7**
inf	**71.6**	-	**64.7**	18.5	**78.0**	**61.0**	**58.8**
pnt	70.5	**43.4**	-	19.5	77.0	**65.3**	**55.1**
qdr	**51.6**	**29.3**	38.0	-	46.5	**43.3**	**41.7**
rel	**77.1**	**48.7**	69.6	19.1	-	**66.2**	**56.1**
skt	**76.2**	**45.2**	**68.1**	23.0	**78.9**	-	**58.3**
Avg	**69.4**	**42.2**	**61.3**	20.5	71.8	**60.5**	**54.3**
DSP(M)	2591.30						

Fine-Tuning, highlighting the efficiency of our approach. The effectiveness of our method is further confirmed with the ViT-B architecture. We attain an average accuracy of **87.7%** with just **1.02M** DSP, a significant reduction in downstream task model storage compared to Fine-Tuning. Furthermore, our approach significantly outperforms the state-of-the-art method of PEFT in the LLM domain. For instance, with the ViT-B architecture, RST outperforms BitFiT, LoRA$_{r=1}$, and AdaLoRA$_{r=2\to1}$ by **+0.7%**, **+1.2%**, and **+1.0%**, respectively. Additionally, RST requires **-0.38**, **-2.38**, and **-1.54** less DSP than BitFiT, LoRA$_{r=1}$, and AdaLoRA$_{r=2\to1}$, respectively. Surprisingly, we have observed that the advanced LoRA series does not appear to outperform in the case of large data trained but small-scale models like CLIP. In addition to RST we proposed, even a simple method like BitFIT, which only trains Bias, outperforms the entire LoRA family, both in terms of average accuracy and DSP.

Throughout the experiments, we did not observe that the PEFT method outperformed Fine-Tuning in terms of average accuracy, which differs from the LLM domain. Therefore, the pursuit of efficient fine-tuning on large data with small models presents a promising and worthwhile problem to explore, particularly in scenarios like the UDA domain with numerous downstream tasks.

D. Ablation Study

Baseline Configuration. To the best of our knowledge, we are the first to fine-tuning with visual-language pre-training model in the UDA domain. Therefore, it is necessary to explore the effect of hyperparameters on the Baseline performance. In this section, we focus on investigating the impact of the learning rate. Following the previous UDA approaches, for the Office-Home dataset, we set the initial learning rate of the backbone to 3e-4 and the task head learning rate to 10 times that of the backbone (referred to as Configuration1). However, due to CLIP being pre-trained on a large-scale dataset, the model parameters are particularly sensitive to the learning rate. As a result, in Configuration2, we set the initial learning rate of the backbone to 3e-7 for ResNet50 and 3e-6 for ViT-B, while the task head learning rate is set to 10,000 and 1,000 times that of the backbone. Besides, "Batch Normalization" in RseNet is Frozen. The remaining hyperparameter details are kept the same, as detailed in the "Implementation Details" of Section IV.A. In addition to exploring the hyperparameter settings, we also aim to understand the impact of different pre-training models on the baseline performance. Therefore, we conduct experiments using Configuration1 on common pre-training models (e.g., ResNet50 pre-trained on ImageNet-1k and ViT-B pre-trained on ImageNet-21k), denoted as "ImageNet-1k" or "ImageNet-21k" on the Table VIII. We also utilized configuration1 and configuration2 for the pre-training model of CLIP,

TABLE VII
COMPARISON WITH SOTA PEFT METHODS ON OFFICE-HOME.

Method	Ar→Cl	Ar→Pr	Ar→Re	Cl→Ar	Cl→Pr	Cl→Re	Pr→Ar	Pr→Cl	Pr→Re	Re→Ar	Re→Cl	Re→Pr	Avg.	DSP(M)
ResNet50:														
Fine-Tuning⋆	65.9	86.6	87.3	74.4	87.7	85.8	75.9	64.4	87.9	79.1	67.2	90.0	79.3	460.40
Linear-Probe⋆	59.2	86.3	87.1	73.7	87.6	85.4	72.4	58.8	87.2	75.6	60.0	89.5	76.9	0.80
LoRA$^{\star}_{r=1}$	59.1	85.2	87.1	72.5	86.2	84.6	73.0	60.7	87.1	76.5	61.1	88.7	76.8	1.76
LoRA$^{\star}_{r=4}$	60.6	86.1	**87.4**	72.5	87.2	**85.4**	73.2	60.8	87.8	75.9	61.6	89.4	77.3	4.04
BitFit⋆	60.1	86.2	87.1	73.8	87.8	85.1	73.1	60.4	87.4	76.5	61.4	**89.6**	77.4	**1.12**
RST⋆	**64.9**	**87.3**	87.1	**74.2**	**87.5**	**85.4**	**74.5**	**63.0**	**88.0**	**78.1**	**66.5**	**89.6**	**78.8**	1.23
ViT-B-16:														
Fine-Tuning⋆	79.4	94.2	92.7	86.3	93.4	92.2	86.7	79.5	92.1	88.2	81.2	94.5	88.4	1034.80
Linear-Probe⋆	76.5	93.0	92.2	83.9	93.0	90.5	83.2	76.7	91.9	85.2	78.6	93.5	86.5	0.40
LoRA$^{\star}_{r=1}$	76.1	93.0	92.1	84.3	92.9	91.0	83.3	76.8	**92.0**	85.0	78.6	93.4	86.5	3.40
LoRA$^{\star}_{r=4}$	75.7	93.0	92.3	83.8	**93.2**	90.0	83.8	77.1	91.8	85.1	78.3	93.4	86.5	12.0
BitFit⋆	77.4	93.0	92.4	84.3	93.0	91.0	83.8	77.8	91.9	85.7	79.7	93.7	87.0	1.40
AdaLoRA$^{\star}_{r=2\to1}$	76.2	93.0	92.3	84.5	93.1	91.0	83.6	77.1	**92.0**	85.2	78.5	93.3	86.7	2.56
AdaLoRA$^{\star}_{r=8\to4}$	76.4	93.1	92.2	84.2	93.0	90.6	83.6	76.7	91.9	85.4	78.8	93.4	86.6	6.09
RST⋆	**78.9**	**93.8**	**92.7**	**85.3**	93.0	**91.7**	**85.3**	**78.4**	**92.0**	**87.2**	**80.3**	**94.1**	**87.7**	**1.02**

TABLE VIII
BASELINE FINE-TUNING RESULTS WITH DIFFERENT CONFIGURATIONS ON OFFICE-HOME.

Method	Ar→Cl	Ar→Pr	Ar→Re	Cl→Ar	Cl→Pr	Cl→Re	Pr→Ar	Pr→Cl	Pr→Re	Re→Ar	Re→Cl	Re→Pr	Avg.
ResNet50:													
ImageNet-1K	51.0	68.2	74.8	**54.2**	63.6	**66.8**	53.6	45.4	74.6	65.6	53.6	79.3	62.5
CLIP-1⋆	24.7	25.1	36.1	14.8	29.4	25.9	17.1	28.2	38.5	45.5	45.3	59.9	32.6
CLIP-2⋆	**52.1**	**69.7**	**75.2**	52.5	**64.3**	63.9	**56.2**	**48.4**	**74.9**	**68.9**	**54.4**	**82.7**	**63.7**
ViT-B-16:													
ImageNet-21K^{o}	64.6	**86.0**	**88.6**	**81.4**	**87.4**	**88.5**	**79.2**	60.7	**88.8**	81.6	61.4	89.3	79.8
CLIP-1⋆	40.4	65.5	75.1	41.0	60.0	62.0	44.7	49.3	73.1	67.2	48.6	81.6	59.0
CLIP-2⋆	**71.2**	83.6	88.5	78.9	85.8	85.5	71.9	**71.2**	86.3	**81.8**	73.2	**90.7**	**80.7**

TABLE IX
CONTRIBUTION OF EACH PART IN CMKD BASED ON OFFICE-HOME USING RESNET50.

$\mathcal{L}_{\text{cls}}$	$\mathcal{L}_{\text{task}}$	$\mathcal{L}_{\text{distill}}$	$\mathcal{L}_{\text{reg}}$	Ar→Cl	Ar→Pr	Ar→Re	Cl→Ar	Cl→Pr	Cl→Re	Pr→Ar	Pr→Cl	Pr→Re	Re→Ar	Re→Cl	Re→Pr	Avg.
✔				52.1	69.7	75.2	52.5	64.3	63.9	56.2	48.4	74.9	68.9	54.4	82.7	63.7
✔	✔			64.2	85.6	86.9	73.0	86.4	84.6	75.1	63.4	87.0	78.4	66.4	89.3	78.4
✔		✔		63.3	83.9	85.3	71.5	85.7	82.9	73.4	61.5	86.0	77.7	64.3	88.6	77.0
✔	✔	✔		65.1	85.9	87.1	73.9	87.1	**85.8**	75.3	63.9	**87.9**	78.7	66.9	89.8	79.0
✔	✔	✔	✔	**65.9**	**86.6**	**87.3**	**74.4**	**87.7**	**85.8**	**75.9**	**64.4**	**87.9**	**79.1**	**67.2**	**90.0**	**79.3**

denoted as "CLIP-1" and "CLIP-2", respectively. This allows us to compare the contribution of the pre-training datasets from CLIP to that from ImageNet to the Baseline.

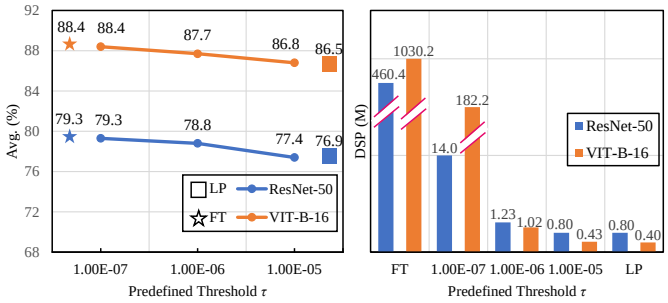
As shown in Table VIII, the configuration used in the previous method is not suitable for the VLP model, and the performance of the CLIP-based Baseline is notably inferior to that of the ImageNet pre-training model. For instance, based on ResNet50, there is a difference of **-29.9%** between CLIP-1 and ImageNet-1k. However, when Configuration2 is applied, the performance of the Baseline significantly improves, with a **+31.1%** and **+21.7%** enhancement in CLIP-2 compared to CLIP-1 for ResNet50 and ViT-B architectures, respectively. Nevertheless, based on the experimental results, CLIP's dataset does not lead to significant improvements, and the average accuracy of CLIP-2 is only **+1.2%** and **+0.9%** higher than that of ImageNet.

Contribution of components in CMKD. Contribution of the components. CMKD comprises three components: task-term $\mathcal{L}_{\text{task}}$, distillation term $\mathcal{L}_{\text{distill}}$ and regularization $\mathcal{L}_{\text{reg}}$. To evaluate the individual contributions of each component to the overall approach, we conducted an additional experiment to assess the effectiveness of average accuracy on the Office-Home dataset using ResNet50. The classification loss $\mathcal{L}_{\text{cls}}$ is used throughout the experiment. As shown in Table IX, $\mathcal{L}_{\text{task}}$ alone achieves **78.4%** and $\mathcal{L}_{\text{distill}}$ achieves **77.0%**, respectively, outperforming the baseline by **+14.7%** and **+13.3%**. The former represents the performance of the model learning on the target domain from the student's perspective, while the latter is the result of learning with the teacher's perspective (text encoder) as an aid. When the two are used in combination, the performance reaches **79.0%**, demonstrating that the student perspective and the teacher perspective complement each other. Additionally, when regularization was applied to prevent the teacher perspective from being biased, the average accuracy reached **79.3%**.

Trade-off α in CMKD. CMKD is a KD-inspired UDA method. In KD [32], a trade-off parameter "α" is used as an artificially designed hyperparameter to balance $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{distill}}$ losses. However, in the UDA domain, where real labels for knowledge transfer are lacking, we propose using a dynamic mechanism to generate a different trade-off for each sample. In Table X, we compare the experimental results of the

TABLE X
COMPARISON WITH DIFFERENT α IN CMKD BASED ON OFFICE-HOME USING RESNET50

α	Ar→Cl	Ar→Pr	Ar→Re	Cl→Ar	Cl→Pr	Cl→Re	Pr→Ar	Pr→Cl	Pr→Re	Re→Ar	Re→Cl	Re→Pr	Avg.
0.5	64.7	85.6	86.8	72.6	87.4	85.2	74.5	63.7	87.0	78.5	66.6	89.4	78.5
$sg(\exp(-\mathrm{GE}(p_h^t)))$	65.7	85.8	87.2	**75.0**	**88.3**	85.7	**76.3**	**64.5**	87.6	**79.2**	67.0	89.9	79.1
$sg(\exp(-\mathrm{KL}(p_h^t \parallel p_g^t)))$	**65.9**	**86.6**	**87.3**	74.4	87.7	**85.8**	75.9	64.4	**87.9**	79.1	**67.2**	**90.0**	**79.3**



(a) Average Accuracy in Different Threshold (b) DSP in Different Threshold

Fig. 4. Average accuracy and DSP comparison in different predefined threshold on Office-Home. LP and FT stand for Linear-Probe and Fine-Tuning, respectively.

artificial selection with the two dynamic mechanisms, namely $sg(\exp(-\mathrm{GE}(p_h^t)))$ and $sg(\exp(-\mathrm{KL}(p_h^t \parallel p_g^t)))$. The first dynamic mechanism uses entropy to estimate the uncertainty of the current task head, and when the uncertainty is large, the teacher's perspective is used as the dominant one. The second mechanism considers the consistency between the student's perspective and the teacher's perspective, and when they are consistent, the student's perspective is dominant; otherwise, the teacher's perspective is dominant. In Table X, the experimental results show that the manually designed trade-off performs poorly, achieving only **78.5%** average accuracy. In contrast, the two dynamic mechanisms outperform the former by **+0.6%** and **+0.8%**, respectively. Ultimately, we choose $sg(\exp(-\mathrm{KL}(p_h^t \parallel p_g^t)))$ as the trade-off.

Different Predefined Threshold τ in RST. Similar to the related work of PEFT, RST can control the number of downstream task parameters by adjusting the hyperparameter τ. Generally, a larger DSP (closer to Fine-Tuning) leads to better performance, while a smaller DSP (closer to Linear-Probe) results in less effectiveness. To strike a better balance between DSP and performance, we set different values of τ for residual sparse training on the Office-Home benchmark. As shown in Figure 4, when the τ is chosen as 1e-6, a better trade-off can be achieved. Specifically, for ResNet50, it achieves **78.8%** accuracy with a DSP of **1.23**, while for ViT-B-16, it achieves **87.7%** accuracy with a DSP of **1.02**. In comparison, the average accuracy only drops by **-0.7%** and **-0.5%** while DSP drops by **-99.61%** and **-99.90%**, when compared to Fine-Tuning. The flexibility and effectiveness of tau selection allow it to be chosen according to the specific task requirements.

[Additional ablation studies and analyses have been included in the Appendix. Furthermore, the Appendix showcases the performance of the proposed method on various other tasks.]

V. CONCLUSION

In this work, we have presented a comprehensive study on unsupervised domain adaptation tasks, focusing on two significant challenges that have been underexplored in the literature. Firstly, we identified the potential of large-scale visual-language pre-training models for UDA and proposed a novel method, cross-modal knowledge distillation, which effectively harnesses VLP models as teachers to guide the target domain model's learning process. Our experiments demonstrate that CMKD outperforms traditional UDA methods based on ImageNet pre-trained models, achieving state-of-the-art results in various UDA tasks. Secondly, we addressed the issue of storage overhead in UDA by introducing residual sparse training, which efficiently reduces the parameter count of VLP models while preserving competitive performance with Fine-Tuning. The RST approach allows for the deployment of UDA models across multiple transfer tasks with significantly reduced storage requirements, making it more practical and scalable for real-world applications. We believe that our findings and methodologies will be beneficial to researchers and practitioners working on UDA tasks, facilitating the development of resource-efficient domain adaptation techniques.

REFERENCES

[1] R. Xia, G. Li, Z. Huang, H. Meng, and Y. Pang, "Cbash: Combined backbone and advanced selection heads with object semantic proposals for weakly supervised object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 10, pp. 6502–6514, 2022.

[2] J. Ji, R. Shi, S. Li, P. Chen, and Q. Miao, "Encoder-decoder with cascaded crfs for semantic segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 5, pp. 1926–1938, 2020.

[3] W. Dong, T. Yang, J. Qu, T. Zhang, S. Xiao, and Y. Li, "Joint contextual representation model-informed interpretable network with dictionary aligning for hyperspectral and lidar classification," *IEEE Trans. Circuits Syst. Video Technol.*, 2023.

[4] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. Int. Conf. Mach. Learn*, 2015, pp. 97–105.

[5] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1180–1189.

[6] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He, "Deep subdomain adaptation network for image classification," *IEEE Trans. Neural Net. Learn. Syst.*, vol. 32, no. 4, pp. 1713–1722, 2020.

[7] T. Xu, W. Chen, P. Wang, F. Wang, H. Li, and R. Jin, "Cdtrans: Cross-domain transformer for unsupervised domain adaptation," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn*, 2021, pp. 8748–8763.

[9] Y. Li, H. Fan, R. Hu, C. Feichtenhofer, and K. He, "Scaling language-image pre-training via masking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 23 390–23 400.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015.

[12] C. Ge, R. Huang, M. Xie, Z. Lai, S. Song, S. Li, and G. Huang, "Domain adaptation via prompt learning," *arXiv preprint arXiv:2202.06687*, 2022.

[13] X. Dong, J. Bao, T. Zhang, D. Chen, S. Gu, W. Zhang, L. Yuan, D. Chen, F. Wen, and N. Yu, "Clip itself is a strong fine-tuner: Achieving 85.7% and 88.0% top-1 accuracy with vit-b and vit-l on imagenet," *arXiv preprint arXiv:2212.06138*, 2022.

[14] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *Int. J. Comput. Vis.*, vol. 130, no. 9, pp. 2337–2348, 2022.

[15] L. Yan, C. Han, Z. Xu, D. Liu, and Q. Wang, "Prompt learns prompt: exploring knowledge-aware generative prompt collaboration for video captioning," in *Proc. IJCAI*, 2023, pp. 1622–1630.

[16] 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," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[17] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[18] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13 733–13 742.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[20] 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," in *Proc. Int. Conf. Learn. Represent.*, 2021.

[21] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 2790–2799.

[22] W. Kim, B. Son, and I. Kim, "Vilt: Vision-and-language transformer without convolution or region supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5583–5594.

[23] Z. Wang, J. Yu, A. W. Yu, Z. Dai, Y. Tsvetkov, and Y. Cao, "Simvlm: Simple visual language model pretraining with weak supervision," in *ICLR*, 2022.

[24] J. Ke, K. Ye, J. Yu, Y. Wu, P. Milanfar, and F. Yang, "Vila: Learning image aesthetics from user comments with vision-language pretraining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 10 041–10 051.

[25] H. Li, N. Dong, Z. Yu, D. Tao, and G. Qi, "Triple adversarial learning and multi-view imaginative reasoning for unsupervised domain adaptation person re-identification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 5, pp. 2814–2830, 2021.

[26] H. Rangwani, S. K. Aithal, M. Mishra, A. Jain, and V. B. Radhakrishnan, "A closer look at smoothness in domain adversarial training," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 18 378–18 399.

[27] H. Liu, J. Wang, and M. Long, "Cycle self-training for domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 22 968–22 981.

[28] Z. Deng, Y. Luo, and J. Zhu, "Cluster alignment with a teacher for unsupervised domain adaptation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9944–9953.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.

[30] J. Yang, J. Liu, N. Xu, and J. Huang, "Tvt: Transferable vision transformer for unsupervised domain adaptation," in *Proc. WACV*, 2023, pp. 520–530.

[31] J. Zhu, H. Bai, and L. Wang, "Patch-mix transformer for unsupervised domain adaptation: A game perspective," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 3561–3571.

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

[33] S. Yang, L. Xu, M. Zhou, X. Yang, J. Yang, and Z. Huang, "Skill-transferring knowledge distillation method," *IEEE Trans. Circuits Syst. Video Technol.*, 2023.

[34] X. Zhang, X. Wang, and P. Cheng, "Unsupervised hashing retrieval via efficient correlation distillation," *IEEE Trans. Circuits Syst. Video Technol.*, 2023.

[35] C. Han, Q. Wang, Y. Cui, Z. Cao, W. Wang, S. Qi, and D. Liu, "Eˆ 2vpt: An effective and efficient approach for visual prompt tuning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023.

[36] C. Han, Q. Wang, Y. Cui, W. Wang, L. Huang, S. Qi, and D. Liu, "Facing the elephant in the room: Visual prompt tuning or full finetuning?" *Proc. Int. Conf. Learn. Represent.*, 2024.

[37] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. ACL-IJCNLP*, 2021, pp. 4582–4597.

[38] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. ACL-IJCNLP*, 2021, pp. 4582–4597.

[39] 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 *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.

[40] L. Yang, Q. Wang, J. Wang, X. Quan, F. Feng, Y. Chen, M. Khabsa, S. Wang, Z. Xu, and D. Liu, "Mixpave: Mix-prompt tuning for few-shot product attribute value extraction," in *Proc. ACL*, 2023, pp. 9978–9991.

[41] F. Ma, C. Zhang, L. Ren, J. Wang, Q. Wang, W. Wu, X. Quan, and D. Song, "Xprompt: Exploring the extreme of prompt tuning," *arXiv preprint arXiv:2210.04457*, 2022.

[42] S. Dong, Y. Feng, Q. Yang, Y. Huang, D. Liu, and H. Fan, "Efficient multimodal semantic segmentation via dual-prompt learning," *arXiv preprint arXiv:2312.00360*, 2023.

[43] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 3, no. 2, 2013, pp. 896–902.

[44] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," *arXiv preprint arXiv:2012.13255*, 2020.

[45] P. Mateos-Aparicio and A. Rodríguez-Moreno, "The impact of studying brain plasticity," *Frontiers in cellular neuroscience*, vol. 13, no. 66, 2019.

[46] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5018–5027.

[47] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 213–226.

[48] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011.

[49] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[50] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 16, no. 5, pp. 550–554, 1994.

[51] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2208–2217.

[52] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "Visda: The visual domain adaptation challenge," *arXiv preprint arXiv:1710.06924*, 2017.

[53] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1406–1415.

[54] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 281–296.

[55] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1647–1657.

[56] X. Chen, S. Wang, M. Long, and J. Wang, "Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1081–1090.

[57] X. Wang, Y. Jin, M. Long, J. Wang, and M. I. Jordan, "Transferable normalization: Towards improving transferability of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[58] C.-Y. Lee, T. Batra, M. H. Baig, and D. Ulbricht, "Sliced wasserstein discrepancy for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10 285–10 295.

[59] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han, "Domain-specific batch normalization for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7354–7362.

[60] S. Cui, S. Wang, J. Zhuo, L. Li, Q. Huang, and Q. Tian, "Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3941–3950.
[61] S. Li, C. Liu, Q. Lin, B. Xie, Z. Ding, G. Huang, and J. Tang, "Domain conditioned adaptation network," in *Proc. AAAI*, 2020, pp. 11 386–11 393.
[62] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6028–6039.
[63] J. Liang, D. Hu, and J. Feng, "Domain adaptation with auxiliary target domain-oriented classifier," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16 632–16 642.
[64] Z. Du, J. Li, H. Su, L. Zhu, and K. Lu, "Cross-domain gradient discrepancy minimization for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3937–3946.
[65] A. Chadha and Y. Andreopoulos, "Improved techniques for adversarial discriminative domain adaptation," *IEEE Trans. Image Process.*, vol. 29, pp. 2622–2637, 2019.
[66] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Adversarial dropout regularization," in *Proc. Int. Conf. Learn. Represent.*, 2017.
[67] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1989–1998.
[68] T. Sun, C. Lu, T. Zhang, and H. Ling, "Safe self-refinement for transformer-based domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 7191–7200.
[69] Y. Jin, X. Wang, M. Long, and J. Wang, "Minimum class confusion for versatile domain adaptation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 464–480.
[70] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 443–450.
[71] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *Proc. AAAI*, 2018, pp. 7618–7625.
[72] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *Proc. ACL*, 2022, pp. 1–9.
[73] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao, "Adaptive budget allocation for parameter-efficient fine-tuning," in *Proc. Int. Conf. Learn. Represent.*, 2023.
[74] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 596–608.

Wenlve Zhou received the M.S. degree in Information and Communication Engi-neering from Wuyi University, Jiangmen, China. He is currently pursuing the Ph.D. degree in Information and Communication Engineering from South China University of Technology, Guangzhou, China. His research interests include: computer vision, especially transfer learning and representation learning.

Zhiheng Zhou received the B.S. and M.S. degrees in Applied Mathematics and the Ph.D. degree in Electronic and Information Engineering from South China University of Technology, Guangzhou, China, in 2000, 2002, and 2005, respectively. He is currently a Professor with South China University of Technology. His research interests include image processing and image and video transmission.

Unsupervised Domain Adaption Harnessing Vision-Language Pre-training —Appendix—

A. *Complement Hyper-Parameters Detailed in Different Benchmarks*

In this section, we will provide more comprehensive details about the experimental configurations and training tricks.

Training Iterations. We train 10,000 iterations for Office-Home [1], Office-31 [2] with ImageCLEF-DA [3] datasets; 20,000 iterations for Digits (including SVHN [4], MNIST [5], and USPS [6]) with VISDA-2017 [7] benchmark; and 40,000 iterations for DomainNet [8] dataset.

Data Augmentation. In our experiments, we adhered to the widely used settings of the relevant UDA methods and employed specific data augmentation designs for different benchmarks. For clarity and convenience, we describe the data augmentation operators using the PyTorch style.

****

Office-Home, Office-31, ImageCLEF-DA and DomainNet

Testing:

Resize (224) - ToTensor () - Normalize (mean,std)

Training:

Resize (256) - RandomCrop (224) - RandomHorizontalFlip() - ToTensor () - Normalize (mean,std)

VisDA-2017

Testing:

Resize (224) - CenterCrop (224) - ToTensor () - Normalize (mean,std)

Training:

Resize (224) - RandomHorizontalFlip () - CenterCrop (224) - ToTensor () - Normalize (mean,std)

Digits

Testing:

Lambda (lambda x: x.convert("RGB")) - Resize (224) - ToTensor () - Normalize (mean,std)

Training:

SVHN→MNIST

SVHN

Lambda (lambda x: x.convert("RGB")) - RandomResizedCrop (size=224, scale=(0.75, 1.2)) - ToTensor () - Normalize (mean,std)

MNIST

Lambda (lambda x: x.convert("RGB")) - RandomResizedCrop (size=224, scale=(0.75, 1.2)) - ToTensor () - Normalize (mean,std)

USPS→MNIST

USPS

Lambda (lambda x: x.convert("RGB")) - Resize (28) -pad (4) - RandomCrop(28) - RandomRotation(10) - Resize(224) - ToTensor () - Normalize (mean,std)

MNIST

Lambda (lambda x: x.convert("RGB")) - RandomResizedCrop (size=224, scale=(0.75, 1.2)) - ToTensor () - Normalize (mean,std)

MNIST→USPS

MNIST

Lambda (lambda x: x.convert("RGB")) - RandomResizedCrop (size=224, scale=(0.75, 1.2)) - ToTensor () - Normalize (mean,std)

USPS

Lambda (lambda x: x.convert("RGB")) - RandomResizedCrop (size=224, scale=(0.75, 1.2)) - ToTensor () - Normalize (mean,std)

As we utilized CLIP [9] as pre-training model, we adopted the mean and standard deviation values provided in the original paper, respectively:

$$\text{mean} = (0.48145466, 0.4578275, 0.40821073)$$

$$\text{std} = (0.26862954, 0.26130258, 0.27577711)$$

****

Trade-Off in CMKD. In the majority of experiments, we set the values of λ_1 to 0.25, λ_2 to 0.1, and λ_1 to 0.025. However, in the case of Office-31, we set both λ_2 and λ_1 to 0. For the DomainNet dataset, which contains a large number of categories, requiring a lower weight for self-training, we set λ_1 to 0.025 and λ_3 to 0.0025.

Predefined Threshold τ in RST. The parameter τ in RST is directly linked to the average accuracy and DSP. Throughout our experiments, we discovered that the optimal value of τ varies with the backbone learning rate. Hence, in most

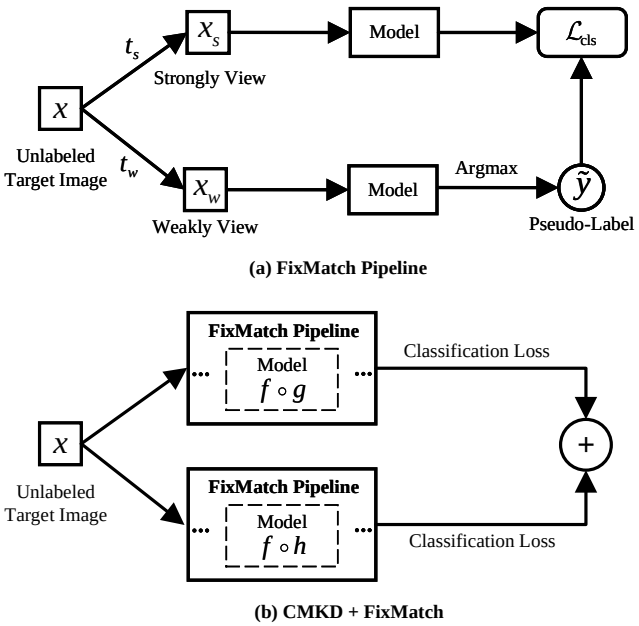


Fig. 1. FixMatch pipeline and implementation in our paper. t_s represents strong data augmentation, and t_w represents weak data augmentation. f serves as the visual encoder, g as the text encoder, and h as the task head.

experiments, we set τ to 1e-6. However, for the VisDA-2017 benchmark with ViT-B-16, τ is set to 1e-7, and for ResNet-50, τ is set to 1e-8.

B. FixMatch with CMKD

In our experiments, we observe that CMKD, when combined with FixMatch [10], demonstrates more effective performance. We provide a detailed implementations and configurations of FixMatch [10] in our method. FixMatch [10] is a semi-supervised technique that applies strong and weak augmentations to unlabeled samples separately. Afterward, the model generates pseudo-labels for the weakly augmented samples with higher confidence to guide the training of the strongly augmented samples. Since CMKD requires using the teacher's perspective to guide the student's visual learning, when combined with FixMatch, the training process needs to be performed separately based on the teacher's and student's viewpoints, as illustrated in Figure 1. The objective function of FixMatch can be defined as:

$$\lambda_{fm} \cdot \mathbb{1}_{\max(p_w)>\tau_{fm}} \arg\max\left(p_w\right) \cdot \log\left(p_s\right) \tag{1}$$

where λ_{fm} represents the trade-off, $\mathbb{1}$ is the indicator function, τ_{fm} represents the predefined threshold, p_w denotes the prediction of weakly view, and p_s denotes the prediction of strongly view. In most benchmark, λ_{fm} is set to 0.5 and τ_{fm} to 0.95, whereas on the VisDA-2017 dataset, λ_{fm} is set to 2.0 and τ_{fm} to 0.80, and on the DomainNet dataset, λ_{fm} is set to 0.1 and τ_{fm} to 0.95.

For weak augmentation, we employ the data augmentation methods referenced in Section A that are specific to the corresponding benchmarks. For strong augmentation, our specific configuration is as follows:

****

Lambda (lambda x: x.convert("RGB")) - Resize (224) - CenterCrop(224) - RandomHorizontalFlip() - ColorJitter (brightness = 0.4, contrast = 0.4, saturation = 0.4, hue = 0.0) - RandAugment('rand-n2-m10-mstd0.5') - ToTensor () - Normalize (mean, std)

"RandAugment" [11] is a method of automated data augmentation, and 'rand-n2-m10-mstd0.5' is the specific hyperparameter setting.

****

C. Relevant Model Parameters

In this paper, we consider the number of parameters necessary for deploying downstream tasks for the first time. Therefore, in this subsection, we provide a separate presentation of the parameter counts for both the backbone $f(\cdot)$ and task-head $h(\cdot)$ of the model. This will facilitate the computation of the DSP, as demonstrated in Table I.

TABLE I
NUMBER OF PARAMETERS FOR DIFFERENT MODELS.

Model	Pre-trained	Backbone	Task-Head
ResNet-50 [12]	ImageNet [13]	23508032 (23.5M)	2048*c
	CLIP [14]	38316896 (38.3M)	1024*c
ResNet-101 [12]	ImageNet [13]	42500160 (42.5M)	2048*c
	CLIP [14]	56259936 (56.3M)	512*c
ViT-B [15]	ImageNet [16]	85798656 (85.8M)	768*c
	CLIP [14]	86192640 (86.2M)	512*c

The design of the task head varies across different papers. Due to the relatively low number of parameters associated with the task head, for the sake of convenience, we assume that all task head are computed as 1-layer fully connected networks. Furthermore, when it comes to the same Backbone, there are slight distinctions between the ImageNet pre-training model and the CLIP pre-training model.

D. Classifier Configuration

Our classifier is intentionally designed to be lightweight while maintaining strong performance in comparison to existing classifiers. Taking inspiration from CDTrans [17], we've structured our classifier network as "BatchNorm(dim)-LayerNorm(dim)-FFN(dim, c)", where "BatchNorm" refers to Batch Normalization [18], "LayerNorm" refers to Layer Normalization [19], and "FFN" stands for Feed Forward Networks. Regarding the initialization of the classifier, we initialize the FFN using a normal distribution with a standard deviation set to 0.001. For Batch Normalization, we set the weight of the affine layer to 1.0 and the bias to 0.0 during initialization. The bias remains fixed throughout the training process.

E. CMKD Generalize to ImageNet Pre-trained Models

In this paper, our aim is to harness the capabilities of visual language pre-training models to tackle two significant challenges within unsupervised domain adaptation. Specifically,

TABLE II
COMPARISON WITH SOTA UDA METHODS ON VISDA-2017 USING IMAGENET PRE-TRAINED MODELS. ^{o} IMPLIES ITS PRE-TRAINED FROM ON IMAGENET-21K INSTEAD OF IMAGENET-1K. * IS PRE-TRAINED FROM CLIP. THE BEST PERFORMANCE IS MARKED AS BOLD.

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Avg.	DSP(M)
ResNet101:														
CDAN+E	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.9	42.52
DSAN	90.9	66.9	75.7	62.4	88.9	77.0	93.7	75.1	92.8	67.6	89.1	39.4	75.1	42.52
BNM	89.6	61.5	76.9	55.0	89.3	69.1	81.3	65.5	90.0	47.3	89.1	30.1	70.4	42.52
MSTN+DSBN	94.7	86.7	76.0	72.0	95.2	75.1	87.9	81.3	91.1	68.9	88.3	45.5	80.2	42.52
CGDM	92.8	85.1	76.3	64.5	91.0	93.2	81.3	79.3	92.4	83.0	85.6	44.8	80.8	42.52
SHOT	95.5	87.5	80.1	54.5	93.6	94.2	80.2	80.9	90.0	**89.9**	87.1	58.4	82.7	42.52
CDAN+MCC	94.5	80.8	78.4	65.3	90.6	79.4	87.5	82.2	**94.7**	81.0	86.0	44.6	80.4	42.52
DAPL*	**97.8**	83.1	**88.8**	77.9	**97.4**	91.5	**94.2**	79.7	88.6	89.3	92.5	**62.0**	**86.9**	42.52
Baseline	90.6	64.1	51.3	**83.7**	66.2	16.5	80.6	20.9	66.9	31.1	**94.7**	5.7	56.0	42.52
CMKD	95.6	88.0	73.2	67.9	92.3	92.4	80.0	82.3	93.3	82.9	87.7	50.8	82.2	42.52
CMKD+RST	94.3	83.8	72.7	70.8	91.8	91.9	85.5	**83.4**	91.0	82.1	89.3	41.5	81.5	**0.52**
CMKD+FixMatch	96.4	**88.5**	82.5	75.6	96.0	**95.7**	90.2	79.9	94.3	89.4	87.9	46.1	85.2	42.52
ViT-B-16:														
CDTrans	97.1	90.5	82.4	77.5	96.6	96.1	93.6	**88.6**	**97.9**	86.9	90.3	62.8	88.4	85.81
TVT^o	92.9	85.6	77.5	60.5	93.6	98.2	89.4	76.4	93.6	92.0	91.7	55.7	83.9	85.81
SDAT^o	98.4	90.9	85.4	82.1	98.5	97.6	96.3	86.1	96.2	96.7	92.9	56.8	89.8	85.81
SSRT-B^o	98.9	87.6	**89.1**	**84.7**	98.3	**98.7**	96.2	81.0	94.8	97.9	94.5	43.1	88.7	85.81
PMTrans^o	98.9	93.7	84.5	73.3	**99.0**	98.0	96.2	67.8	94.2	**98.4**	96.6	49.0	87.5	85.81
Baseline^o	**99.1**	73.6	82.5	68.6	93.5	84.5	**96.9**	13.4	70.2	96.5	**97.5**	21.6	74.8	85.81
CMKD ^{o}	98.8	92.9	81.7	74.6	96.9	95.5	91.4	72.1	93.5	96.4	95.5	54.7	87.0	85.81
CMKD+RST^o	98.7	93.4	81.7	72.3	97.3	96.9	92.7	69.9	89.5	97.5	96.4	57.2	87.0	**0.62**
CMKD+FixMatch^o	98.8	**96.3**	85.6	78.0	**99.0**	97.4	91.5	75.3	96.7	98.0	96.2	**70.2**	**90.3**	85.81

CMKD employs CLIP's zero-shot inference as a teacher's perspective, guiding the model in learning from unlabeled data, and achieving state-of-the-art results across multiple benchmarks. On the other hand, RST effectively alleviates the deployment burden in UDA by training highly sparse weights for the sub-tasks. Nevertheless, despite numerous experiments that validate the efficacy of the proposed methods, their performance under a conventionally pre-trained ImageNet model remains uncertain. To delve deeper into the effectiveness and adaptability of our approach, this subsection is dedicated to implementing our method based on the ImageNet pre-trained model.

We first begin by reviewing the CMKD paradigm introduced in this paper, which can be described as follows:

$$\mathcal{L}_{\text{cmkd}} = \lambda_1 \cdot (\alpha \cdot \mathcal{L}_{\text{task}} + (1-a) \cdot \mathcal{L}_{\text{distill}}) + \mathcal{L}_{\text{reg}} \tag{2}$$

$$\mathcal{L}_{\text{task}} = \text{GI}(p_h^t) = 1 - \sum_{i=1}^{c} [p_h^t(y=i|x^t)]^2 \tag{3}$$

$$\mathcal{L}_{\text{distill}} = \text{GI}\left(p_m^t\right) = 1 - \sum_{i=1}^{c} \left[p_m^t\left(y=i \mid x^t\right)\right]^2 \tag{4}$$

$$p_m^t = 0.5 \cdot \left(p_h^t + sg\left(p_{\text{g}}^t\right)\right) \tag{5}$$

$$\mathcal{L}_{\text{reg}} = \lambda_2 \cdot \text{KL}\left(y||p_{\text{g}}^s\right) + \lambda_3 \cdot \text{GI}\left(p_{\text{g}}^t\right) \tag{6}$$

Based on the equation above, it becomes evident that the primary challenge when applying CMKD using ImageNet as a foundation is the absence of the teacher's perspective p_{g} provided by the pre-trained model. Taking inspiration from Co-tuning [20], we adopt a strategy wherein we learn the class cluster centers from the output of the ImageNet classifier, resulting in the acquisition of class cluster centers denoted as M. In particular, we partition the source domain dataset into a training set and a validation set following a 1:1 ratio. Subsequently, we derive the prototype M through the process of category relationship learning [20]. Subsequently, we compute the sample probabilities based on the discrepancy associated with each class cluster center, effectively using it as the teacher's viewpoint. This enables the realization of CMKD on the ImageNet pre-trained model. Thus, p_{g} can be written as:

$$p_{\text{g}}(y=i|x) = \frac{\exp\left(-\text{KL}\left(M_i \| l(f(x))\right)\right)}{\sum_{k=1}^{c} \exp\left(-\text{KL}\left(M_k \| l(f(x))\right)\right)} \tag{7}$$

where $l(\cdot)$ denotes ImageNet classifier.

Based on the ImageNet pre-trained model, we verify the effectiveness of CMKD with RST on VisDA benchmark. Regarding the hyperparameter configuration, we set λ_1 to 0.25, λ_2 to 0.0, λ_3 to 0.0 and τ to 1e-6. For normalization in the preprocessing step, the mean is set to (0.485, 0.456, 0.406) and std is set to (0.229, 0.224, 0.225). As evident from Table II, our approach can be effectively applied to ImageNet pre-trained models as well. When employed in conjunction with the ViT-B model, we achieved a result of **90.3%** when combined with FixMatch [10], surpassing the previous SoTA (SDAT) by an increment of **+0.5%**. Moreover, when integrated with RST, merely **0.62M** parameters are required to attain equivalent performance as CMKD, thereby significantly reducing the parameter count necessary for downstream task deployment. Regarding ResNet-50, even though it trails behind the current state-of-the-art (DAPL) by **-1.7%**, it still demonstrates remarkable performance when contrasted with ImageNet-based pre-trained modeling approaches. It outperforms SHOT, CGDM,

TABLE III
UDA METHODS PERFORMED WITH IMAGE ENCODER OF VLP ON VISDA-2017

Method	Origin Hyperparameters	Ours Hyperparameters
ResNet101:		
BNM⋆	61.1 (-9.3)	68.5 (-1.9)
CGDM⋆	73.1 (-7.3)	80.9 (+0.1)
SHOT⋆	72.3 (-10.4)	81.6 (-1.1)
CMKD⋆ (Ours)	-	**87.0**
ViT-B-16		
CDTrans⋆	76.4 (-12.0)	86.5 (-1.9)
SSRT⋆	77.1 (-11.6)	87.0 (-1.7)
PMTrans⋆	77.8 (-9.7)	85.0 (-2.5)
CMKD⋆ (Ours)	-	**90.3**

and CDAN+MCC by margins of **+2.5%**, **+4.4%**, and **+4.8%**, respectively. Furthermore, with the application of RST, a mere **0.52M** parameters are required to attain performance comparable to CMKD, resulting in a negligible drop of just **-0.7%**.

In the experiments, the CMKD paradigm showcased its ability to generalize to UDA tasks, offering promising outcomes. Conversely, RST yielded unexpected results. Initially, we held the belief that RST could be applied to downstream tasks by acquiring knowledge from ultra-sparse weights due to the extensive pre-training of visual language pre-training models. However, when put into practice using the ImageNet pre-trained model, RST also led to a notable reduction in the DSP. This prompts us to eagerly anticipate the future trajectory of RST and its potential advancements. We're hopeful that RST's paradigm can be further expanded to encompass additional tasks. Its ability to demonstrate effectiveness across diverse domains such as object detection, semantic segmentation, etc., solely utilizing the ImageNet pre-trained model, is a tantalizing prospect.

F. *Other UDA methods using VLP pre-trained models*

In prior Unsupervised Domain Adaptation tasks, the conventional practice involves utilizing ImageNet-1k or ImageNet-21k as initial weights for training. However, this paper introduces the VLP model into UDA training. To ensure a fair comparison, we train based on CLIP's Image Encoder and commonly used UDA methods to validate the effectiveness of CMKD. Throughout the experiment, we maintain two sets of hyperparameters, where "Origin Hyperparameter" represents the original paper's parameters. Notably, due to CLIP's sensitivity to hyperparameters, we additionally employ the "Ours Hyperparameter" explored in this paper (refer to Section IV in the main text for detailed insights).

From Table III, it's evident that CMKD effectively leverages both the visual and text encoders of the CLIP model, substantially improving UDA performance and achieving notable results. However, when applying other methods to CLIP's visual encoder, using both hyperparameter sets 1 and 2, we observed a performance decrease compared to models trained on ImageNet pre-training weights. Our analysis suggests two potential reasons: firstly, it could be attributed to the challenge of finding optimal hyperparameters; secondly, traditional UDA methods focus on aligning feature spaces between the source and target domains, potentially disrupting the distribution of the pre-trained model's feature space. Consequently, there remains significant room for exploration in leveraging VLP models to further enhance UDA tasks.

TABLE IV
UDA METHODS COMBINED WITH RST ON OFFICE-HOME

Method	Avg.	DSP (M)
ResNet101:		
BNM	68.2 (+0.3)	2.11 (-99.25%)
DSAN	67.1 (-0.5)	1.98 (-99.3%)
SHOT	71.3 (-0.5)	2.83 (-99.0%)
ViT-B-16		
CDTrans^o	80.1 (-0.4)	10.3 (-99.0%)
SSRT^o	84.7 (-0.8)	9.27 (-99.1%)
PMTrans^o	87.3 (-1.4)	6.18 (-99.4%)

TABLE V
DIFFERENT DISTILLATION TERMS ON OFFICE-HOME USING RESNET-50.

Distillation term	Avg.
$\mathrm{KL}(sg(p_{\mathrm{g}}^t)\|\|p_h^t)$	76.5
$\mathrm{GI}\left(p_m^t\right)$	**79.3**

G. *Other UDA methods using RST*

RST offers schemes aimed at significantly diminishing the model storage overhead. In this subsection, we validate the compatibility of RST with other UDA methods. The predefined thresholds τ associated with RST are intricately linked to the learning rate, thus, we set tau to 1e-5 in this section. To visually highlight the parameter reduction differences, we'll opt for the office-home benchmark, which encompasses a broader range of tasks for our experiments. From Table IV, it's evident that despite varying degrees of model performance reduction post-RST implementation, there's a notable decrease in DSP. This reduction effectively alleviates the storage strain on the model.

H. *Further Hyperparameters Discussion*

In this section, our primary focus was on examining CMKD's distillation term and the associated trade-off configurations. We contend in the paper that the teacher model (CLIP, in this case) doesn't consistently outperform, and the alignment of student and teacher distributions via KL divergence might negatively impact model performance. This is validated in the experimental results showcased in Table V, where the performance experiences a considerable drop compared to $\mathrm{GI}\left(p_m^t\right)$.

Furthermore, various hyperparameter configurations in CMKD are deliberated upon. Table VI illustrates that the most optimal performance is attained at the settings of (0.25, 0.1, 0.025).

I. *Convergence and Stability of Empirical Results*

To further substantiate the training stability and convergence speed of our approach, we measure the accuracy of their training processes using the ResNet-50 and ViT-B-16 architectures

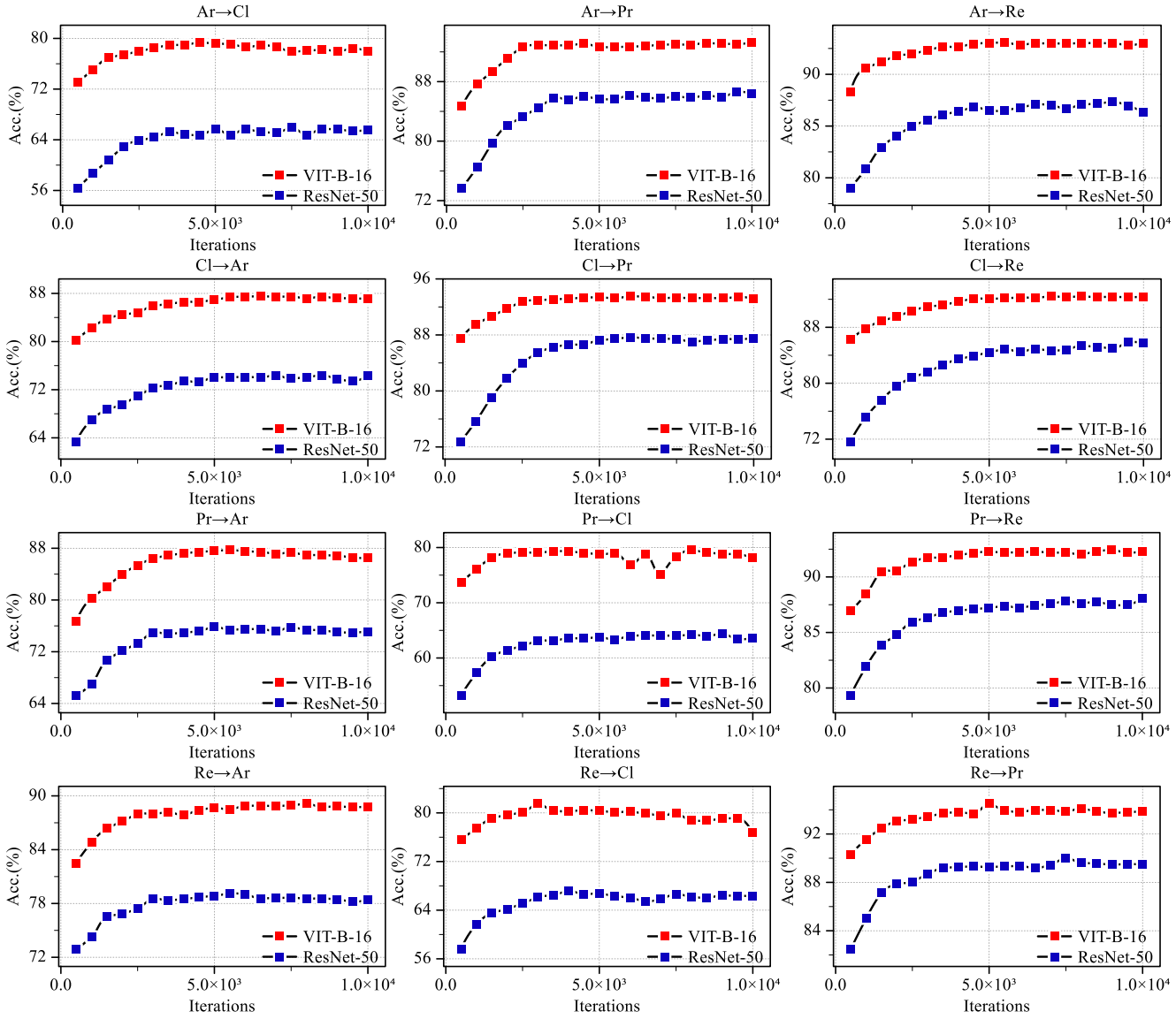


Fig. 2. Validation Accuracy across iterations on different splits of Office-Home.

TABLE VI
DIFFERENT TRADE-OFF OF CMKD ON OFFICE-HOME USING RESNET-50.

$(\lambda_1, \lambda_2, \lambda_3)$	Avg.
(0.05, 0.02, 0.005)	78.1
(0.25, 0.1, 0.025)	**79.3**
(0.50, 0.2, 0.05)	77.3

on the Office-Home Benchmark. As depicted in Figure 2, the training process for all 12 subtasks displays remarkable smoothness, and the validation results throughout the training exhibit minimal fluctuations. This consistency suggests a high level of training stability inherent in our method. Additionally, the curves within the graph demonstrate the impressive convergence capability of our approach, characterized by swift convergence followed by a sustained smooth phase during the initial training stages.

J. RST with ImageNet Pre-trained Model on Segmentation

Unsupervised Domain Adaptive Semantic Segmentation (UDASS) presents a common challenge in unsupervised domain adaptation, alongside classification tasks. This task mirrors the setup of classification tasks within UDA, where the model is trained on labeled source domain data to facilitate object segmentation in the unlabeled target domain. In order to further demonstrate the versatility of RST, we opted to compare it with various PEFT methods, notably BitFit [21], LoRA [22], etc., on the GTA [23]→Cityscape [24] benchmark, which is a standard task in UDASS. Given that in the UDASS task, the pre-training weights are derived from the ImageNet classification task, and these pre-training features may not adequately align with the representations required for downstream dense detection tasks. Therefore, by setting approximately 10% of the trainable parameters and setting the rank of LoRA to 16, we aimed to address this mismatch.

TABLE VII
COMPARISON WITH SOTA PEFT METHODS ON GTA→CITYSCAPE.

Method	Road	S.walk	Build.	Wall	Fence	Pole	Tr.Light	Sign	Veget.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.bike	Bike	mIOU	DSP(M)
Fine-Tuning	**95.7**	**70.2**	87.8	36.2	35.2	**38.3**	50.2	53.8	88.1	45.5	**88.3**	**69.6**	**43.7**	87.7	49.4	50.2	0.0	24.9	48.3	56.0	42.51
Linear-Probe	76.6	11.1	60.7	1.67	0.0	15.3	0.9	0.0	59.6	6.0	59.9	0.8	2.4	43.9	1.3	1.4	0.1	4.6	3.8	18.4	**0.70**
BitFit	83.5	8.2	75.9	11.8	1.5	21.4	8.6	3.2	75.7	14.9	79.0	29.8	10.8	72.4	18.8	5.7	0.1	10.8	4.0	28.3	**0.75**
LoRA	85.6	34.5	80.8	24.8	20.7	25.6	26.6	38.8	83.6	35.1	81.3	58.1	28.2	75.5	31.8	37.9	23.4	**26.1**	45.8	45.5	5.39
$\text{RST}_{config1}$	93.8	58.9	86.8	32.0	32.6	35.8	42.6	49.2	87.4	41.8	87.5	65.1	33.1	87.0	49.5	49.0	0.7	25.7	**50.8**	53.1	5.88
$\text{RST}_{config2}$	94.0	57.3	**88.4**	**39.2**	**36.6**	37.5	49.4	54.3	**88.4**	**47.5**	88.1	69.4	43.2	**88.7**	**58.2**	**52.8**	**13.7**	24.6	45.2	**56.7**	12.6

For network architecture, we utilize the DeepLab-V2 [25] with ResNet101 [12] as the backbone. As a standard practice, we adhere to the self-training approach outlined in DAFormer [26] along with its associated training parameters. Specifically, we employ AdamW optimization with a learning rate of 6×10^{-5} for the encoder and 6×10^{-4} for the decoder, utilizing a batch size of 2. Additionally, we incorporate linear learning rate warmup, and maintain a momentum parameter 0.999. For RST, we have introduced a slight modification. Instead of using a stable predefined threshold, we now use a value that changes over time. We set up two sets of configurations, with the threshold for *config1* ranging from 5×10^{-4} to 3.5×10^{-2} as the number of training rounds increases, and the threshold for *config2* ranging from 5×10^{-4} to 2×10^{-2}. The trainable parameters for *config1* are approximately 10%, mainly for fair comparison with LoRA; *Config2*'s trainable parameters are about 20%, mainly for comparison with Fine-Tuning. This dynamic adjustment mechanism mirrors the approach outlined in the implementation details provided in the manuscript. The rationale behind this adjustment lies in the divergence between the pre-training task and the downstream task. Given the significant differences between the two, we believe that involving more parameters in the initial stages of training allows for a more comprehensive assessment of which parameters should be retained and which ones should be reset back to the pre-training weights throughout the training process. This adaptive approach ensures better alignment between the model's learned representations and the requirements of the downstream task.

Table VII presents intriguing experimental findings:

(1) In the context of Unsupervised Domain Adaptive Segmentation, utilizing ImageNet classification pre-training models leads to a significant gap between the pre-training task and the downstream task. Consequently, methodologies like Linear-Probe and BitFit [21], despite training very few parameters, exhibit a substantial performance degradation compared to Fine-Tuning. However, approaches like LoRA and RST can adjust the trainable parameters based on task difficulty, resulting in superior performance.

(2) Under Configuration 1, both RST and LoRA train approximately 10% of the parameters. Remarkably, RST surpasses LoRA by +7.6% mIOU, with only a slight difference of -2.8% mIOU compared to Fine-Tuning. This highlights two key points: firstly, despite a significant feature gap between the pre-trained and downstream tasks, a substantial number of parameters in the model remain beneficial for the downstream task. Secondly, in the scenario of smaller models, RST outperforms LoRA [22] with the same number of trainable parameters, showcasing the versatility of RST across tasks like sense prediction.

(3) An intriguing experiment involves increasing the number of trainable parameters based on Config2, resulting in the model training 20% of the parameters compared to full parameter fine-tuning. Surprisingly, RST outperforms Fine-Tuning by +0.7 mIOU despite a greater number of trainable parameters. This suggests that RST not only excels in reducing model deployment complexity but also holds potential as a novel regularization technique capable of enhancing downstream task performance.

K. *Partial-set Domain Adaptation*

In real-world scenarios, it is common for the categories in the target domain to be a subset of the source domain, which can be regarded as a special case of the traditional UDA, called Partial-set Domain Adaptation (PDA) [27]. To further validate the validity and flexibility of our method, we try to apply CMKD to the PDA task while only making simple modifications to the original method. When utilizing CMKD and RST for PDA, the key aspect is to estimate the categories for the target dataset. This prevents the model from adapting to classes that are not part of the target domain, thereby avoiding potential performance degradation. Inspired by SHOT [28], we count the number of predicted categories and estimate the categories on the target domain using a thresholding method. The counting of category i can be formulated as follows:

$$n_i = \sum_{j=1}^{n_t} \mathbb{1}_{\left[\text{argmax}\left(p_{hj}^t(y|x_j^t)\right)=i\right]} \tag{8}$$

Next, PDA can be implemented with CMKD by making a simple modification to the predicted probability of the samples x_j^t, represented as follows:

$$\hat{p}_{hj}^t(y=c|x_j^t) = \mathbb{1}_{[n_c \geq T_{pda}]} p_{hj}^t(y=c|x_j^t) \tag{9}$$

where $\hat{p}_{hj}^t(y=c|x_j^t)$ represents the modified probability of category c, $p_{hj}^t(y=c|x_j^t)$ denotes the original probability of category c, n_c signifies the counting of categories c, and T_{pda} is a predefined threshold. The PDA task can be realized by employing CMKD with the modified probability instead of the original probability. In the PDA task, the weight of the classification loss is set to one-half of that in the UDA task, while the transfer loss is reduced to one-tenth of the UDA task's weight. T_{pda} is selected to be 14.

For the experiment settings, we adopt the protocol outlined in [29] for the Office-Home [1] dataset. Specifically, the target

TABLE VIII
COMPARISON WITH SOTA METHODS ON OFFICE-HOME (65→25) FOR PARTIAL-SET UDA.

Method	Ar→Cl	Ar→Pr	Ar→Re	Cl→Ar	Cl→Pr	Cl→Re	Pr→Ar	Pr→Cl	Pr→Re	Re→Ar	Re→Cl	Re→Pr	Avg.	DSP(M)
ResNet50:														
DRCN	54.0	76.4	83.0	62.1	64.5	71.0	70.8	49.8	80.5	77.5	59.1	79.9	69.0	283.60
BA^{3}US	60.6	83.2	88.4	71.8	72.8	83.4	75.5	61.6	86.5	79.3	62.8	86.1	76.0	283.60
TSCDA	63.6	82.5	89.6	73.7	73.9	81.4	75.4	61.6	87.9	83.6	67.2	88.8	77.4	283.60
SHOT	64.6	85.1	92.9	78.4	76.8	86.9	79.0	65.7	89.0	81.1	67.7	86.4	79.5	283.60
SHOT++	65.0	85.8	**93.4**	78.8	77.4	**87.3**	79.3	66.0	89.6	81.3	68.1	86.8	79.9	283.60
CLIP*	63.2	84.3	86.9	76.4	84.3	86.9	76.4	63.2	86.9	76.4	63.2	84.3	77.7	**0.00**
Baseline*	58.4	75.2	82.7	61.9	67.7	72.5	66.0	56.0	80.5	75.4	60.0	81.5	69.8	460.4
CMKD*	72.4	87.5	90.9	78.1	80.0	85.4	79.2	70.3	89.8	87.6	71.9	89.6	81.9	460.4
CMKD+RST*	68.3	**89.9**	91.4	78.0	**81.5**	85.3	77.8	70.0	**91.5**	85.7	70.4	89.5	81.6	**0.98**
CMKD+FixMatch*	**73.1**	87.7	91.8	**80.3**	79.8	86.1	**80.0**	**74.4**	91.1	**88.7**	**74.9**	**89.9**	**83.1**	460.40
ViT-B-16:														
CLIP*	79.5	89.3	91.3	84.3	89.3	91.3	84.3	79.5	91.3	84.3	79.5	89.3	86.1	**0.00**
Baseline*	78.1	84.1	89.8	79.3	82.6	84.1	77.1	77.8	87.5	81.5	79.2	87.8	82.4	1034.80
CMKD*	83.4	89.6	**95.4**	86.5	91.2	91.5	**89.4**	89.9	94.4	90.9	87.3	93.3	90.2	1034.80
CMKD+RST*	82.3	89.1	94.6	85.5	91.0	91.3	85.1	89.4	94.4	89.0	86.0	**93.6**	89.3	**0.88**
CMKD+FixMatch*	**85.0**	**90.1**	**95.4**	**87.1**	**93.4**	**91.8**	86.1	**91.2**	**95.2**	**91.0**	**90.3**	93.5	**90.8**	1034.80

TABLE IX
USING MASK2FORMER FOR UDASS ON CITYSCAPE→ACDC.

Method	Road	S.walk	Build.	Wall	Fence	Pole	Tr.Light	Sign	Veget.	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	M.bike	Bike	mIOU	DSP(M)
Deeplabv2:																					
Deeplabv2	71.9	26.2	51.1	18.8	22.5	19.7	33.0	27.7	67.9	28.6	44.2	43.1	22.1	71.2	29.8	33.3	48.4	26.2	35.8	38.0	42.51
DACS	58.5	**34.7**	**76.4**	20.9	22.6	**31.7**	32.7	**46.8**	58.7	**39.0**	36.3	43.7	20.5	72.3	39.6	34.8	51.1	24.6	38.2	41.2	42.51
FDA	**73.2**	**34.7**	59.0	**24.8**	**29.5**	28.6	**43.3**	44.9	**70.1**	28.2	**54.7**	**47.0**	**28.5**	**74.6**	**44.8**	**52.3**	**63.3**	**28.3**	**39.5**	**45.7**	42.51
Mask2Former:																					
Mask2Former	85.4	55.7	52.4	23.3	23.0	41.2	68.4	52.6	71.8	30.2	68.9	48.1	21.4	74.5	30.7	19.4	14.5	24.4	23.2	43.6	44.0
CMKD	**86.2**	**62.0**	**77.3**	**26.9**	**29.4**	**45.3**	**70.6**	**56.4**	**76.1**	**31.6**	**74.4**	**50.6**	20.0	**78.0**	36.4	**35.5**	**32.3**	**38.8**	40.4	**51.0**	44.0
CMKD+RST	**86.2**	61.4	77.0	24.9	27.6	43.5	69.0	55.6	75.5	30.3	74.0	48.6	**25.1**	77.2	**39.1**	32.4	29.2	35.2	**45.5**	50.4	**5.52**

domain in Office-Home [1] consists of a total of 65 classes, and we focus on 25 classes (the first 25 in alphabetical order) for our analysis. CMKD is compared with the SoTA method for PDA, respectively: DRCN [29], BA^{3}US [30], TSCDA [31], SHOT [28], SHOT++ [32]. As observed from Table VIII, it is evident that CMKD and RST demonstrates exceptional performance in the PDA task, achieving state-of-the-art results. Moreover, this highlights the versatility of CMKD and RST in the PDA task, as it remains effective across different architectures.

L. *Generalized to Universal Image Segmentation Model*

In this section, we opt for the Universal Image Segmentation Model to delve into the Unsupervised Domain Adaptive Segmentation (UDASS) task, specifically utilizing the Mask2Former [33] architecture. Typically, in UDASS experiments, the Deeplabv2 [25] model with ResNet101 [12] backbone is commonly employed. However, as Mask2Former [33] contains more parameters in the decoder, to ensure a fair comparison with similar parameter counts, we select the Mask2Former model with ResNet50 backbone for our experiments. Considering that Mask2Former has been supervised trained on the Cityscape [24] dataset in the upstream task, we opt for the Cityscape [24]→ACDC [34] benchmark for exploration in this subsection, preventing data leakage.

In the classification task, CMKD was primarily designed based on the technique of Gini impurity. However, in the segmentation task, methods related to entropy optimization are not directly applicable to UDASS. Therefore, we need to adapt CMKD to suit the segmentation task. We draw inspiration from the mean-teacher [35] approach, which generates pseudo-labels to guide the training of student models using an Exponential Moving Average (EMA) teacher model. As a result, $\mathcal{L}_{\text{task}}$ can be redefined as:

$$\mathcal{L}_{task} = \mathcal{L}_{m2f}(x^t, \tilde{y}^t; F) \tag{10}$$

$$\tilde{y}^t_{ij} = \underset{c}{argmax}\ \overline{F}(x^t)_{ijc} \tag{11}$$

$$\overline{F} \leftarrow m\overline{F} + (1-m)F_T \tag{12}$$

where $F(\cdot) = E(\cdot) \circ D_h(\cdot)$. E represents the backbone network, while D_h encompasses the Mask2Former Head, comprising the Transformer Decoder and Pixel Decoder. D_h is the functional equivalent of the task-head in the classification task $\mathcal{L}_{m2f}$ denotes the loss function associated with the Mask2Former, for which no modifications were made to the network structure or the loss function utilized in the original paper. $\overline{F}$ represents the teacher model generated through Exponential Moving Average (EMA) and F_T is the model of training step T.

For the distillation term $\mathcal{L}_{distill}$, we adhere to the concept of CMKD and generate pseudo-labels by averaging the predictions of the pre-trained head $D_g(\cdot)$ and the predictions of the EMA model. During training, the pre-training head does not update the weights. These pseudo-labels are then employed to guide the training of the model.

$$\mathcal{L}_{distill} = \mathcal{L}_{m2f}(x^t, \check{y}^t; F) \tag{13}$$

$$\check{y}^t_{ij} = \underset{c}{argmax}\ 0.5 \cdot (\overline{F}(x^t)_{ijc} + \check{F}(x^t)_{ijc}) \tag{14}$$

where $\check{F}(\cdot) = E(\cdot) \circ D_g(\cdot)$. In the segmentation task, the constraint term is removed, and the modified CMKD formulation is denoted as:

$$\mathcal{L}_{\text{cmkd}} = \alpha \cdot \mathcal{L}_{\text{task}} + (1 - a) \cdot \mathcal{L}_{\text{distill}} \tag{15}$$

$$\alpha = sg(\exp(-\text{KL}(\overline{F}(x^t) \parallel \check{F}(x^t)))) \tag{16}$$

Finally, the total loss is written as:

$$\mathcal{L}_{\text{total}} = \lambda_{sup} \cdot \mathcal{L}_{\text{sup}} + \lambda_{cmkd} \cdot \mathcal{L}_{\text{cmkd}} \tag{17}$$

$$\mathcal{L}_{\text{sup}} = \mathcal{L}_{m2f}(x^s, y^s; F) \tag{18}$$

$\mathcal{L}_{\text{sup}}$ is training loss on source domain. We experimented with MMSegmentation[1]. For the choice of hyperparameters, it remains basically the same as Mask2Former, except that the initial learning rate we set to 1×10^{-5}. λ_{sup} is set to 0.1, while λ_{cmkd} uses the dynamic mechanism in the Implementation details in the manuscript, β is set to 0.25. Image size cropped to 512×512 and the momentum m factor is set to 0.99. We adhered to the DACS [36] data augmentation approach and utilized RCS [26]. For RST, similar to Section J, we utilize a dynamic mechanism to set the threshold. Specifically, the threshold gradually increases from 2×10^{-6} to 1×10^{-4} as the number of training rounds progresses. We compare with methods such as Deeplabv2 [25], DACS [36], and FDA [37]. In contrast to previous transfer learning tasks where the Backbone is typically retained and a completely new randomly initialized task head is created, we directly initialize the task head using pre-trained weights in our approach. This decision is motivated by the fact that the decoder component of Mask2Former contains a substantial number of weights, accounting for nearly half of the model's total weights, and can provide an efficient initialization for the model. Additionally, previous discussions regarding PEFT have focused on its application to the backbone, while the task head is preserved in its entirety. However, this approach results in significant parameter storage requirements in Mask2Former due to the large number of task head parameters. Therefore, we employ RST for both the backbone and decoder components in this part.

Table IX illustrates the generalization capabilities of CMKD and RST on Mask2Former, along with the robustness of Mask2Former to domain shifts compared to deeplabv2, both with similar parameter counts. This comparison underscores the superiority of the Mask2Former structure. Moreover, we made slight modifications to the implementation of CMKD based on Mask2Former, resulting in a notable improvement of +7.4 mIOU compared to Mask2Former alone. This enhancement underscores the potential of CMKD in image segmentation. Furthermore, when CMKD is combined with RST, only approximately 10% of the total number of parameters need to be preserved to achieve results comparable to full parameter fine-tuning. We anticipate that this compression ratio can be further reduced with larger pre-trained models and datasets.

TABLE X
COMPARISON WITH PEFT METHODS ON KITTI2015.

Method	EPE(↓)	DSP (M)
Fine-Tuning	**0.67**	4.1878
Linear-Probe	**0.67**	3.1210
BitFit	1.12	**0.0043**
LoRA	1.32	0.0163
RST	0.96	0.0181
RST$_{bias}$	**0.69**	**0.0043**

M. RST on Optical Flow

In this subsection, we focus on the task of optical flow, which involves estimating per-pixel motion between video frames. To evaluate the effectiveness of RST, we employ the RAFT [38] architecture. There exist several settings for the optical flow task. For instance, models can be randomly initialized and trained on FlyingChairs [39] and FlyingThings [40] datasets, and then directly tested on the Sintel dataset. Alternatively, after pre-training on the serval datasets, models can be fine-tuned on the down-stream dataset. Given that RST is a parameter-efficient training algorithm that relies on pre-training knowledge, we solely consider the second task setting. We adopt the setup employed in the MMFlow[2] implementation of RAFT [38], which involves pre-training on a mixed dataset followed by fine-tuning on KITTI2015 [41]. The mixed dataset consisted of FlyingChairs [39], FlyingThing3d [40], Sintel [42], KITTI2015 [41], and HD1K [43]. Additionally, we utilize the pre-training weights provided by MMFlow[2] for fine-tuning on downstream tasks. We opted to compare it with various PEFT methods, notably BitFit [21], LoRA [22]. As with Section L, PEFT is used for both encoder and decoder. The rank of LoRA is set to 1. During the reproduction process, we encountered a significant deviation in the results when using the profiles and pre-training weights provided by MMFlow. These results were notably distant from those reported in the original paper. Upon examination, it became apparent that the model was experiencing underfitting. To address this issue, we extended the training duration to 100,000 steps, leveraging the original profile as a reference.

For RST, a slight adjustment is necessary. Given the limited capacity of the model used for RAFT, it becomes challenging to determine the optimal thresholds for parameter filtering using predefined thresholds. As a workaround, we adopt a strategy wherein we select, the parameter with the top $r\%$ change, gradually decreasing the value of r from 100 to 0.2 over time. It's important to note that this parameter selection method is utilized only when the model possesses a very small parameter count. In scenarios where the parameter count is large, this selection method may strain the CPU significantly,

[1]https://github.com/open-mmlab/mmsegmentation/blob/main/configs/mask2former/README.md

[2]https://github.com/open-mmlab/mmflow/blob/master/configs/raft/README.md

and employing the predefined threshold method would be a more efficient choice.

Table X reveals unexpected experimental findings. Despite RAFT undergoing extensive pre-training, techniques like BitFit and LoRA prove challenging to adapt effectively to downstream tasks. Although RST outperforms BitFit and LoRA, it still falls behind methods that train a larger number of parameters, such as Fine-Tuning. This phenomenon may stem from RAFT's relatively small model size, making it challenging for parameter-efficient fine-tuning methods to effectively adjust the model weights. Based on this observation, we analyzed the positions of the non-zero weights obtained by RST and found that many of them are Bias terms. Given this, one might wonder why BitFit cannot achieve similar results. Intuitively, we decided to utilize RST to exclusively train the Bias items while retaining all Bias items, denoted as RST$_{bias}$. Conceptually, RST$_{bias}$ aligns with BitFit. However, since optical flow tasks often involve gradient norm clipping techniques during training, BitFit's gradient constraints are derived solely from the Bias term, whereas RST$_{bias}$'s gradient constraints encompass all parameters. Surprisingly, this simple modification significantly enhances model performance, approaching results close to those achieved by Fine-Tuning approximation, and requires only fine-tuning of the Bias term. This underscores the considerable potential of related techniques such as parameter-efficient fine-tuning in optical flow tasks.

N. *Analysis of CMKD and RST*

In this section, we will analyze and discuss CMKD and RST, focusing primarily on elucidating the reasons behind their effectiveness and exploring their applicability in deployment scenarios.

Q1: How does CMKD reconcile the perspectives of the student and teacher models?

To address this question, we initially simplify the gradient formula for CMKD. Given that the primary components of CMKD are $\mathcal{L}_{task}$ and $\mathcal{L}_{distill}$, and for the sake of analytical convenience, we exclude $\mathcal{L}_{reg}$ from our discussion. Thus, we define $\mathcal{L}$ as the combination of $\mathcal{L}_{task}$ and $\mathcal{L}_{distill}$. The gradient formula can be expressed as follows:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta} =& \alpha \cdot \frac{\partial \mathcal{L}_{\text{task}}}{\partial \theta} + (1-\alpha) \cdot \frac{\partial \mathcal{L}_{\text{distill}}}{\partial \theta} \\ =& -\alpha \cdot \sum_{i=1}^{c} [2 \cdot p_h^t(y=i|x^t) \cdot \frac{\partial p_h^t(y=i|x^t)}{\partial \theta}] \\ & - (1-\alpha) \cdot \sum_{i=1}^{c} p_h^t\left(y=i \mid x^t\right) \cdot \frac{\partial p_h^t\left(y=i \mid x^t\right)}{\partial \theta} \\ & - (1-\alpha) \cdot \sum_{i=1}^{c} p_{\mathrm{g}}^t\left(y=i \mid x^t\right) \cdot \frac{\partial p_h^t\left(y=i \mid x^t\right)}{\partial \theta} \\ =& -\sum_{i=1}^{c} [(1+\alpha) \cdot p_h^t\left(y=i \mid x^t\right) \\ & + (1-\alpha) \cdot p_g^t\left(y=i \mid x^t\right)] \cdot \frac{\partial p_h^t\left(y=i \mid x^t\right)}{\partial \theta}\end{aligned} \tag{19}$$

For ease of discussion, we denote Eq. (10) as:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta} =& \alpha \cdot \frac{\partial \mathcal{L}_{\text{task}}}{\partial \theta} + (1-\alpha) \cdot \frac{\partial \mathcal{L}_{\text{distill}}}{\partial \theta} \\ =& -\sum_{i=1}^{c} [(1+\alpha) \cdot p_{h(i)}^t + (1-\alpha) \cdot p_{g(i)}^t] \cdot \frac{\partial p_{h(i)}^t}{\partial \theta}\end{aligned} \tag{20}$$

For detailed explanations regarding the calculation of $\mathcal{L}_{\text{task}}$, $\mathcal{L}_{\text{distill}}$ and α please refer to Section III.B of the manuscript. From equation (20), we observe that the scaling factor representing the student's perspective is $(1+\alpha) \cdot p_{h(i)}^t$, while the scaling factor representing the teacher's perspective is $(1-\alpha) \cdot p_{g(i)}^t$. Given that α takes values in the range [0, 1], training from the student's perspective will invariably dominate. This mechanism ensures that the model does not become excessively influenced by generic knowledge in instances where the teacher's perspective proves ineffective in providing adequate guidance.

Further, we can briefly explore the design philosophy of CMKD self-training from the perspective of whether the student and the teacher are aligned.

Alignment of teacher and student perspectives. In this scenario, when both the teacher perspective and the student perspective predictions are correct, the model undergoes further training in a positive direction. Conversely, when both predicted labels are incorrect, the dynamic design of λ_1 can effectively mitigate the issue. Essentially, the model utilizes smaller training weights during the early stage, gradually increasing the trade-off as the training progresses. This straightforward dynamic training mechanism proves effective in preventing the model from being misled by incorrect training signals too early in the process. For specific implementation details, please refer to Section IV.A Implementation Details, of the manuscript.

Misalignment of teacher and student perspectives. When the student perspective predicts incorrectly, there are two potential scenarios for the teacher perspective: it may predict correctly or incorrectly. However, due to the discrepancy in the distributions of the two predictions, the value of α may become very small, possibly nearing 0 at its minimum. This scaling factor $(1+\alpha) \cdot p_{h(i)}^t + (1-\alpha) \cdot p_{g(i)}^t$ effectively smooths the learning intensity for each category, thereby reducing the model's learning rate for such samples. This adjustment prevents the model from being further misled in cases where the student perspective is incorrect, aiding in maintaining model stability and performance. Indeed, it is plausible to encounter scenarios where the student's perspective is correct while the teacher's perspective is incorrect. However, based on our previous analysis, we understand that the student perspective consistently dominates, with the teacher perspective serving more as a regularizer. Consequently, the model will ultimately learn in the correct direction, guided primarily by the accurate predictions from the student's perspective.

Q2: Scenarios for the application of RST.

From the experiments conducted in the paper, it is evident that RST effectively reduces the number of weights required to be preserved for downstream task fine-tuning. Moreover, the larger the scale of upstream task training, the more RST can reduce the number of parameters, as extensive upstream training already covers features necessary for the downstream

task. Additionally, RST exhibits broad applicability beyond its original purpose of addressing the "small model and large number of tasks" challenge in UDA tasks. It demonstrates excellent generalization across various tasks and different pre-training weights.

For instance, taking CLIP as an example, we can utilize the original CLIP model as a solution for general problems. When faced with domain-specific problems, we can combine the sparse weights obtained from RST training with the pre-trained CLIP model, effectively transforming it into a specialized model for solving domain-specific knowledge. As the weights obtained from RST training are lightweight, we can store multiple specialized weights on deployment devices, enabling the system to solve problems in diverse domains while maintaining generality.

While multitasking techniques can enable a single model to tackle multiple tasks, conflicts between tasks may arise as the number of tasks increases. Moreover, adding a new task often necessitates retraining the model along with existing tasks to prevent forgetting previous task knowledge. Furthermore, accommodating multiple tasks can significantly increase model capacity. RST effectively mitigates these challenges by training individual tasks independently, thus eliminating the possibility of task conflicts and preserving the weights of previous tasks.

While structured reparameterization PEFT techniques like LoRA could provide similar advantages, experiments have shown that RST outperforms them, particularly for small-scale models and insufficient pre-training. However, it is important to acknowledge that RST has its limitations. RST does not possess features that mitigate explicit memory during model training. Nonetheless, in tasks where explicit memory is not a primary concern due to limited model capacity, RST proves highly effective. Looking ahead, further exploration of the RST technique could potentially contribute to fine-tuning large language models.

REFERENCES

[1] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5018–5027.

[2] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 213–226.

[3] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2208–2217.

[4] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011.

[5] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[6] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 16, no. 5, pp. 550–554, 1994.

[7] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "Visda: The visual domain adaptation challenge," *arXiv preprint arXiv:1710.06924*, 2017.

[8] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1406–1415.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn*, 2021, pp. 8748–8763.

[10] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 596–608.

[11] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 702–703.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[14] X. Dong, J. Bao, T. Zhang, D. Chen, S. Gu, W. Zhang, L. Yuan, D. Chen, F. Wen, and N. Yu, "Clip itself is a strong fine-tuner: Achieving 85.7% and 88.0% top-1 accuracy with vit-b and vit-l on imagenet," *arXiv preprint arXiv:2212.06138*, 2022.

[15] 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," in *Proc. Int. Conf. Learn. Represent.*, 2021.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015.

[17] T. Xu, W. Chen, P. Wang, F. Wang, H. Li, and R. Jin, "Cdtrans: Cross-domain transformer for unsupervised domain adaptation," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[19] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[20] K. You, Z. Kou, M. Long, and J. Wang, "Co-tuning for transfer learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 17 236–17 246.

[21] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *Proc. ACL*, 2022, pp. 1–9.

[22] 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," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[23] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Eur. Conf. Comput. Vis.* Springer, 2016, pp. 102–118.

[24] 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 *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.

[25] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2017.

[26] L. Hoyer, D. Dai, and L. Van Gool, "Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9924–9935.

[27] Z. Cao, M. Long, J. Wang, and M. I. Jordan, "Partial transfer learning with selective adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2724–2732.

[28] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6028–6039.

[29] S. Li, C. H. Liu, Q. Lin, Q. Wen, L. Su, G. Huang, and Z. Ding, "Deep residual correction network for partial domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 7, pp. 2329–2344, 2020.

[30] J. Liang, Y. Wang, D. Hu, R. He, and J. Feng, "A balanced and uncertainty-aware approach for partial domain adaptation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 123–140.

[31] C.-X. Ren, P. Ge, P. Yang, and S. Yan, "Learning target-domain-specific classifier for partial domain adaptation," *IEEE Trans. Neural Net. Learn. Syst*, vol. 32, no. 5, pp. 1989–2001, 2020.

[32] J. Liang, D. Hu, Y. Wang, R. He, and J. Feng, "Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8602–8617, 2021.
[33] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1290–1299.
[34] C. Sakaridis, D. Dai, and L. Van Gool, "Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10 765–10 775.
[35] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
[36] W. Tranheden, V. Olsson, J. Pinto, and L. Svensson, "Dacs: Domain adaptation via cross-domain mixed sampling," in *Proc. WACV*, 2021, pp. 1379–1389.
[37] Y. Yang and S. Soatto, "Fda: Fourier domain adaptation for semantic segmentation," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4085–4095.
[38] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Eur. Conf. Comput. Vis.* Springer, 2020, pp. 402–419.
[39] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.
[40] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4040–4048.
[41] M. Menze, C. Heipke, and A. Geiger, "Object scene flow," *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018.
[42] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, ser. Part IV, LNCS 7577, A. Fitzgibbon et al. (Eds.), Ed. Springer-Verlag, Oct. 2012, pp. 611–625.
[43] D. Kondermann, R. Nair, K. Honauer, K. Krispin, J. Andrulis, A. Brock, B. Gussefeld, M. Rahimimoghaddam, S. Hofmann, C. Brenner *et al.*, "The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 19–28.