DINE: Domain Adaptation from Single and Multiple Black-box Predictors

Jian Liang [1] Dapeng Hu [2] Jiashi Feng [3] Ran He [1,4,5]
[1] CRIPAC & NLPR, CASIA [2] NUS [3] ByteDance [4] UCAS [5] CEBSIT, CAS
liangjian92@gmail.com dapeng.hu@u.nus.edu jshfeng@gmail.com rhe@nlpr.ia.ac.cn

Abstract

*To ease the burden of labeling, unsupervised domain adaptation (UDA) aims to transfer knowledge in previous and related labeled datasets (sources) to a new unlabeled dataset (target). Despite impressive progress, prior methods always need to access the raw source data and develop data-dependent alignment approaches to recognize the target samples in a transductive learning manner, which may raise privacy concerns from source individuals. Several recent studies resort to an alternative solution by exploiting the well-trained white-box model from the source domain, yet, it may still leak the raw data via generative adversarial learning. This paper studies a practical and interesting setting for UDA, where only black-box source models (*i.e.*, only network predictions are available) are provided during adaptation in the target domain. To solve this problem, we propose a new two-step knowledge adaptation framework called DIstill and fine-tuNE (DINE). Taking into consideration the target data structure, DINE first distills the knowledge from the source predictor to a customized target model, then fine-tunes the distilled model to further fit the target domain. Besides, neural networks are not required to be identical across domains in DINE, even allowing effective adaptation on a low-resource device. Empirical results on three UDA scenarios (*i.e.*, single-source, multi-source, and partial-set) confirm that DINE achieves highly competitive performance compared to state-of-the-art data-dependent approaches. Code is available at `https://github.com/tim-learn/DINE/`.*

1. Introduction

Deep neural networks achieve remarkable progress with massive labeled data, but it is expensive and not efficient to collect enough labeled data for each new task. To reduce the burden of labeling, considerable attention has been devoted to the transfer learning field [55, 89], especially for unsupervised domain adaptation (UDA) [3, 10], where one or many related but different labeled datasets are collected as source domain(s) to help recognize unlabeled instances in the new

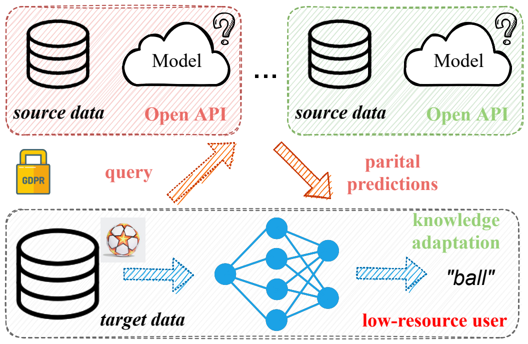


Figure 1. A challenging but interesting domain adaptation problem setting. One or many source agents only provide their black-box predictors (*e.g.*, via the cloud API service) to the target user with certain unlabeled data, where neither *the raw source data* nor *the details about the source models* is accessible during adaptation.

dataset (called target domain). Recently, UDA methods have been widely applied in a variety of computer vision problems, *e.g.*, image classification [18, 50, 73], semantic segmentation [25, 72, 87], and object detection [8, 33, 61].

Existing UDA methods always need to access the raw source data and resort to domain adversarial training [18, 73] or maximum mean discrepancy minimization [49, 74] to align source features with target features. However, in many situations like personal medical records, the raw source data may be missing or must not be shared due to the privacy policy. To tackle this issue, several recent studies [36, 39, 44] attempt to utilize the trained model instead of the raw data from the source domain as supervision and obtain surprisingly good adaptation results. Even so, these methods always require source models to be elegantly trained and provided to the target domain with all the details, which raises two important concerns. First, through generation techniques like generative adversarial learning [21], it is still possible to recover the raw source data, leaking the individual information. Second, the target domain employs the same neural network as the source domain, which is not desirable for low-resource target users. Thus, this paper focuses on a realistic and challenging scenario for UDA, where the source model is trained without bells and whistles and provided to the target domain as a black-box predictor.

For a better illustration, as shown in Fig. 1, the target user exploits the API service offered by one or many source vendors to get the predictions for each instance and utilizes them for adaptation in the unlabeled target domain. To address such a challenging UDA problem, we propose a novel adaptation framework called DIstill and fine-tuNE (DINE). In a nutshell, DINE first distills the knowledge from predictions by source models and then fine-tunes the distilled model with the target data itself, forming a simple two-step approach. Note that, vanilla knowledge distillation [23] requires the existence of labeled data and learns the target model (student) by imitating the full outputs of the source model (teacher). Yet, besides the absence of labeled target data, acquiring the full teacher outputs is also impracticable in many situations, *e.g.*, some predictors merely offer several highest soft-max probability and their associated labels. To alleviate this issue, we devise an adaptive label smoothing technique on source predictions by keeping the largest soft-max value and forcing the rest with the same values.

On top of the point-wise supervision above, we introduce two kinds of structural regularizations into distillation for the first time: interpolation consistency training [77]—which encourages the prediction of interpolated samples to be consistent with the interpolated predictions; and mutual information maximization [26, 44]—which helps increase the diversity among the target predictions. Thereafter, we aim to fit the target structure by adjusting parameters in the learned distilled model using the target data alone. For the sake of simplicity, we re-use mutual information maximization to fine-tune the distilled target model. As for multi-source UDA, we readily extend DINE by aggregating the outputs from multiple source predictors instead. We highlight the main contribution as follows:

- We study a realistic and challenging UDA problem and propose a new adaptation framework (DINE) with only black-box predictors provided from source domains.
- We propose an adaptive label smoothing strategy and a structural distillation method by first introducing structural regularizations into unsupervised distillation.
- Empirical results on various benchmarks validate the superiority of the DINE framework over baselines. Provided with large source predictors like ViT [13], DINE even yields state-of-the-art performance for single-source, multi-source, and partial-set UDA.

Compared to existing methods, DINE has several appealing aspects: 1) **safe**. It does not access the raw source data nor the source model, avoiding information leakage from source agents; 2) **efficient**. It does not assume the same architecture across domains, so it can learn a lightweight target model from large source models. Moreover, it does not involve adversarial training [39] nor data synthesis [36], making the algorithm converge much faster.

2. Related Work

Domain Adaptation. Domain adaptation uses labeled data in one or more source domains to solve new tasks in a target domain. This paper mainly focuses on a challenging problem—unsupervised domain adaptation (UDA), where no labeled data is available in the target domain. At early times, researchers address this problem via instance weighting [27, 66], feature transformation [42, 54, 54], and feature space [17, 20, 67]. In the last decade, benefiting from representation learning, deep domain adaptation methods are prevailing and achieve remarkable progress. To mitigate the gap between features across different domains, domain adversarial learning [18,24,50,73] and discrepancy minimization [32,35,51,74] are widely used within deep UDA methods. Besides, another line of UDA methods [6,11,30,62] focus on the network outputs and develop various regularization terms to pursue implicit domain alignment. In addition, researchers investigate other aspects of neural networks for UDA, *e.g.*, domain-specific normalization-based methods [5, 53] and feature regularization-based methods [7, 79]. To fully verify the effectiveness, we study three UDA cases, *i.e.*, single-source, multi-source [41, 57, 81], and partial-set (source label space subsumes target label space) [4, 47, 86].

Model Transfer (Source data-free UDA). Early parameter adaptation methods [14, 31] adapt the classifier trained in the source domain to the target domain with a small set of labeled examples, hence limiting their application in semi-supervised DA. Besides empirical success, [37] pioneers the theoretical analysis of hypothesis transfer learning for linear regression. Inspired by this paradigm and increasingly important privacy concerns, [9, 43] develop several shallow adaptation methods without source data. Several recent studies [36,39,44,46] introduce the source data-free setting for deep UDA, where the source domain merely offers a trained model. Specifically, [44] freezes the classifier layer and fine-tunes the feature module via information maximization and pseudo-labeling in the target domain, which is further extended to multi-source UDA [1, 16]. [39] leverages a conditional generative adversarial net and incorporates generated images into the adaptation process. However, exposing details of the trained source model is fairly risky due to some committed white-box attacks. Faced with a black-box source model, [46] divides the target dataset into two splits and employs semi-supervised learning to enhance the performance of the uncertain split. [48] focuses on black-box label shift, but it requires a hold-out source set to estimate class confusion matrix, which is sometimes hard to satisfy. Moreover, [85] proposes an iterative noisy label learning approach using full soft labels. DINE focuses on the black-box covariate shift problem and works well even only hard labels from source predictors provided.

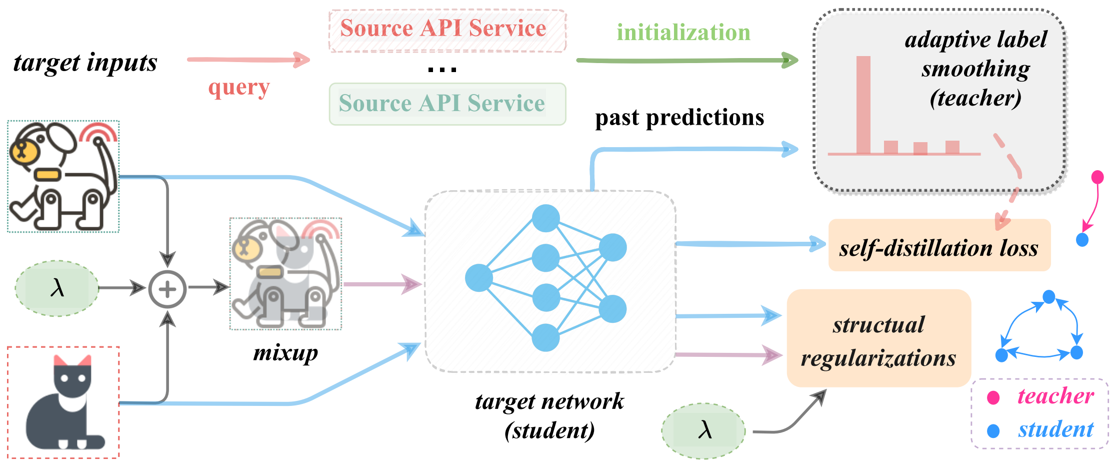


Figure 2. An overview of the proposed DINE framework. Black-box source predictors (*i.e.*, API service) are merely required to initialize the memory bank that stores the predictions of each target instance. In the self-distillation loss, the memory bank could be considered as a teacher that maintains an exponential moving average (EMA) prediction. Structural regularizations, describing batch-wise and pair-wise data structure, are further employed during adaptation. During the fine-tuning step, only the mutual information objective is activated.

Knowledge Distillation. Knowledge distillation aims to transfer knowledge from one model (*a.k.a.*, teacher) to another model (*a.k.a.*, student), usually from a larger one to a smaller one. [23] shows that, augmenting the training of the student with a distillation loss, matching the predictions between teacher and student, is beneficial. In fact, knowledge distillation could be considered as a learned label smoothing regularization [68], which has a similar function with the latter [83]. Recently, [34] proposes self-knowledge distillation and shows that, even within the same neural network, the past predictions could be the teacher itself. Besides supervised training, such self-distillation could be effectively applied with unlabeled data for semi-supervised learning. For instance, [38] proposes to ensemble the predictions during training, using the outputs of a single network on different training epochs, as a teacher for the current epoch. Instead of maintaining an exponential moving average (EMA) prediction in [38], [71] utilizes an average of consecutive student models (past model weights) as a stronger teacher, but is not applicable for black-box UDA here. DINE proposes an adaptive label smoothing technique on source predictions and for the first time introduces structural regularizations [19, 77] into unsupervised distillation.

3. Methodology

We mainly focus on the K-way cross-domain image classification task and aim to address an interesting, realistic but challenging UDA setting, where only single or multiple black-box source predictors are provided to the unlabeled target domain. For the single-source UDA scenario, the source domain $\{x_s^i, y_s^i\}_{i=1}^{n_s}$ consists of n_s labeled instances, where $x_s^i \in \mathcal{X}_s, y_s^i \in \mathcal{Y}_s$, and the target domain $\{x_t^i, y_t^i\}_{i=1}^{n_t}$ consists of n_t unlabeled instances, where $x_t^i \in \mathcal{X}_t, y_t^i \in \mathcal{Y}_t$, and the goal is to infer the values of $\{y_t^i\}_{i=1}^{n_t}$. The label spaces are always assumed the same across domains, *i.e.*, $\mathcal{Y}_s = \mathcal{Y}_t$, for single-source and multi-source UDA. By contrast, partial-set UDA [4] assumes that some source classes does not exist in the target domain, *i.e.*, $\mathcal{Y}_s \supset \mathcal{Y}_t$. Concerning the black-box adaptation setting, only these trained source models are provided, without requiring access to the source data. It also differs from prior model adaptation methods [39, 44] in requiring no details about the source model, *e.g.*, backbone type and network parameters. In particular, *only the hard or soft network predictions of the target instances $\mathcal{X}_t$ from the source model $f_s : \mathcal{X}_s \to \mathcal{Y}_s$* are utilized for single-source adaptation in the target domain.

3.1. Architecture and Source Model Generation

We elaborate on how to obtain the trained model from the source domain as follows. Unlike [44,46] that elegantly design the source model with a bottleneck layer and the weight normalization [63] technique, we just insert a single linear fully-connected (FC) layer after the backbone feature network, and use label smoothing (LS) [68] to train f_s,

$$\mathcal{L}_s(f_s; \mathcal{X}_s, \mathcal{Y}_s) = -\mathbb{E}_{(x_s,y_s)\in\mathcal{X}_s\times\mathcal{Y}_s} \, (q^s)^T \log f_s(x_s), \quad (1)$$

where $q^s = (1-\alpha)\mathbf{1}_{y_s} + \alpha/K$ is the smoothed label vector and α is the smoothing parameter empirically set to 0.1, and $\mathbf{1}_j$ denotes a K-dimensional one-hot encoding with only the j-th value being 1.

As for the self-defined target network $f_t : \mathcal{X}_t \to \mathcal{Y}_t$, we follow [44,46] and adopt the bottleneck layer consisting of a batch-normalization layer and an FC layer, and the classifier consisting of a weight-normalization layer and an FC layer.

3.2. Adaptive Self-Knowledge Distillation

In order to extract knowledge from a black-box model, there exists a natural solution called knowledge distillation (KD) [23] by forcing the target model (student) to learn similar predictions to the source model (teacher). However, existing KD methods are applied to a supervised training task, and the consistency loss in the following works well by acting as a regularization term,

$$\mathcal{L}_{kd}(f_t;\mathcal{X}_t,f_s)=\mathbb{E}_{x_t\in\mathcal{X}_t}\mathcal{D}_{kl}\left(f_s(x_t)\mid\mid f_t(x_t)\right), \tag{2}$$

where D_{kl} denotes the Kullback-Leibler (KL) divergence loss. However, the network outputs from the source model f_s for target instances are *not accurate and sometimes even incomplete*. For the studied black-box adaptation problem, highly relying on the teacher $f_s(x_t)$ via a consistency loss above sounds not desirable anymore. Thus, we propose to revise the teacher output p via top-r largest values as,

$$\text{AdaLS}(p,r)_i=\begin{cases}p_i, & i\in\mathcal{T}_p^r,\\ (1-\sum_{j\in\mathcal{T}_p^r}p_j)/(K-r), & \text{otherwise.}\end{cases} \tag{3}$$

Here $\mathcal{T}_p^r$ denotes the index set of top-r classes in p, and we term this transformation in Eq. (3) as adaptive label smoothing (**adaptive LS**), since these instance-specific top-r values are kept which are not the same for different samples. As a byproduct, using the smoothed output ($r=1$) means that we merely need the predicted class along with its maximum probability, which sounds more flexible when using an API service provided by other companies. The refined output AdaLS(p,r) is believed to work better than the original output p for several reasons below, 1) it partially neglects some redundant and noisy information by only focusing on the pseudo label (class associated with the largest value) and forcing a uniform distribution on other classes like label smoothing [68]; 2) it does not solely rely on the noisy pseudo label but utilizes the largest value as confidence, similar to self-weighted pseudo labeling [28]. Generally, we first obtain the smoothed predictions from single or multiple source predictors as the initialized teacher as,

$$P^T(x_t)\leftarrow\frac{1}{M}\sum\nolimits_{m=1}^{M}\text{AdaLS}(f_s^{(m)}(x_t)), \tag{4}$$

where M denotes the number of predictors, $f_s^{(m)}(x_t)$ denotes the predictions of x_t through m-th source predictor.

To further alleviate the noise in the teacher prediction, we follow [34, 38] and adopt a self-distillation strategy, shown in Fig. 2, maintaining an EMA prediction by

$$P^T(x_t)\leftarrow\gamma P^T(x_t)+(1-\gamma)f_t(x_t),\ \forall x_t\in\mathcal{X}_t, \tag{5}$$

where γ is a momentum hyper-parameter. Following [38], we update teacher predictions after every training epoch. When $\gamma=1$, there exists no temporal ensembling, *i.e.*, the source predictions act as a teacher throughout distillation.

3.3. Distillation with Structural Regularizations

As stated above, the teacher output from the source model is highly possible to be inaccurate and noisy due to the domain shift. Even we devise a promising solution in Eq. (4), only the point-wise information is considered during the distillation process, it ignores the data structure in the target domain, thus not enough for effective noisy knowledge distillation. As such, we incorporate the structural information in the target domain to regularize the distillation. First, we consider the pairwise structural information via MixUp [84], and employ the interpolation consistency training [77] technique as below,

$$\begin{aligned}\mathcal{L}_{mix}(f_t;\mathcal{X}_t)=&\,\mathbb{E}_{x_i^t,x_j^t\in\mathcal{X}_t}\mathbb{E}_{\lambda\in\text{Beta}(\alpha,\alpha)}\\ &l_{ce}\left(\text{Mix}_\lambda\left(f_t'(x_i^t),f_t'(x_j^t)\right),f_t\left(\text{Mix}_\lambda(x_i^t,x_j^t)\right)\right),\end{aligned} \tag{6}$$

where l_{ce} denotes the cross-entropy loss, and $\text{Mix}_\lambda(a,b)=\lambda\cdot a+(1-\lambda)\cdot b$ denotes the MixUp operation, and λ is sampled from a Beta distribution, and α is the hyper-parameter empirically set to 0.3 [84]. f_t' just offers the values of f_t but needs no gradient optimization. Here we do not adopt the EMA update strategy in [77] for f_t'. Eq. (6) can be treated to augment the target domain with more interpolated samples, which is beneficial for better generalization ability.

In addition, we also consider the global structural information during distillation in the target domain. In fact, during distillation, the classes with a large number of instances are relatively easy to learn, which may wrongly recognize some confusing target instances as such classes in turn. To circumvent this problem, we attempt to encourage diversity among the predictions of all the target instances. Specifically, we try to maximize the widely-used mutual information objective [19, 26, 44] in the following,

$$\begin{aligned}\mathcal{L}_{im}(f_t;\mathcal{X}_t)&=H(\mathcal{Y}_t)-H(\mathcal{Y}_t|\mathcal{X}_t)\\ &=h\left(\mathbb{E}_{x_t\in\mathcal{X}_t}f_t(x_t)\right)-\mathbb{E}_{x_t\in\mathcal{X}_t}\,h\left(f_t(x_t)\right),\end{aligned} \tag{7}$$

where $h(p)=-\sum_i p_i\log p_i$ represents the conditional entropy function. Note that, increasing the marginal entropy $H(\mathcal{Y}_t)$ encourages the label distribution to be uniform while decreasing the conditional entropy $H(\mathcal{Y}_t|\mathcal{X}_t)$ encourages unambiguous network predictions.

Integrating these objectives introduced in Eqs. (2, 6, 7) together, we obtain the final loss function as follows,

$$\mathcal{L}_t=\mathcal{D}_{kl}\left(P^T(x_t)\mid\mid f_t(x_t)\right)+\beta\mathcal{L}_{mix}-\mathcal{L}_{im}, \tag{8}$$

where β is a hyper-parameter empirically set to 1, controlling the importance of $\mathcal{L}_{mix}$ during structural distillation.

Different from the most closely related work [85] that iteratively refines the pseudo labels and optimizes the target network, DINE directly learns good network predictions for the target data as a unified approach, which is more desirable to capture the data structure of the target domain.

3.4. Fine-tuning the Distilled Model

Through the proposed structural knowledge distillation method from black-box source predictors $\{f_s^{(m)}\}_{m=1}^{M}$, it is expected to learn a well-performing white-box target model. However, the distilled model seems sub-optimal since it is mainly optimized via the point-wise knowledge distillation term in Eq. (2), which highly depends on the source predictions. Inspired by DIRT-T [65], we hypothesize that a better network is achievable by introducing a secondary training phase that solely minimizes the target-side cluster assumption violation. Rather than employ the parameter-sensitive virtual adversarial training [65], we again employ the mutual information maximization in Eq. (7) to refine the distilled target model. So far, we have shown all the details of two steps within the proposed framework (DINE).

4. Experiments

4.1. Setup

a) Datasets. Office [59] is a popular benchmark on cross-domain object recognition, consisting of three different domains in 31 categories. **Office-Home** [76] is a challenging medium-sized benchmark on object recognition, consisting of four different domains in 65 categories. **VisDA-C** [58] is a large-scale benchmark developed for 12-class synthetic-to-real object recognition. The source domain contains 152 thousand synthetic images generated by rendering 3D models while the target domain has 55 thousand real object images from Microsoft COCO. Gong et al. [20] further extract 10 shared categories between Office and Caltech-256 to form a new benchmark named **Office-Caltech**. **Image-CLEF** is a benchmark for ImageCLEF 2014 domain adaptation challenge [1], organized by selecting the 12 common categories shared by three public datasets.

b) Implementation details. Generally, we randomly run our methods three times with different random seeds {2019, 2020, 2021} via **PyTorch** and report the average accuracies. Regarding the source model f_s, we train it using all the samples in the source domain. In this paper, we mainly consider three different backbones, ResNet-50, ResNet-101 [22] and ViT-B_16 [13] (ViT for simplicity). Following [44], mini-batch SGD is employed to learn the layers initialized from the ImageNet pre-trained model or last stage with the learning rate (1e-3), and new layers from scratch with the learning rate (1e-2). Besides, we use the suggested training settings in [44, 50], including learning rate scheduler, momentum (0.9), weight decay (1e-3), bottleneck size (256), and batch size (64). Concerning the parameters in DINE, $r = 1$ and $T_m = 30$ are adopted for all datasets and tasks except $T_m = 10$ for **VisDA-C**. Moreover, two hyper-parameters $\beta = 1.0, \gamma = 0.6$ are fixed throughout this paper.

[1] http://imageclef.org/2014/adaptation

Table 1. Accuracies (%) on **Office** [59] for single-source closed-set UDA. (Best value of source-prediction-based (Pred.) methods in **bold**), 'Mod.' denotes source-model-based, 'Data' denotes source-data-dependent. *** denotes ViT-based**.

Method	Type	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
No Adapt.	Pred.	79.9	76.6	56.4	92.8	60.9	98.5	77.5
NLL-OT [2]	Pred.	88.8	85.5	64.6	95.1	66.7	98.7	83.2
NLL-KL [85]	Pred.	89.4	86.8	65.1	94.8	67.1	98.7	83.6
HD-SHOT [44]	Pred.	86.5	83.1	66.1	95.1	68.9	98.1	83.0
SD-SHOT [44]	Pred.	89.2	83.7	67.9	95.3	71.1	97.1	84.1
DINE	Pred.	91.6	86.8	72.2	96.2	73.3	98.6	86.4
DINE (full)	Pred.	91.7	87.5	72.9	96.3	73.7	98.5	86.7
ResNet-50$\upuparrows$, **ViT**$\downdownarrows$ (source backbone) → **ResNet-50** (target backbone)								
No Adapt.	Pred.	88.2	89.2	74.5	97.2	77.2	99.3	87.6
NLL-OT [2]	Pred.	91.3	91.4	76.4	97.2	78.2	99.4	89.0
NLL-KL [85]	Pred.	91.7	91.8	76.3	97.2	78.4	99.0	89.1
HD-SHOT [44]	Pred.	88.8	90.9	75.3	97.7	77.7	99.5	88.3
SD-SHOT [44]	Pred.	91.6	92.8	77.8	98.7	78.5	**99.7**	89.8
DINE	Pred.	94.2	94.6	80.7	**98.8**	81.5	99.5	91.6
DINE (full)	Pred.	**95.5**	**94.8**	**81.2**	98.5	**82.0**	**99.7**	**91.9**
SHOT [46]	Mod.	93.9	90.1	75.3	98.7	75.0	99.9	88.8
SHOT++ [46]	Mod.	94.5	90.9	76.3	98.6	75.8	99.9	89.3
A^{2}Net [78]	Mod.	94.5	94.0	76.7	99.2	76.1	100.	90.1
TransDA* [80]	Mod.	97.2	95.0	73.7	99.3	79.3	99.6	90.7
SCDA$_{\text{MDD}}$ [40]	Data	95.4	95.3	77.2	99.0	75.9	100.	90.5
SRDC [69]	Data	95.8	95.7	76.7	99.2	77.1	100.	90.8
RADA$_{\text{CDAN}}$ [29]	Data	96.1	96.2	77.5	99.3	77.4	100.	91.1

c) Baselines. Since the black-box UDA setting is fairly new in this field, we come up with several baseline methods below. **No Adapt.** is also known as 'source only' in this field that infers the class label from the predictions. **NLL-KL** mainly follows the idea of noisy label learning (NLL) [85] and adopts the diversity-promoting KL divergence to refine the noisy pseudo labels, then train a network with the refined pseudo labels iteratively. **NLL-OT** differs from NLL-KL only in that the optimal transport (OT) technique [2] is employed instead of the KL divergence in the refining step. **HD-SHOT** first learns a white-box model by self-training in the target domain and then exploits SHOT [44] for further adaptation. It treats the class predicted by f_s as the true label for each target instance and employs a cross-entropy loss to train the model. **SD-SHOT** differs from HD-SHOT only in that a weighted cross-entropy loss is utilized where the predictive confidence is treated as the instance weight.

We provide DINE (w/o FT) and DINE (full) besides DINE. When dropping the second fine-tuning step, DINE becomes DINE (w/o FT). DINE becomes DINE (full) where $r = K$, *i.e.*, full source predictions are utilized. For comparison, we choose SOTA source-model-based (Mod.) methods (*e.g.* [44, 46]) and source-data-dependent (Data) methods (*e.g.*, [29, 40]) which show the best UDA results so far. In addition, we provide the results of a recent ViT-based UDA method—TransDA [80] for fair comparison.

Table 2. Accuracies (%) on **Office-Home** [76] for single-source closed-set UDA.

Method	Type	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.
No Adapt.	Pred.	44.1	66.9	74.2	54.5	63.3	66.1	52.8	41.2	73.2	66.1	46.7	77.5	60.6
NLL-OT [2]	Pred.	49.1	71.7	77.3	60.2	68.7	73.1	57.0	46.5	76.8	67.1	52.3	79.5	64.9
NLL-KL [85]	Pred.	49.0	71.5	77.1	59.0	68.7	72.9	56.4	46.9	76.6	66.2	52.3	79.1	64.6
HD-SHOT [44]	Pred.	48.6	72.8	77.0	60.7	70.0	73.2	56.6	47.0	76.7	67.5	52.6	80.2	65.3
SD-SHOT [44]	Pred.	50.1	75.0	78.8	63.2	72.9	76.4	60.0	48.0	79.4	69.2	54.2	81.6	67.4
DINE	Pred.	52.2	78.4	81.3	65.3	76.6	78.7	62.7	49.6	82.2	69.8	55.8	84.2	69.7
DINE (full)	Pred.	54.2	77.9	81.6	65.9	77.7	79.9	64.1	50.5	82.1	71.1	58.0	84.3	70.6
ResNet-50↑, **ViT**↓ (source backbone) → **ResNet-50** (target backbone)														
No Adapt.	Pred.	54.5	83.2	87.2	78.0	83.8	86.1	74.5	49.7	87.4	78.6	52.6	86.2	75.1
NLL-OT [2]	Pred.	58.8	84.4	87.6	78.2	84.7	86.7	76.0	54.0	88.0	79.7	57.2	87.2	76.9
NLL-KL [85]	Pred.	59.5	84.3	87.6	77.4	84.8	86.8	75.1	54.9	88.0	79.0	57.9	87.2	76.9
HD-SHOT [44]	Pred.	57.2	84.2	87.3	78.4	84.9	86.4	74.8	56.0	87.6	78.9	57.5	87.0	76.7
SD-SHOT [44]	Pred.	59.4	85.2	87.8	79.6	86.6	87.1	76.4	58.3	87.8	80.0	59.5	87.9	78.0
DINE	Pred.	**64.9**	87.4	88.8	80.5	**89.6**	87.8	79.0	**62.9**	89.1	81.5	64.6	**90.0**	80.5
DINE (full)	Pred.	64.4	**87.9**	**89.0**	**80.9**	**89.6**	**88.7**	**79.6**	62.5	**89.4**	**81.7**	**65.2**	89.7	**80.7**
SHOT [46]	Mod.	57.7	79.1	81.5	67.6	77.9	77.8	68.1	55.8	82.0	72.8	59.7	84.4	72.0
A^{2}Net [78]	Mod.	58.4	79.0	82.4	67.5	79.3	78.9	68.0	56.2	82.9	74.1	60.5	85.0	72.8
SHOT++ [46]	Mod.	58.1	79.5	82.4	68.6	79.9	79.3	68.6	57.2	83.0	74.3	60.4	85.1	73.0
TransDA* [80]	Mod.	67.5	83.3	85.9	74.0	83.8	84.4	77.0	68.0	87.0	80.5	69.9	90.0	79.3
RADA$_{\text{CDAN}}$ [29]	Data	56.5	76.5	79.5	68.8	76.9	78.1	66.7	54.1	81.0	75.1	58.2	85.1	71.4
ATDOC-NA [45]	Data	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
SCDA$_{\text{DCAN}}$ [40]	Data	60.7	76.4	82.8	69.8	77.5	78.4	68.9	59.0	82.7	74.9	61.8	84.5	73.1

Table 3. Accuracies (%) on **VisDA-C** [58] for single-source closed-set UDA.

Method	Type	plane	bcycl	bus	car	horse	knife	mcycle	person	plant	sktbrd	train	truck	Per-class
No Adapt.	Pred.	64.3	24.6	47.9	75.3	69.6	8.5	79.0	31.6	64.4	31.0	81.4	9.2	48.9
NLL-OT [2]	Pred.	82.6	84.1	76.2	44.8	90.8	39.1	76.7	72.0	82.6	81.2	82.7	50.6	72.0
NLL-KL [85]	Pred.	82.7	83.4	76.7	44.9	90.9	38.5	78.4	71.6	82.4	80.3	82.9	50.4	71.9
HD-SHOT [44]	Pred.	75.8	85.8	78.0	43.1	92.0	41.0	79.9	78.1	84.2	86.4	81.0	65.5	74.2
SD-SHOT [44]	Pred.	79.1	85.8	77.2	43.4	91.6	41.0	80.0	78.3	84.7	86.8	81.1	65.1	74.5
DINE	Pred.	81.4	86.7	77.9	55.1	92.2	34.6	80.8	79.9	87.3	87.9	84.3	58.7	75.6
DINE (full)	Pred.	95.3	85.9	80.1	53.4	93.0	37.7	80.7	79.2	86.3	89.9	85.7	60.4	77.3
source backbone: **ResNet-101** (↑↑), **ViT** (↓↓) → **ResNet-101** (target backbone)														
No Adapt.	Pred.	97.0	56.2	81.0	**74.4**	91.8	52.0	92.5	10.1	73.4	92.7	**97.0**	17.5	69.6
NLL-OT [2]	Pred.	**97.8**	90.8	81.9	49.7	95.7	93.5	85.2	45.4	88.9	**96.6**	91.2	54.4	80.9
NLL-KL [85]	Pred.	97.6	91.1	82.1	49.2	**95.8**	93.5	**86.2**	44.6	89.0	96.4	91.4	54.8	81.0
HD-SHOT [44]	Pred.	96.7	91.7	81.8	48.4	95.1	**98.5**	83.1	60.1	**92.2**	87.7	88.4	**65.3**	82.4
SD-SHOT [44]	Pred.	96.3	91.1	80.3	46.4	93.9	98.2	81.5	58.6	90.9	85.5	88.0	63.8	81.2
DINE	Pred.	96.6	**91.9**	**83.1**	58.2	95.3	97.8	85.0	**73.6**	91.9	94.9	92.2	60.7	**85.1**
DINE (full)	Pred.	96.6	**91.9**	82.9	57.9	95.4	97.8	84.5	73.1	91.7	95.1	92.0	60.9	85.0
A^{2}Net [78]	Mod.	94.0	87.8	85.6	66.8	93.7	95.1	85.8	81.2	91.6	88.2	86.5	56.0	84.3
SHOT [46]	Mod.	95.8	88.2	87.2	73.7	95.2	96.4	87.9	84.5	92.5	89.3	85.7	49.1	85.5
SHOT++ [46]	Mod.	95.8	88.3	90.5	84.7	97.9	98.0	92.9	85.3	97.5	92.9	93.9	32.3	87.5
TransDA* [80]	Mod.	97.2	91.1	81.0	57.5	95.3	93.3	82.7	67.2	92.0	91.8	92.5	54.7	83.0
ATDOC-NA [45]	Data	93.7	83.0	76.9	58.7	89.7	95.1	84.4	71.4	89.4	80.0	86.7	55.1	80.3
STAR [52]	Data	95.0	84.0	84.6	73.0	91.6	91.8	85.9	78.4	94.4	84.7	87.0	42.2	82.7
CAN [32]	Data	97.0	87.2	82.5	74.3	97.8	96.2	90.8	80.7	96.6	96.3	87.5	59.9	87.2

4.2. Results

a) single-source. We first show the results of cross-domain object recognition on **Office** in Table 1. and **Office-Home** in Table 2. As stated above, we adopt two different backbone networks for training the source domain. Typically, UDA methods assume the same network structure across domains, *e.g.*, ResNet-50 in [29, 40, 50]. With the same ResNet-50 backbone, DINE performs better than other baselines, indicating the effectiveness of the proposed distillation strategy. Trained with much stronger source models like ViT, all the black-box UDA methods are significantly strengthened, and DINE achieves the best mean accuracy 91.9% on Office and 80.7% on Office-Home. These results even beat SOTA model-based and data-based UDA methods with clear margins. As can be seen from Table 3, the results on **VisDA-C** again validate the superiority of DINE over other baselines. Similar observations are dis-

Table 4. Accuracies (%) on **Office-Home** [76] for single-source partial-set UDA.

Method	Type	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.
No Adapt.	Pred.	44.9	70.5	80.7	57.5	61.3	67.2	60.9	40.8	76.0	70.9	47.6	76.9	62.9
NLL-OT [2]	Pred.	42.7	64.2	71.7	57.2	58.5	64.5	56.7	41.6	67.5	64.2	45.1	69.0	58.6
NLL-KL [85]	Pred.	38.9	53.8	60.5	49.2	50.5	55.9	50.0	38.9	58.0	57.0	41.7	59.6	51.2
HD-SHOT [44]	Pred.	51.2	76.2	85.7	68.8	70.6	77.5	69.2	49.6	81.4	75.9	54.1	80.7	70.1
SD-SHOT [44]	Pred.	54.2	81.8	88.9	74.8	76.5	81.0	73.5	50.6	84.2	79.8	58.4	83.7	74.0
DINE	Pred.	58.1	83.4	89.2	78.0	80.0	80.6	74.2	56.6	85.9	80.6	62.9	84.8	76.2
DINE (full)	Pred.	55.6	79.0	85.3	75.3	77.6	78.5	74.1	56.7	83.8	78.4	59.6	83.1	73.9
source backbone: **ResNet-50** (↑↑), **ViT** (↓↓) → **ResNet-50** (target backbone)														
No Adapt.	Pred.	55.4	83.5	89.1	79.0	80.6	84.3	77.7	45.8	87.7	83.3	52.6	85.5	75.4
NLL-OT [2]	Pred.	49.7	75.0	80.2	68.8	73.1	77.2	69.3	44.6	80.2	75.8	49.1	78.5	68.5
NLL-KL [85]	Pred.	45.9	59.4	62.2	57.7	59.4	59.6	56.0	41.5	60.8	58.4	44.8	60.1	55.5
HD-SHOT [44]	Pred.	54.9	82.2	88.0	79.6	77.7	83.6	78.1	48.4	86.0	83.4	53.1	81.3	74.7
SD-SHOT [44]	Pred.	58.2	84.1	89.7	82.5	81.7	84.8	81.8	51.0	88.9	85.8	57.7	83.6	77.5
DINE	Pred.	**67.8**	**92.0**	**91.8**	**84.5**	89.1	**87.7**	83.5	**63.9**	**91.7**	**87.0**	**65.9**	**91.3**	**83.0**
DINE (full)	Pred.	65.3	90.3	91.1	84.3	**89.7**	86.4	**83.7**	62.1	91.2	86.3	64.7	90.3	82.1
SHOT [46]	Mod.	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
SHOT++ [46]	Mod.	66.0	86.1	92.8	77.9	77.5	87.6	78.6	66.4	89.7	81.5	67.9	87.2	79.9
TransDA* [80]	Mod.	73.0	79.5	90.9	72.0	83.4	86.0	81.1	71.0	86.9	87.8	74.9	89.2	81.3
MCC [30]	Data	63.1	80.8	86.0	70.8	72.1	80.1	75.0	60.8	85.9	78.6	65.2	82.8	75.1
JUMBOT [15]	Data	62.7	77.5	84.4	76.0	73.3	80.5	74.7	60.8	85.1	80.2	66.5	83.9	75.5
BA^{3}US [47]	Data	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

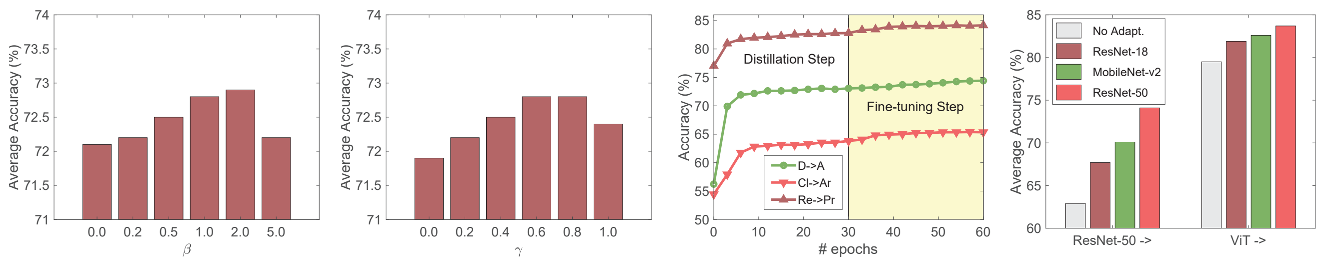
Table 5. Accuracies (%) on four different datasets for multi-source closed-set UDA.

Dataset		**Office** [59]				**Image-CLEF** [51]				**Office-Caltech** [20]					**Office-Home** [76]				
Method	Type	→A	→D	→W	Avg.	→C	→I	→P	Avg.	→A	→C	→D	→W	Avg.	→Ar	→Cl	→Pr	→Re	Avg.
No Adapt.	Pred.	64.5	82.3	80.7	75.8	92.1	87.4	72.4	84.0	84.9	88.7	93.0	88.5	88.8	54.9	49.9	69.6	76.7	62.8
DINE (w/o FT)	Pred.	69.2	98.6	96.9	88.3	96.2	91.4	78.3	88.6	95.0	92.0	98.5	97.3	95.7	70.8	57.1	80.9	82.1	72.7
DINE	Pred.	76.8	**99.2**	98.4	91.5	**98.0**	93.4	80.2	90.5	95.9	95.2	98.5	**98.9**	97.1	74.8	64.1	85.0	84.6	77.1
DINE (full)	Pred.	77.1	**99.2**	98.2	91.5	97.8	93.0	79.7	90.2	96.1	95.3	98.1	**98.9**	97.1	74.9	62.6	84.6	84.7	76.7
source backbone: **ResNet-101** (↑↑), **ViT** (↓↓) → **ResNet-101** (target backbone)																			
No Adapt.	Pred.	77.2	88.2	89.2	84.9	95.3	90.2	72.0	85.9	92.9	95.9	98.7	95.8	95.8	74.5	54.5	83.2	87.2	74.8
DINE (w/o FT)	Pred.	80.7	98.4	97.1	92.1	97.2	**96.6**	80.9	91.6	96.4	96.0	99.4	98.2	97.5	82.4	61.0	88.6	90.8	80.7
DINE	Pred.	**82.4**	**99.2**	98.4	**93.4**	97.8	**96.6**	81.3	**91.9**	**96.8**	**97.0**	99.6	98.8	**98.0**	**83.6**	**67.0**	**90.9**	**91.8**	**83.3**
DINE (full)	Pred.	81.4	99.0	**98.5**	93.0	97.8	96.4	**81.4**	**91.9**	96.7	96.4	**99.8**	98.6	97.9	83.4	65.2	90.3	91.5	82.6
SHOT [46]	Mod.	-	-	-	-	-	-	-	-	96.2	96.2	98.5	99.8	97.7	73.0	60.4	83.9	83.3	75.2
DECISION [1]	Mod.	75.4	98.4	99.6	91.1	-	-	-	-	95.9	95.9	100.	99.6	98.0	74.5	59.4	84.4	83.6	75.5
SHOT++ [46]	Mod.	-	-	-	-	-	-	-	-	96.2	96.5	99.4	100.	98.0	73.1	61.3	84.3	84.0	75.7
CAiDA [12]	Mod.	75.8	99.8	98.9	91.6	-	-	-	-	96.8	97.1	100.	99.8	98.4	75.2	60.5	84.7	84.2	76.2
SImpAl$_{101}$ [75]	Data	71.2	99.4	97.9	89.5	95.2	91.7	78.0	88.3	95.6	94.6	100.	100.	97.5	73.4	62.4	81.0	82.7	74.8
MFSAN [88]	Data	72.7	99.5	98.5	90.2	95.4	93.6	79.1	89.4	-	-	-	-	-	72.1	62.0	80.3	81.8	74.1
MIAN-γ [56]	Data	76.2	99.2	98.4	91.3	-	-	-	-	-	-	-	-	-	69.9	64.2	80.9	81.5	74.1
PCT [70]	Data	-	-	-	-	-	-	-	-	-	-	-	-	-	76.3	64.1	84.9	84.3	77.4

covered on this dataset, *i.e.*, the stronger the source model is, the better results black-box UDA methods obtain. Typically, UDA methods always assume the same network structure (ResNet-101) across domains. Compared with these methods via ResNet-101, DINE obtains competitive results and even beats ViT-based TransDA [80].

b) partial-set. In addition to the closed-set UDA problem, we also investigate the generalization ability of these black-box methods for partial-set UDA. We follow [4, 47] and only select the first 25 classes in alphabetical order in the target domain. As shown in Table 4, NLL-OT and NLL-KL even under-perform 'No Adapt.' due to the challenging asymmetric label spaces. Moreover, the proposed DINE achieves consistently better results than HD-SHOT and SD-SHOT. Different from previous tables, DINE even outperforms DINE (full) for both source backbones by utilizing the proposed adaptive label smoothing technique. With the ViT-based source predictor, DINE obtains the highest average accuracy (83.0%) which is also fairly higher than previous state-of-the-arts in [46,47] via ResNet-50 and TransDA.

c) multi-source. We also study the performance of multi-source UDA [41, 57, 81] where multiple source domains exist. As shown in Table 5, we choose four different multi-source datasets. Within each dataset, we aim to trans-



(a) sensitivity to β ($\gamma = 0.6$) (b) sensitivity to γ ($\beta = 1.0$) (c) accuracy convergence (d) different target backbones

Figure 3. **Analysis on DINE for three UDA tasks**. (a-b) plot the average accuracies with different values of β, γ; (c) shows the accuracies during the training process; (d) shows the average accuracies when choosing target backbone in {MobileNet-v2, ResNet-18, ResNet-50}.

Table 6. **Ablation**. Results of different variants for single-source closed-set UDA with ResNet-50 (source) → ResNet-50 (target).

Method	FT	D → A	Cl → Ar	Re → Pr	Avg.
DINE (w/o $\mathcal{L}_{im}$)	×	62.7±2.2	60.6±0.7	80.4±0.3	67.9
	✓	63.6±3.5	59.6±1.1	81.4±0.1	68.2
DINE (w/o $\mathcal{L}_{mix}$)	×	70.2±1.4	63.2±0.4	82.7±0.4	72.1
	✓	72.0±1.7	64.6±0.2	84.0±0.3	73.5
DINE	×	70.6±2.0	64.1±0.3	83.7±0.4	72.8
	✓	**72.2**±1.8	**65.3**±0.2	**84.7**±0.1	**74.1**

Table 7. **Study on the AdaLS**. Results for closed-set UDA with ResNet-50 (source) → ResNet-50 (target). Hard: one-hot label.

Method	FT	D → A	Cl → Ar	Re → Pr	Avg.
DINE (w/ Hard)	×	63.6±2.1	60.9±0.4	80.2±0.6	68.2
	✓	67.3±1.7	62.3±0.2	81.7±0.4	70.4
DINE (w/ LS)	×	64.8±1.9	61.7±0.4	81.0±0.5	69.2
	✓	68.4±1.8	63.0±0.5	82.6±0.3	71.3
DINE ($r = 1$)	×	70.6±2.0	64.1±0.3	83.7±0.4	72.8
	✓	72.2±1.8	65.3±0.2	84.7±0.1	74.1
DINE ($r = 3$)	×	71.4±2.5	64.4±0.5	83.7±0.3	73.1
	✓	**72.9**±2.0	65.6±0.5	**84.7**±0.3	**74.4**
DINE ($r = K$)	×	71.3±2.7	64.6±0.4	83.3±0.4	73.1
	✓	**72.9**±2.5	**65.9**±0.7	84.3±0.3	74.3

fer knowledge from the other subsets to the target subset. It is found that DINE outperforms 'No Adapt.' and DINE (w/o FT), indicating the effectiveness of fine-tuning and structural distillation. Compared with DINE (full), DINE obtains competitive results on all these datasets. With the ViT-based source predictor, DINE again beats prior data-based and model-based multi-source UDA methods.

4.3. Analysis

We study the contribution of different components within DINE, with results shown in Table 6 and Table 7. When $\mathcal{L}_{im}$ or $\mathcal{L}_{mix}$ is dropped, the performance of DINE decrease for all three tasks, verifying their importance. The second step called FT is also universally vital, which enhances a variety of variants in two tables. Regarding the devised AdaLS technique in Eq. (3), we compare it with one-hot encoding (Hard) and vanilla labeling smoothing (LS). With or without the FT step, AdaLS ($r = 1$) works better than Hard and LS and is competitive to DINE ($r = K$). If we remain more classes, *e.g.*, $r = 3$, AdaLS even works better than using full vectors within DINE.

We study the parameter sensitivity of β, γ in Fig. 3(a-b), where β is in the range of [0.0, 0.2, 0.5, 1.0, 2.0, 5.0], and γ is in the range of [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]. It is easy to find the results around the selected parameters $\beta = 1.0, \gamma = 0.6$ are quite stable. Note that other parameters $\beta = 2.0, \gamma = 0.8$ may be better via oracle validation. Besides, in Fig. 3(c), in both steps of DINE, the accuracies keep increasing and become convergent.

4.4. Discussion & Limitation

To evaluate DINE with small models for a low-resource target user, we provide the results of ResNet-18 and MobileNet-v2 [64] being the target backbone, respectively. As shown in Fig. 3(d), small target models manage to achieve competitive performance given a strong source predictor like ViT. The main limitation is that this paper does not cover UDA with unknown classes in the target domain [36, 60, 82], which needs to be investigated in the future.

5. Conclusion

We explore a new but realistic UDA setting where each source domain only provides its black-box predictor to the target domain, allowing different networks for different domains. Thereafter, we propose a simple yet effective two-step structural distillation framework called Distill and Fine-tune (DINE). DINE elegantly refines the noisy teacher output via adaptive smoothing and fully considers the data structure in the target domain during distillation. Experiments on multiple datasets verify the superiority of DINE over baselines for various UDA tasks. Provided with strong pre-trained source models, DINE even achieves state-of-the-art adaptation results with a small efficient target model.

Acknowledgment. This work was partially funded by National Natural Science Foundation of China under Grants U21B2045 and U20A20223; and Beijing Nova Program under Grant Z211100002121108.

References

[1] Sk Miraj Ahmed, Dripta S Raychaudhuri, Sujoy Paul, Samet Oymak, and Amit K Roy-Chowdhury. Unsupervised multi-source domain adaptation without access to source data. In *Proc. CVPR*, 2021. 2, 7

[2] YM Asano, C Rupprecht, and A Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *Proc. ICLR*, 2019. 5, 6, 7

[3] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. In *Proc. NeurIPS*, 2007. 1

[4] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proc. ECCV*, 2018. 2, 3, 7

[5] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *Proc. CVPR*, 2019. 2

[6] Minghao Chen, Hongyang Xue, and Deng Cai. Domain adaptation for semantic segmentation with maximum squares loss. In *Proc. ICCV*, 2019. 2

[7] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *Proc. ICML*, 2019. 2

[8] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *Proc. CVPR*, 2018. 1

[9] Boris Chidlovskii, Stéphane Clinchant, and Gabriela Csurka. Domain adaptation in the absence of source domain data. In *Proc. KDD*, 2016. 2

[10] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017. 1

[11] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations. In *Proc. CVPR*, 2020. 2

[12] Jiahua Dong, Zhen Fang, Anjin Liu, Gan Sun, and Tongliang Liu. Confident anchor-induced multi-source free domain adaptation. In *Proc. NeurIPS*, 2021. 7

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. ICLR*, 2021. 2, 5

[14] Lixin Duan, Ivor W Tsang, Dong Xu, and Stephen J Maybank. Domain transfer svm for video concept detection. In *Proc. CVPR*, 2009. 2

[15] Kilian Fatras, Thibault Séjourné, Rémi Flamary, and Nicolas Courty. Unbalanced minibatch optimal transport; applications to domain adaptation. In *Proc. ICML*, 2021. 7

[16] Hao-Zhe Feng, Zhaoyang You, Minghao Chen, Tianye Zhang, Minfeng Zhu, Fei Wu, Chao Wu, and Wei Chen. Kd3a: Unsupervised multi-source decentralized domain adaptation via knowledge distillation. In *Proc. ICML*, 2021. 2

[17] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proc. ICCV*, 2013. 2

[18] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 1, 2

[19] Ryan Gomes, Andreas Krause, and Pietro Perona. Discriminative clustering by regularized information maximization. In *Proc. NeurIPS*, 2010. 3, 4

[20] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proc. CVPR*, 2012. 2, 5, 7

[21] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NeurIPS*, 2014. 1

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 5

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 2, 3, 4

[24] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proc. ICML*, 2018. 2

[25] Dapeng Hu, Jian Liang, Qibin Hou, Hanshu Yan, and Yunpeng Chen. Adversarial domain adaptation with prototype-based normalized output conditioner. *IEEE Transactions on Image Processing*, 30:9359–9371, 2021. 1

[26] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proc. ICML*, 2017. 2, 4

[27] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. In *Proc. NeurIPS*, 2006. 2

[28] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proc. CVPR*, 2019. 4

[29] Xin Jin, Cuiling Lan, Wenjun Zeng, and Zhibo Chen. Re-energizing domain discriminator with sample relabeling for adversarial domain adaptation. In *Proc. ICCV*, 2021. 5, 6

[30] Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation. In *Proc. ECCV*, 2020. 2, 7

[31] Thorsten Joachims et al. Transductive inference for text classification using support vector machines. In *Proc. ICML*, 1999. 2

[32] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proc. CVPR*, 2019. 2, 6

[33] Mehran Khodabandeh, Arash Vahdat, Mani Ranjbar, and William G Macready. A robust learning approach to domain adaptive object detection. In *Proc. ICCV*, 2019. 1

[34] Kyungyul Kim, ByeongMoon Ji, Doyoung Yoon, and Sangheum Hwang. Self-knowledge distillation with progressive refinement of targets. In *Proc. ICCV*, 2021. 3, 4

[35] Piotr Koniusz, Yusuf Tas, and Fatih Porikli. Domain adaptation by mixture of alignments of second-or higher-order scatter tensors. In *Proc. CVPR*, 2017. 2

[36] Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *Proc. CVPR*, 2020. 1, 2, 8

[37] Ilja Kuzborskij and Francesco Orabona. Stability and hypothesis transfer learning. In *Proc. ICML*, 2013. 2

[38] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *Proc. ICLR*, 2017. 3, 4

[39] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *Proc. CVPR*, 2020. 1, 2, 3

[40] Shuang Li, Mixue Xie, Fangrui Lv, Chi Harold Liu, Jian Liang, Chen Qin, and Wei Li. Semantic concentration for domain adaptation. In *Proc. ICCV*, 2021. 5, 6

[41] Yunsheng Li, Lu Yuan, Yinpeng Chen, Pei Wang, and Nuno Vasconcelos. Dynamic transfer for multi-source domain adaptation. In *Proc. CVPR*, 2021. 2, 7

[42] Jian Liang, Ran He, Zhenan Sun, and Tieniu Tan. Aggregating randomized clustering-promoting invariant projections for domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(5):1027–1042, 2018. 2

[43] Jian Liang, Ran He, Zhenan Sun, and Tieniu Tan. Distant supervised centroid shift: A simple and efficient approach to visual domain adaptation. In *Proc. CVPR*, 2019. 2

[44] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *Proc. ICML*, 2020. 1, 2, 3, 4, 5, 6, 7

[45] Jian Liang, Dapeng Hu, and Jiashi Feng. Domain adaptation with auxiliary target domain-oriented classifier. In *Proc. CVPR*, 2021. 6

[46] Jian Liang, Dapeng Hu, Yunbo Wang, Ran He, and Jiashi Feng. Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021. 2, 3, 5, 6, 7

[47] Jian Liang, Yunbo Wang, Dapeng Hu, Ran He, and Jiashi Feng. A balanced and uncertainty-aware approach for partial domain adaptation. In *Proc. ECCV*, 2020. 2, 7

[48] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *Proc. ICML*, 2018. 2

[49] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proc. ICML*, 2015. 1

[50] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Proc. NeurIPS*, 2018. 1, 2, 5, 6

[51] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *Proc. ICML*, 2017. 2, 7

[52] Zhihe Lu, Yongxin Yang, Xiatian Zhu, Cong Liu, Yi-Zhe Song, and Tao Xiang. Stochastic classifiers for unsupervised domain adaptation. In *Proc. CVPR*, 2020. 6

[53] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *Proc. ICCV*, 2017. 2

[54] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks*, 22(2):199–210, 2010. 2

[55] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2009. 1

[56] Geon Yeong Park and Sang Wan Lee. Information-theoretic regularization for multi-source domain adaptation. In *Proc. ICCV*, 2021. 7

[57] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proc. ICCV*, 2019. 2, 7

[58] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017. 5, 6

[59] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proc. ECCV*, 2010. 5, 7

[60] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, and Kate Saenko. Universal domain adaptation through self supervision. In *Proc. NeurIPS*, 2020. 8

[61] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Strong-weak distribution alignment for adaptive object detection. In *Proc. CVPR*, 2019. 1

[62] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proc. CVPR*, 2018. 2

[63] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Proc. NeurIPS*, 2016. 3

[64] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. CVPR*, 2018. 8

[65] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *Proc. ICLR*, 2018. 5

[66] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proc. NeurIPS*, 2007. 2

[67] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proc. AAAI*, 2016. 2

[68] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proc. CVPR*, 2016. 3, 4

[69] Hui Tang, Ke Chen, and Kui Jia. Unsupervised domain adaptation via structurally regularized deep clustering. In *Proc. CVPR*, 2020. 5

[70] Korawat Tanwisuth, Xinjie Fan, Huangjie Zheng, Shujian Zhang, Hao Zhang, Bo Chen, and Mingyuan Zhou. A prototype-oriented framework for unsupervised domain adaptation. In *Proc. NeurIPS*, 2021. 7

[71] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. NeurIPS*, 2017. 3

[72] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proc. CVPR*, 2018. 1

[73] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proc. CVPR*, 2017. 1, 2

[74] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 1, 2

[75] Naveen Venkat, Jogendra Nath Kundu, Durgesh Kumar Singh, Ambareesh Revanur, and R Venkatesh Babu. Your classifier can secretly suffice multi-source domain adaptation. In *Proc. NeurIPS*, 2020. 7

[76] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proc. CVPR*, 2017. 5, 6, 7

[77] Vikas Verma, Kenji Kawaguchi, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. In *Proc. IJCAI*, 2019. 2, 3, 4

[78] Haifeng Xia, Handong Zhao, and Zhengming Ding. Adaptive adversarial network for source-free domain adaptation. In *Proc. ICCV*, 2021. 5, 6

[79] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *Proc. ICCV*, 2019. 2

[80] Guanglei Yang, Hao Tang, Zhun Zhong, Mingli Ding, Ling Shao, Nicu Sebe, and Elisa Ricci. Transformer-based source-free domain adaptation. *arXiv preprint arXiv:2105.14138*, 2021. 5, 6, 7

[81] Luyu Yang, Yogesh Balaji, Ser-Nam Lim, and Abhinav Shrivastava. Curriculum manager for source selection in multi-source domain adaptation. In *Proc. ECCV*, 2020. 2, 7

[82] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proc. CVPR*, 2019. 8

[83] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proc. CVPR*, 2020. 3

[84] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. ICLR*, 2017. 4

[85] Haojian Zhang, Yabin Zhang, Kui Jia, and Lei Zhang. Unsupervised domain adaptation of black-box source models. *arXiv preprint arXiv:2101.02839v1*, 2021. 2, 4, 5, 6, 7

[86] Jing Zhang, Zewei Ding, Wanqing Li, and Philip Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *Proc. CVPR*, 2018. 2

[87] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *Proc. ICCV*, 2017. 1

[88] Yongchun Zhu, Fuzhen Zhuang, and Deqing Wang. Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources. In *Proc. AAAI*, pages 5989–5996, 2019. 7

[89] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proc. IEEE*, 109(1):43–76, 2020. 1