Prior Knowledge Guided Unsupervised Domain Adaptation

Tao Sun[1(✉)], Cheng Lu[2], and Haibin Ling[1]

[1] Stony Brook University, Stony Brook, USA
{tao,hling}@cs.stonybrook.edu
[2] XPeng Motors, Guangzhou, China
luc@xiaopeng.com

Abstract. The waive of labels in the target domain makes Unsupervised Domain Adaptation (UDA) an attractive technique in many real-world applications, though it also brings great challenges as model adaptation becomes harder without labeled target data. In this paper, we address this issue by seeking compensation from target domain prior knowledge, which is often (partially) available in practice, *e.g.*, from human expertise. This leads to a novel yet practical setting where in addition to the training data, some prior knowledge about the target class distribution are available. We term the setting as Knowledge-guided Unsupervised Domain Adaptation (KUDA). In particular, we consider two specific types of prior knowledge about the class distribution in the target domain: *Unary Bound* that describes the lower and upper bounds of individual class probabilities, and *Binary Relationship* that describes the relations between two class probabilities. We propose a general rectification module that uses such prior knowledge to refine model generated pseudo labels. The module is formulated as a Zero-One Programming problem derived from the prior knowledge and a smooth regularizer. It can be easily plugged into self-training based UDA methods, and we combine it with two state-of-the-art methods, SHOT and DINE. Empirical results on four benchmarks confirm that the rectification module clearly improves the quality of pseudo labels, which in turn benefits the self-training stage. With the guidance from prior knowledge, the performances of both methods are substantially boosted. We expect our work to inspire further investigations in integrating prior knowledge in UDA. Code is available at https://github.com/tsun/KUDA.

Keywords: Unsupervised Domain Adaptation · Class prior

1 Introduction

Deep neural networks have shown significant performance improvement in a variety of vision tasks [8,12,23,30]. However, such performance highly relies on massive annotated data, which is often expensive to obtain. Unsupervised Domain

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-19827-4_37.

S. Avidan et al. (Eds.): ECCV 2022, LNCS 13693, pp. 639–655, 2022.
https://doi.org/10.1007/978-3-031-19827-4_37

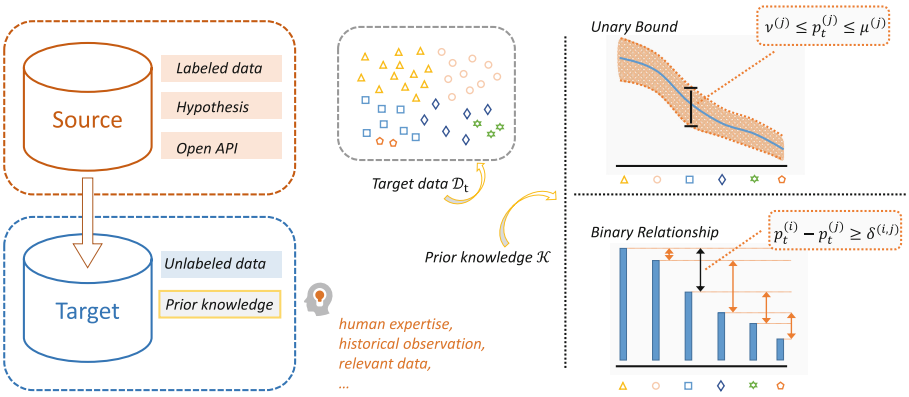

Fig. 1. (Left) Knowledge-guided Unsupervised Domain Adaptation (KUDA). In addition to target data, some prior knowledge about target class distribution is available. (Right) two types of prior knowledge considered in the paper.

Adaptation (UDA) addresses this issue by transferring a predictive model learned from a labeled source domain to an unlabeled target domain [25,38,40]. Despite the advancement made in recent years, UDA remains a challenging task due to the absence of labels in the target domain. On the other hand, in many real-world applications, prior knowledge about the target domain is often readily available. In particular, some information about class distribution is often available without bothering labeling specific target samples. For example, botanists can estimate the proportion of wild species within a reserve using historical information; economists can tell whether vans are more possessed than other vehicles based on the local industrial structure; *etc.* Such prior knowledge may provide valuable clues that are complementary to the unlabeled training data, and can be especially beneficial when there exists a large distribution shift between source and target domains. In fact, prior knowledge has been used to compensate the deficiency of labeled data [14,33], but its systematical integration into UDA solutions remains under-explored.

Inspired by the above observation, in this paper we study a novel setting of UDA, named *Knowledge-guided Unsupervised Domain Adaptation* (KUDA), as illustrated in Fig. 1. Specifically, in addition to target training samples $\mathcal{D}_t$, a collection of prior knowledge $\mathcal{K}$ on target class distribution $p_t(y)$ is accessible. In particular, we consider two types of prior knowledge: *Unary Bound* that describes the lower and upper bounds of individual class probability $p_t^{(c)}$ (*e.g.*, the probability of "square" is between 0.1 and 0.3), and *Binary Relationship* that describes the relations between probabilities of two classes, $p_t^{(c_1)}$ and $p_t^{(c_2)}$ (*e.g.*, there are more "triangles" than "squares"). The task of KUDA is to adapt a predictive model learned from a source domain to a target domain under the guidance from such prior knowledge. It is worth mentioning that there can be many other types of prior knowledge which may help to improve UDA performance, and we choose unary and binary statistics over the class distribution for their generality and accessibility in practice.

To incorporate the prior knowledge into domain adaptation, we propose a novel *rectification module* to refine model generated pseudo labels. We formulate the rectification procedure using prior knowledge as a *Zero-One Programming* (ZOP) [41] problem, where its optimal solution returns the updated pseudo labels. Moreover, smooth regularization is applied to maintain consistency of pseudo labels in neighboring samples. This module can be easily integrated into self-training-based UDA methods. To validate its effectiveness, we choose two recent state-of-the-art UDA methods, SHOT [18] and DINE [19], and improve them with the rectification module.

The experimental validation is conducted on four commonly used UDA benchmarks, two of which have a large label distribution shift by design. The results confirm that the rectification module improves the quality of pseudo labels and hence benefits the self-training stage. Consequently, the performances of two methods under the guidance of prior knowledge, named respectively kSHOT and kDINE, are substantially boosted compared with the vanilla versions. Our work demonstrates that it is important to consider target class prior knowledge, especially when the domain gap is large.

In summary, we make the following contributions:

- We study a novel and practical setting of Knowledge-guided Unsupervised Domain Adaptation (KUDA), where prior knowledge about target class distribution available in addition to unlabeled training samples.
- We introduce a general rectification module that refines pseudo labels with the guidance from prior knowledge. It can be easily plugged into self-training based UDA methods.
- Extensive experiments on both standard and label-shifted benchmarks validate that incorporating prior knowledge can significantly boost the performance of adapted models, reducing the reliance on target training data.

2 Related Work

Incorporating Prior Knowledge. There has been a long history of incorporating prior knowledge into machine learning tasks. Using prior knowledge removes or reduces the reliance on training data. The knowledge can be expressed in various forms, such as statistical descriptions from other data or human expertise, inductive biases, physical models, *etc.* The most related one to our work is *target prior*, where the distribution of target variable $p(y)$ is known [14,33]. In [24], the class distribution prior conditioned on certain inputs is captured by generalized expectation. Zhu *et al.* [46] employ class priors to set thresholds on the propagation of labels. Wang *et al.* [39] assume that a parametric target prior model $p_\eta(y)$ can be obtained from relevant subjects yet having no correspondence with training data. Inductive biases have been widely used in deep neural networks. A canonical one is translation equivariance through convolutions [10,13,37]. Lin *et al.* [20] add geometric priors based on Hough transform in line detection. Physical models of image formation have been integrated into the tasks of image decomposition [1], rain image restoration [17], day-night adaptation [15], *etc.*

Domain Adaptation Settings. Domain Adaptation (DA) presents under many different settings. In the vanilla Unsupervised Domain Adaptation (UDA) [3,29,35,45], only labeled data from a source domain and unlabeled data from a target domain are available. Since no labeled target data is available, UDA can be a challenging task when the domain gap is large. Semi-supervised DA (SSDA) [11,16,32] assumes a few labeled target data is available, which often greatly boosts performance compared with UDA. Active DA [5,28,42] further selects the most informative samples to query their labels from the oracle. Then human-defined criteria like uncertainty and diversity can be injected to measure the informativeness of samples. SSDA and Active DA can be viewed as incorporating additional instance-level label information compared with UDA. Another line is to reduce the information released by source domain, usually due to some privacy issues. Source-data free UDA [2,4,9,18] assumes only a trained model is offered by the source domain while source data are inaccessible. To conceal model details, the black-box source model [19,21,43] is further studied.

Our Study. Our proposed KUDA methods incorporate class distribution-level information and is complementary to all of the above-mentioned settings. It is the first work along this direction, and we expect to see further studies to explore richer prior knowledge for UDA or to extend the idea to general DA scenarios.

3 Knowledge-Guided UDA

Preliminaries. In this paper, we focus on C-way classification problem for UDA tasks. We use $\mathcal{X}$ and $\mathcal{C} = \{0, 1, \ldots, C-1\}$ to denote the input space and the label space respectively. In a vanilla UDA task, we are given labeled samples $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=0}^{n_s-1}$ from a source domain $\mathcal{P}_S(\mathcal{X}, \mathcal{C})$, and unlabeled samples $\mathcal{D}_t = \{(\boldsymbol{x}_i^t)\}_{i=0}^{n_t-1}$ from a target domain $\mathcal{P}_T(\mathcal{X}, \mathcal{C})$. The goal of UDA is to learn a labeling function $f_t = h_t \circ g_t : \mathcal{X} \to \mathcal{C}$ for target domain, where g_t is the feature extractor and h_t is the label predictor.

Prior Knowledge of Target Class Distribution. The class distribution of target domain, $p_t(y)$, is an important quantity while inaccessible in UDA. One way is to estimate it from model predictions on unlabeled target data [22]. However, this can often be unreliable when the domain gap is large. The deficiency of labeled target samples can be compensated with prior knowledge, *e.g.*, from human expertise. In fact, it is often possible to obtain some information about class distribution without bothering labeling specific target samples in real-world applications. Table 1 lists two types of prior knowledge considered in this paper. *Unary Bound* describes the lower and upper bounds of individual class probability $p_t^{(c)}$, and *Binary Relationship* describes the relations between probabilities of two classes, $p_t^{(c_1)}$ and $p_t^{(c_2)}$. Both statistics over the class distribution are general and easy to obtain in practice. Other types of prior knowledge beyond these can be similarly defined in terms of three or more probabilities.

Setting of KUDA. We study a novel and realistic setting termed **Knowledge-guided Unsupervised Domain Adaptation (KUDA)**. In KUDA, in addition to $\mathcal{D}_s$ and $\mathcal{D}_t$, we have access to some prior knowledge $\mathcal{K}$ about the target

Table 1. Two types of prior knowledge considered in the paper.

Knowledge type	Formulation
Unary Bound (UB)	$\mathcal{K} = \left\{ \left(\nu^{(c)} \leq p_t^{(c)} \leq \mu^{(c)} \right) \middle\vert\, c \in \mathcal{C} \right\}$
Binary Relationship (BR)	$\mathcal{K} = \left\{ \left(p_t^{(c_1)} - p_t^{(c_2)} \geq \delta^{(c_1,c_2)} \right) \middle\vert\, c_1, c_2 \in \mathcal{C} \right\}$

class distribution p_t^c. Such prior knowledge may provide valuable clues that are complementary to the unlabeled training data, and can be especially beneficial when there exists a large distribution shift between source and target domains. In particular, we assume that $\mathcal{K}$ can be expressed in a collection of inequality constraints as listed in Table 1. The goal is to learn an optimal target labeling function f_t under the guidance from the prior knowledge $\mathcal{K}$.

4 Method

4.1 Rectify Pseudo Labels with Prior Knowledge

Let us consider a general situation. Suppose we have a model predicted class probability matrix $P \in \mathbb{R}^{n_t \times C}$ of target data. The pseudo label of the i-th sample can be obtained via $\hat{y}_i^t = \arg\max \boldsymbol{p}_i$, where $\boldsymbol{p}_i$ is the i-th row of P. This procedure can be equivalently expressed in a more compact form using one-hot label representation $\boldsymbol{l}_i$ and its matrix form L (*i.e.*, $\boldsymbol{l}_i$ is the i-th row of L)

$$\hat{L} = \arg\max_{L} \langle L, P \rangle, \quad \text{s.t.} \begin{cases} \sum_c L_{i,c} = 1, \forall i \in [n_t] \\ L_{i,c} \in \{0,1\}, \forall c \in \mathcal{C}, i \in [n_t] \end{cases} \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two matrices, and $[n_t] \triangleq \{0, 1, \ldots, n_t - 1\}$. For the optimal solution $\hat{L}$ of Eq. 1, $\hat{L}_{i,\hat{y}_i^t} = 1$ and $\hat{L}_{i,c} = 0 \; \forall c \neq \hat{y}_i^t$.

Without any prior knowledge, the optimal $\boldsymbol{l}_i$ is assigned independently for each target sample. The empirical class probability $\hat{p}_t^{(c)} = \sum_i L_{i,c}/n_t$ is expected to be close to $p_t^{(c)}$. However, this is often violated when the model predictions are noisy. When the prior knowledge $\mathcal{K}$ about p_t is available, we can use it to rectify pseudo labels so that $\hat{p}_t$ is more compliant with p_t.

Hard Constraint Form. Given the inequalities listed in Table 1, we plug in $\hat{p}_t$ and add the constraints to the optimization problem in Eq. 1. Then the optimization problem in hard constraint form can be formulated as:

– **Unary Bound**.

$$\hat{L} = \arg\max_{L} \langle L, P \rangle, \quad \text{s.t.} \begin{cases} \sum_c L_{i,c} = 1 \quad \forall i \in [n_t] \\ L_{i,c} \in \{0,1\}, \quad \forall c \in \mathcal{C}, i \in [n_t] \\ \sum_i L_{i,c} \geq n_t \nu^{(c)}, \quad \forall c \in \mathcal{C} \\ -\sum_i L_{i,c} \geq -n_t \mu^{(c)}, \quad \forall c \in \mathcal{C} \end{cases} \tag{2}$$

– **Binary Relationship**.

$$\hat{L} = \arg\max_{L} \langle L, P \rangle, \text{ s.t. } \begin{cases} \sum_c L_{i,c} = 1 \quad \forall i \in [n_t] \\ L_{i,c} \in \{0,1\}, \quad \forall c \in \mathcal{C}, i \in [n_t] \\ \sum_i (L_{i,c_1} - L_{i,c_2}) \geq n_t \delta^{(c_1,c_2)}, \quad \forall c_1, c_2 \in \mathcal{C} \end{cases} \tag{3}$$

Equation 2 and Eq. 3 are *Zero-One Programming* problems [41], and can be solved with standard solvers [7]. However, using hard constraint form is not favored. When these constraints are inconsistent, the optimization problem becomes infeasible.

Soft Constraint Form. To overcome the drawbacks of hard constraint form, we convert prior knowledge into soft constraints by introducing slack variables:

- **Unary Bound**.

$$\hat{L} = \arg\max_{L} \langle L, P \rangle - M \sum_c (\xi_c^{(\nu)} + \xi_c^{(\mu)})$$
$$\text{s.t.} \quad \begin{cases} \sum_c L_{i,c} = 1 \quad \forall i \in [n_t] \\ L_{i,c} \in \{0,1\}, \quad \forall c \in \mathcal{C}, i \in [n_t] \\ \xi_c^{(\nu)} = \max\left(0, -\sum_i L_{i,c} + n_t \nu^{(c)}\right), \quad \forall c \in \mathcal{C} \\ \xi_c^{(\mu)} = \max\left(0, \sum_i L_{i,c} - n_t \mu^{(c)}\right), \quad \forall c \in \mathcal{C} \end{cases} \tag{4}$$

- **Binary Relationship**.

$$\hat{L} = \arg\max_{L} \langle L, P \rangle - M \sum_{c_1,c_2} \xi_{c_1,c_2}$$
$$\text{s.t.} \quad \begin{cases} \sum_c L_{i,c} = 1 \quad \forall i \in [n_t] \\ L_{i,c} \in \{0,1\}, \quad \forall c \in \mathcal{C}, i \in [n_t] \\ \xi_{c_1,c_2} = \max\left(0, -\sum_i (L_{i,c_1} - L_{i,c_2}) + n_t \delta^{(c_1,c_2)}\right), \quad \forall c_1, c_2 \in \mathcal{C} \end{cases} \tag{5}$$

In both Eq. 4 and Eq. 5, M is a pre-defined non-negative constant. When M is large enough, their solutions will be the same as those of Eq. 2 and Eq. 3 respectively, providing the hard constraints from prior knowledge are satisfiable. When $M = 0$, Eq. 4 and Eq. 5 will degenerate to the vanilla problem in Eq. 1.

Smooth Regularization. Previous optimization problems utilize prior knowledge about class distribution to refine pseudo labels. However, this solely relies on the model predicted probability matrix P without considering the data distribution in the feature space. In classification tasks, it is expected that the label prediction is locally smoothed. Hence, we add a smooth regularization that enforces the pseudo label of neighboring samples to be consistent.

We select a subset of target samples $\mathcal{S}_t \subseteq \mathcal{D}_t$ whose model predictions are uncertain. For each $\boldsymbol{x}_i^t \in \mathcal{S}_t$, let its nearest neighbor in $\mathcal{D}_t \backslash \mathcal{S}_t$ be $\boldsymbol{x}_{ki}^t$. The smooth regularization is a collection of equality constraints, $\mathcal{R} = \{(\boldsymbol{l}_i = \boldsymbol{l}_{ki}) | \boldsymbol{x}_i^t \in \mathcal{S}_t\}$. Converting these equalities into soft constraints is non-trivial as it will bring second-order terms in the objective. Instead, we directly add them as hard constraints to the optimization problem in Eq. 4 and Eq. 5.

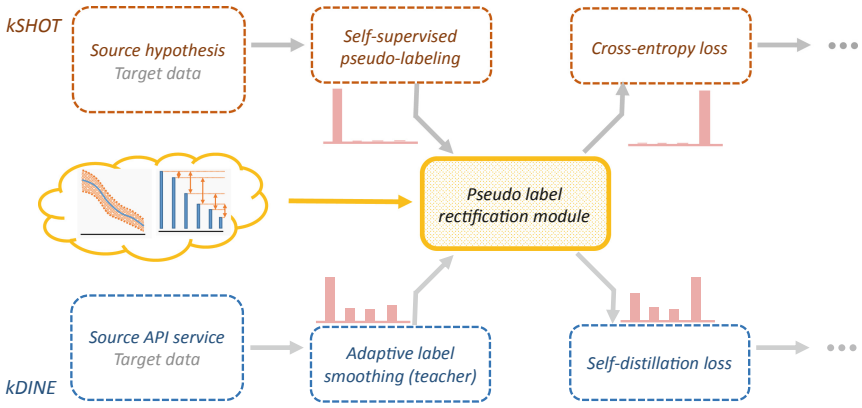

Fig. 2. Illustration of how our proposed rectification module is integrated into SHOT and DINE to get knowledge-guided SHOT and DINE. This can be easily extended to other self-training based UDA methods in a similar manner.

4.2 Knowledge-Guided UDA Methods

Our proposed rectification module is general, and can be easily plugged into self-training based UDA methods. To validate its effectiveness, we choose two recent UDA algorithms, SHOT [18] and DINE [19]. This leads to knowledge-guided SHOT and DINE, dubbed as kSHOT and kDINE, respectively. The frameworks are illustrated in Fig. 2. It should be noted that the main purpose of this part is to demonstrate the benefits of considering class prior knowledge through our rectification module, rather than simply extending the two algorithms.

Knowledge-Guide SHOT. SHOT [18] is a state-of-the-art self-training based UDA method. It assumes a source-data free setting, *i.e.*, only the source hypothesis is available during adaptation. Then both self-supervised pseudo-labeling and mutual information maximization are exploited to fine-tune the feature extractor module of the source hypothesis. Since only target samples are involved, it provides us a convenient platform to observe how prior knowledge affects the performance of adapted models. The full objective is

$$\mathcal{L}_{\text{shot}} = \mathbb{E}_{(\boldsymbol{x}_i^t, \hat{y}_i^t)} \ell_{\text{ce}}(h_t \circ g_t(\boldsymbol{x}_i^t), \hat{y}_i^t) - \alpha \mathcal{L}_{\text{im}} \tag{6}$$

where ℓ_{ce} is the cross entropy loss, $\mathcal{L}_{\text{im}}$ is the Information Maximization loss [6, 34], and α is a hyper-parameter. A critical step is to obtain the pseudo label $\hat{y}_i^t$. SHOT uses the distances between samples and class centroids in the feature space to refine model predictions. While this improves the quality of pseudo labels to some extent, their empirical distribution could still be very different from the ground-truth, as shown in Fig. 5. We show how prior knowledge can be incorporated to alleviate this issue.

We plug the rectification module into SHOT. After obtaining pseudo labels $L^{(\text{shot})}$ (*i.e.*, the one-hot representation of $\hat{y}^t$) and feature-to-centroid distances from SHOT, we convert the distances into class probabilities using `softmax` as

$$P = \texttt{softmax}(-D) \tag{7}$$

where $D \in \mathbb{R}^{n_t \times C}$, $D_{i,k} = d_f(g_t(\boldsymbol{x}_i^t), c_k)$, d_f is some distance metric and c_k is the k-th class centroid.

Let the optimal solution of the Zero-One Programming in Eq. 4 and Eq. 5 under prior knowledge $\mathcal{K}$ and smooth regularization $\mathcal{R}$ be

$$L^* = \mathfrak{S}(P, \mathcal{K}, \mathcal{R}) \tag{8}$$

$\mathfrak{S}(P, \emptyset, \emptyset)$ returns exactly the same pseudo labels as $L^{(\text{shot})}$. Given prior knowledge $\mathcal{K}$, we first obtain $L^{(\text{pk}_0)} = \mathfrak{S}(P, \mathcal{K}, \emptyset)$. Then we create a subset $\mathcal{S}_t = \{x_i^t | \boldsymbol{l}_i^{(\text{shot})} \neq \boldsymbol{l}_i^{(\text{pk}_0)}\}$ that consists of all samples whose pseudo label changed. After that the smooth regularization $\mathcal{R}$ can be constructed using $\mathcal{S}_t$. Finally, we obtain $L^{(\text{pk}_1)} = \mathfrak{S}(P, \mathcal{K}, \mathcal{R})$ and use $L^{(\text{pk}_1)}$ as $\hat{y}^t$ to update model with Eq. 6.

Knowledge-Guide DINE. DINE [19] is a very recent algorithm that assumes only black-box source models (*e.g.*, source API service) are available during adaptation. It first distills knowledge from the source predictor to a target model, and then fine-tunes the distilled model with target data. Two kinds of structural regularization, including interpolation consistency training [44] and mutual information maximization [34], are applied. The objective is

$$\mathcal{L}_{\text{dine}} = \mathbb{E}_{\boldsymbol{x}_i^t} \mathcal{D}_{\text{kl}} \left(P^{\text{tch}}(\boldsymbol{x}_i^t) \| f_t(\boldsymbol{x}_i^t) \right) + \beta \mathcal{L}_{\text{mix}} - \mathcal{L}_{\text{im}} \tag{9}$$

where $\mathcal{D}_{\text{kl}}$ denotes the Kullback-Leibler divergence, $\mathcal{L}_{\text{mix}}$ and $\mathcal{L}_{\text{im}}$ are two regularizers, and β is a trade-off parameter. To obtain the teacher prediction $P^{\text{tch}}(\boldsymbol{x}_i^t)$, the authors propose to revise the predictions of source model with adaptive label smoothing, and maintain an exponential moving average (EMA) prediction.

Given prior knowledge $\mathcal{K}$, we aim to rectify the teacher prediction $P^{\text{tch}}(\boldsymbol{x}_i^t)$ to be more compliant with the ground-truth. We adopt a similar strategy as in kSHOT. The pseudo labels of DINE are obtained by $\hat{y}_i^{(\text{dine})} = \arg\max P^{\text{tch}}(\boldsymbol{x}_i^t)$. Let the corresponding one-hot representation be $\boldsymbol{l}_i^{(\text{dine})}$. We take $P^{\text{tch}}(\boldsymbol{x}_i^t)$ as the i-th row of P, and obtain $L^{(\text{pk}_0)} = \mathfrak{S}(P, \mathcal{K}, \emptyset)$. Then we create a subset $\mathcal{S}^t = \{\boldsymbol{x}_i^t | \boldsymbol{l}_i^{(\text{dine})} \neq \boldsymbol{l}_i^{(\text{pk}_0)}\}$ to construct the smooth regularization $\mathcal{R}$. Finally, we obtain $L^{(\text{pk}_1)} = \mathfrak{S}(P, \mathcal{K}, \mathcal{R})$. The new objective function is

$$\mathcal{L}_{\text{kdine}} = \mathbb{E}_{\boldsymbol{x}_i^t} \mathcal{D}_{\text{kl}} \left(\frac{P^{\text{tch}}(\boldsymbol{x}_i^t) + \tilde{\boldsymbol{l}}_i^{(\text{pk}_1)}}{2} \middle\| f_t(\boldsymbol{x}_i^t) \right) + \beta \mathcal{L}_{\text{mix}} - \mathcal{L}_{\text{im}} \tag{10}$$

where $\tilde{\boldsymbol{l}}_i^{(\text{pk}_1)} = 0.9 \cdot \boldsymbol{l}_i^{(\text{pk}_1)} + 0.1/C$ is the smoothed label.

5 Experiments

5.1 Experimental Setup

Datasets. We report our results on both standard UDA benchmarks and benchmarks designed with label distribution shift. **Office-Home** is an image classification dataset with 65 classes from four environments: Artistic (A), Clip

Table 2. Classification accuracies (%) on **Office-Home RS-UT** and **Office**.

Method	$\mathcal{K}$	σ	Office-Home RS-UT							Office						
			R→P	R→C	P→R	P→C	C→R	C→P	Avg.	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
SHOT	–	–	77.0	50.3	75.9	47.0	64.3	64.6	63.2	94.0	90.1	74.7	98.4	74.3	**99.9**	88.6
kSHOT	UB	0.0	**78.8**	51.3	**79.1**	**49.8**	71.3	**69.7**	**66.6**	**97.6**	**98.5**	75.0	**99.0**	76.2	99.8	**91.0**
	UB	0.1	78.3	**51.7**	79.0	48.6	**71.6**	69.4	66.4	96.7	97.2	75.5	98.7	**76.5**	99.8	90.7
	UB	0.5	76.7	50.6	77.3	48.4	68.4	67.3	64.8	93.9	92.8	**75.7**	97.7	75.2	99.7	89.2
	UB	1.0	76.3	50.0	76.9	48.4	66.7	65.8	64.0	93.7	92.4	75.5	97.7	75.5	99.7	89.1
	UB	2.0	76.4	50.0	76.0	47.9	65.3	64.1	63.3	93.7	92.4	75.0	97.7	75.2	99.7	89.0
	BR	–	78.6	51.6	78.7	49.3	70.1	68.8	66.2	96.9	97.1	74.0	98.8	76.1	99.8	90.5

Art (C), Product (P), and Real-world (R). **Office** [31] contains 31 classes of office objects from three domains: Amazon (A), DSLR (D) and Webcam (W). **VisDA-2017** [27] is a large-scale Synthetic-to-Real dataset with 12 categories of objects. **Office-Home RS-UT** [36] is a subset of Office-Home created with Reverse-unbalanced Source and Unbalanced Target manner. Both source and target label distributions are long-tailed. The majority classes in source domain are minority ones in target domain. Hence, it has a big label distribution shift. **DomainNet** [26] is a large UDA benchmark. We use the subset [36] of 40-commonly seen classes from four domains: Clipart (C), Painting (P), Real (R), Sketch (S). It has a natural label distribution shift.

Creating Prior Knowledge. We create prior knowledge from ground-truth labels of target training data, $\{y_i^t\}_{i=0}^{n_t-1}$, for experimental purposes only. The noisiness and completeness of the prior knowledge are discussed in Sect. 5.3. Let $q_c = \sum_i \mathbb{I}[y_i^t = c]/n_t$ be the empirical probability of the c-th class.

- **Unary Bound**. We create UB as $\left\{(q_c \cdot (1-\sigma) \leq p_t^{(c)} \leq q_c \cdot (1+\sigma))|c \in \mathcal{C}\right\}$, where σ is hyper-parameter controlling the tightness of the bounds. In the experiments, we choose $\sigma \in \{0.0, 0.1, 0.5, 1.0, 2.0\}$.
- **Binary Relationship**. We first sort all classes based on q_c in descending order. Assuming the corresponding indexes are $[c_0^q, c_1^q, \cdots, c_{C-1}^q]$. Then we create BR as $\left\{(p_t^{(c_i^q)} - p_t^{(c_{i+1}^q)} \geq 0)|i \in \{0, 1, \cdots, C-2\}\right\}$. We simply take the right hand to be 0, which makes them relatively loose constraints and more easily available in practice.

Implementation Details. We solve the optimization problem of the rectification module with Gurobi Optimizer [7]. Both kSHOT and kDINE are based on the official Pytorch implementations by the authors. We use a pretrained ResNet-101 [8] backbone for VisDA-2017, and ResNet-50 [8] for others. To fairly compare with SHOT and DINE, we adopt the same hyper-parameters as used in the original papers. We run every task for 3 times and report the mean evaluation values. For standard UDA benchmarks, we report the accuracy. For benchmarks with label distribution shift (*i.e.*, Office-Home RS-UT and DomainNet), we report per-class average accuracy, in consistent with previous works [29,36].

Table 3. Classification accuracies (%) on **Office-Home** and **VisDA-2017**.

Method	$\mathcal{K}$	σ	Office-Home													VisDA
			A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	Avg.	
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	82.9
kSHOT	UB	0.0	**58.2**	**80.0**	82.9	**71.1**	**80.3**	**80.7**	71.3	**56.8**	**83.2**	75.5	60.3	**86.6**	**73.9**	**86.1**
	UB	0.1	58.1	79.2	**83.2**	70.4	80.0	**80.7**	**71.4**	56.5	83.0	**75.6**	**60.8**	86.0	73.7	85.8
	UB	0.5	57.4	79.1	82.1	69.4	78.1	79.5	69.3	55.2	81.8	74.0	60.2	85.1	72.6	83.9
	UB	1.0	57.0	79.0	82.1	68.6	77.8	79.3	68.4	55.1	81.7	73.5	59.3	84.8	72.2	83.0
	UB	2.0	56.4	78.7	82.1	68.3	77.8	79.3	67.9	54.2	81.7	73.3	58.7	84.8	71.9	82.6
	BR	–	57.4	78.8	82.9	70.7	80.0	80.5	70.8	55.0	82.8	74.6	59.9	86.0	73.3	83.6

Table 4. Classification accuracies (%) on **DomainNet**.

Method	$\mathcal{K}$	σ	R→C	R→P	R→S	C→R	C→P	C→S	P→R	P→C	P→S	S→R	S→C	S→P	Avg.
SHOT	–	–	79.4	75.4	72.8	88.4	74.0	75.5	89.8	77.7	76.2	88.3	**80.5**	70.8	79.1
kSHOT	UB	0	**83.6**	77.5	**75.3**	**91.5**	76.4	**77.0**	**91.7**	**82.3**	76.3	**89.7**	80.2	70.3	**81.0**
	UB	0.1	82.2	**77.6**	75.2	89.5	**76.8**	76.9	91.2	81.7	**76.9**	88.5	79.4	70.1	80.5
	UB	0.5	80.3	77.2	73.5	88.8	75.4	75.6	89.0	78.4	76.6	88.3	78.9	70.4	79.4
	UB	1.0	79.9	76.8	72.9	88.8	73.7	75.3	88.6	77.6	76.4	88.0	80.0	69.9	79.0
	UB	2.0	79.2	76.3	73.1	88.8	75.4	75.5	88.6	77.8	76.2	87.9	80.1	**71.1**	79.2
	BR	–	82.1	76.8	74.3	89.1	73.7	76.4	**91.7**	80.6	75.9	88.8	79.1	70.2	79.9

5.2 Results

Results of kSHOT. Tables 2, 4 list results of kSHOT on two benchmarks with label distribution shift. In UB($\sigma = 0$), it improves the accuracy by +3.4% on Office-Home RS-UT and +1.9% on DomainNet. As σ grows, the prior knowledge becomes less informative, and consequently the improvements reduce. Interestingly in BR where only the relative order of class probabilities is known, it still improves +3.0% on Office-Home RS-UT. Since this dataset is manually created to be long-tailed, and class distributions of two domains are reversed version of each other, having prior knowledge about target class distribution would be very helpful. This conforms to our experimental results. Results on three standard benchmarks are listed in Tables 2, 3. Using prior knowledge consistently improves on them. Similar trends on σ can be observed. Compared with previous benchmarks, the phenomenon of label distribution shift is less severe. Still prior knowledge can be helpful to correct mistaken pseudo labels during training.

Results of kDINE. Since using vanilla label smoothing is sub-optimal to the adaptive label smoothing used in DINE [19], the true performance gain from prior knowledge could be reduced. To make a fair comparison, we also provide results when replacing $\boldsymbol{l}_i^{(\mathrm{pk}_1)}$ with the $\boldsymbol{l}_i^{(\mathrm{dine})}$ in Eq. 10, and term it as DINE*. As can be seen in Tables 6, 7, DINE* indeed performs worse than DINE. Nevertheless, incorporated with prior knowledge, kDINE achieves much higher accuracy, and even better performance than DINE.

Table 5. Classification accuracies (%) on **Office-Home** for PDA.

Method	$\mathcal{K}$	σ	:A	:C	:P	:R	Avg.
SHOT	–	–	78.9	65.2	82.9	90.3	79.3
kSHOT	UB	0.0	**85.4**	**74.1**	**94.2**	**93.6**	**86.8**
	BR	–	84.9	72.3	90.2	92.2	84.9
DINE	–	–	77.6	59.2	82.7	85.2	76.2
DINE*	–	–	73.1	54.8	80.0	83.9	73.0
kDINE	UB	0.0	**82.1**	**66.4**	**91.3**	**91.7**	**82.9**
	BR	–	79.7	63.3	88.2	89.5	80.2

Table 6. Classification accuracies (%) on **Office**.

Method	$\mathcal{K}$	σ	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
DINE	–	–	91.6	86.8	**72.2**	96.2	**73.3**	98.6	86.4
DINE*	–	–	90.6	86.5	70.6	95.2	72.0	99.3	85.7
kDINE	UB	0.0	**94.7**	**92.2**	71.0	**96.8**	72.6	99.8	**87.9**
	UB	0.1	93.6	91.2	71.0	96.5	72.1	**99.5**	87.3
	UB	0.5	91.7	88.3	70.4	95.2	71.6	99.3	86.1
	UB	1.0	90.6	86.4	70.6	95.2	72.3	99.3	85.8
	UB	2.0	90.6	86.5	70.7	95.2	72.0	99.3	85.7
	BR	–	93.4	91.1	70.5	96.4	72.1	**99.5**	87.2

Table 7. Classification accuracies (%) on **Office-Home**.

Method	$\mathcal{K}$	σ	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	Avg.
DINE	–	–	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*	–	–	51.8	76.0	79.6	63.1	75.1	76.5	60.4	48.5	80.7	69.4	55.9	83.5	68.4
kDINE	UB	0.0	54.8	78.6	**81.7**	**67.1**	78.3	**79.6**	**66.8**	**52.3**	82.5	**72.0**	**58.1**	**85.4**	**71.4**
	UB	0.1	**55.0**	78.8	81.1	66.4	77.7	79.2	66.4	51.8	82.3	71.5	58.0	84.9	71.1
	UB	0.5	52.9	76.7	79.9	64.5	76.3	77.8	63.8	51.0	80.9	70.5	57.1	84.2	69.6
	UB	1.0	52.3	76.0	79.6	63.5	75.2	76.5	62.1	49.0	80.7	69.9	56.4	83.4	68.7
	UB	2.0	51.8	76.0	79.6	63.0	75.1	76.5	60.8	49.2	80.7	69.6	55.5	83.5	68.4
	BR	–	54.2	**79.4**	81.5	66.8	**78.6**	79.2	65.6	50.9	**82.6**	71.4	**58.1**	85.3	71.1

Results for PDA. We further evaluate on Office-Home for Partial-set DA (PDA), where there are totally 25 classes (the first 25 classes in alphabetical order) in the target domain and 65 classes in the source domain. This thus can be viewed as an extreme situation where class probabilities of the rest 40 classes are all zero. Table 5 lists the results averaged over tasks with the same target domain (*e.g.*, the :A column averages over C→A, P→A and R→A). As can be seen, using prior knowledge significantly improves in this situation.

5.3 Analysis

How prior knowledge guides UDA? To see why prior knowledge is helpful, we consider the following two aspects:

– **Ambiguous samples**. As illustrated Fig. 3 (left), our rectification module updates pseudo labels globally to match prior knowledge. This in effect moves decision boundaries in the feature space. The pseudo labels of ambiguous samples lying near the boundaries could be corrected during this process. Figure 3 (center) plots the accuracies of three set of pseudo labels, $L^{(\text{shot})}$, $L^{(\text{pk}_0)}$ and $L^{(\text{pk}_1)}$, during the training process of kSHOT with UB($\sigma = 0.0$) on Office A→W. Clearly using prior knowledge obtains more accurate pseudo labels. This in turn benefits the subsequent self-training stage.

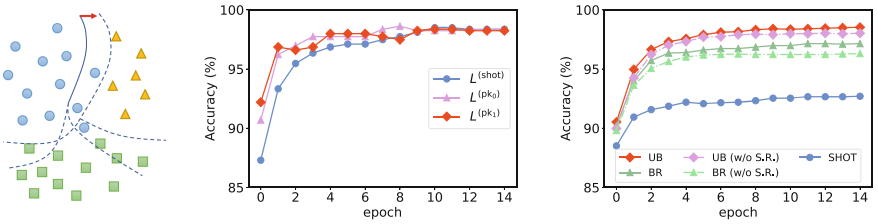

Fig. 3. (Left) Prior knowledge rectifies the pseudo label of ambiguous samples; (center) accuracies of pseudo labels before and after rectification using prior knowledge during the training of kSHOT on Office A→W; (right) convergence curves on Office A→W for comparison methods (S.R. is short for Smooth Regularization).

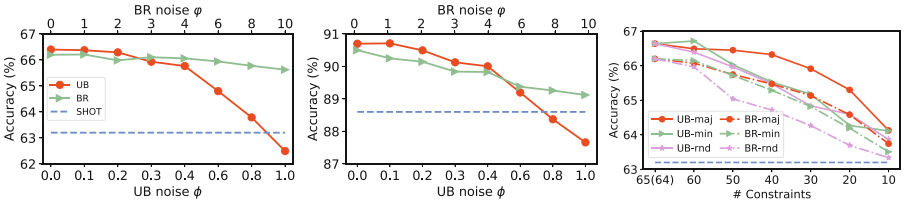

Fig. 4. Using noisy prior knowledge in kSHOT on (left) Office-Home RS-UT and (center) Office; (right) using partial prior knowledge in kSHOT on Office-Home RS-UT.

– **Label distribution**. Figure 5 plots distributions of ground-truth labels and pseudo labels on Office-Home RS-UT P→C in one seudo-labeling step. The accuracy of pseudo labels is $\sim 55\%$. As can be seen, the distribution of $L^{(\text{shot})}$ severely deviates from the ground-truth. In contrast, distributions of pseudo labels after rectification with prior knowledge are better compliant with the ground-truth. Figure 6 plots distributions of network predictions throughout the training process. The vanilla SHOT method hardly improves the label distribution after adaptation, whereas using prior knowledge drives it to be more similar to the ground-truth distribution in kSHOT.

Noisiness of the Prior Knowledge. In practice, the prior knowledge might contain some level of noises. To study its effects, we manually add noises to the estimated class prior q_c. For UB, uniform noises are added through $\tilde{q}_c = q_c + \mathcal{U}(-q_c\phi, q_c\phi)$, where $\phi \in [0, 1]$ controls the noise level. The noises have been centered to ensure that $\tilde{q}_c$ is a valid probability. Then $\tilde{q}_c$ is used to create the unary bound discussed in Sect. 5.1. For BR, after sorting all classes based on q_c, we randomly swap neighboring classes. Suppose the index of class c_k in the sorted order is I_{c_k}, uniform noises are added through $\tilde{I}_{c_k} = I_{c_k} + \mathcal{U}(-\varphi, \varphi)$, where φ controls the neighborhood size. Then we sort all classes based on the noisy indexes $\tilde{I}_{c_k}$, and use the resorted order to create binary relationship. Figure 4 (left and center) shows that under moderate noises, incorporating prior knowledge is still helpful and improves over the SHOT baseline.

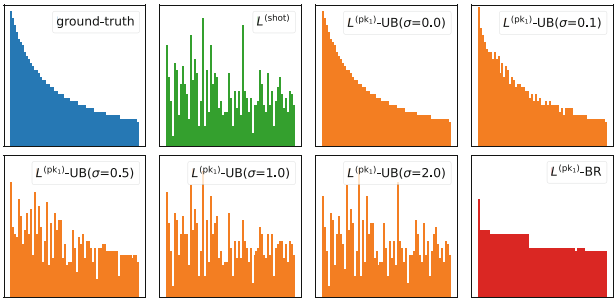

Fig. 5. Prior knowledge rectifies distribution of pseudo labels in one pseudo-labeling step of kSHOT (Office-Home RS-UT P→C).

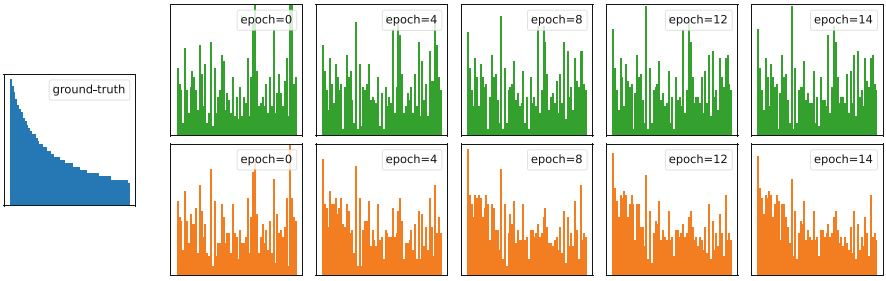

Fig. 6. Label distributions of network (f_t) predictions in SHOT (upper) and kSHOT (lower) (Office-Home RS-UT P→C).

Completeness of the Prior Knowledge. Until then, the prior constraints are assumed to cover every class. It is straightforward to generalize to partial constraints. We randomly select a portion of constraints that corresponds to the major (maj.), minor (min) or random (rnd) classes. Figure 4 (right) present the results under different number of selected constraints. Partial constrains imply less prior information, but still can benefit UDA training. Note that BR-rnd reduces the performance most, partially because the randomly selected binary relationship constraints hardly form the complete order of a subset of classes.

Estimating Class Prior from Partial Data. Table 9 presents the experimental results when estimating q_c from partial target data in kSHOT with UB($\sigma = 0.0$) on VisDA-2017. As sampling ratio reduces, the estimation error (relative deviations from using full data) increases. Nevertheless, even when the average estimation error is about 15.5% at a sampling ratio of 0.5%, using prior knowledge still improves over SHOT by +1.16% (82.9% → 84.06%).

Effects of Smooth Regularization. Table 8 presents the ablation study on smooth regularization. Generally, using smooth regularization achieves comparable or better performance. The penalty depends on $|\mathcal{S}_t| = \sum_i \mathbb{I}[\boldsymbol{l}_i^{(\text{shot})} \neq \boldsymbol{l}_i^{(\text{pk}_0)}]$, and varies across different tasks. In UB with large σ, the prior knowledge is not

Table 8. Ablating Smooth Regularization (S.R.) in kSHOT on Office-Home RS-UT.

	$\mathcal{K}$	σ	R→P	R→C	P→R	P→C	C→R	C→P	Avg.
wo/ S.R.	UB	0.0	77.9	52.0	78.8	50.0	70.3	68.9	66.3
	UB	0.1	77.9	51.4	78.7	49.7	70.5	68.4	66.1
	UB	0.5	76.6	50.5	77.0	48.4	67.7	67.3	64.6
	UB	1.0	76.5	50.1	76.7	48.1	66.4	65.3	63.8
	UB	2.0	76.4	49.9	75.8	47.8	65.1	64.2	63.2
	BR	–	77.9	51.2	78.1	49.2	69.0	67.6	65.5
w/ S.R.	UB	0.0	78.8	51.3	79.1	49.8	71.3	69.7	66.6
	UB	0.1	78.3	51.7	79.0	48.6	71.6	69.4	66.4
	UB	0.5	76.7	50.6	77.3	48.4	68.4	67.3	64.8
	UB	1.0	76.3	50.0	76.9	48.4	66.7	65.8	64.0
	UB	2.0	76.4	50.0	76.0	47.9	65.3	64.1	63.3
	BR	–	78.6	51.6	78.7	49.3	70.1	68.8	66.2

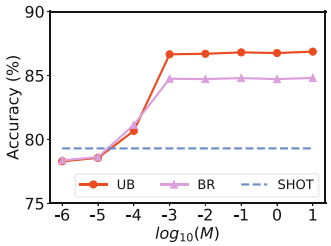

Fig. 7. Effects of constant M (by a scalar n_t) in kSHOT on Office-Home for PDA (averaged over 12 tasks).

Table 9. Estimating class prior from partial target data in kSHOT on VisDA-2017.

Sampling ratio	0.5%	1%	5%	10%	50%	100%
Max. est. err (%)	47.5	25.2	11.2	9.6	4.1	0.0
Avg. est. err (%)	15.5	12.0	5.4	3.4	1.9	0.0
Avg. acc (%)	84.06	84.76	85.77	85.93	86.12	86.13

informative, hence $|\mathcal{S}_t|$ is small. We underline cases where using smooth regularization increases the accuracy significantly. The amount of improvement is most significant on tasks like R→P, C→R, C→P. Comparing two types of prior knowledge, it is more helpful in BR. Since BR only tells the order of class probabilities, adding smooth regularization provides complementary information.

Choice of Constant M. The prior knowledge is considered to be reliable, hence we expect the prior constraints to be satisfied in the rectified pseudo labels. To achieve this, M need to be some large constant in Eq. 4, 5. Figure 7 shows how M affects the performance on Office-Home under PDA setting in kSHOT. When M is very small, the method degenerates to SHOT. When M is larger than some threshold (*e.g.*, $10^{-3} \cdot n_t$), all soft constraints will in fact be satisfied. In our experiments, we use $M = 10 \cdot n_t$ for all tasks in both kSHOT and kDINE.

6 Conclusions

We present a new yet realistic setting termed Knowledge-guided Unsupervised Domain Adaptation (KUDA). In KUDA, in addition to labeled source data and unlabeled target data, we have access to some prior knowledge about target label distribution. We present a novel rectification module that refines pseudo labels using prior knowledge through solving a constrained optimization problem. Then we integrate it into two representative self-training based methods, SHOT and

DINE. Extensive experiments show that using prior knowledge can significantly improve the performance. We expect our work to inspire further investigations along the direction.

References

1. Baslamisli, A.S., Le, H.A., Gevers, T.: CNN based learning using reflection and retinex models for intrinsic image decomposition. In: CVPR, pp. 6674–6683 (2018)
2. Chen, D., Wang, D., Darrell, T., Ebrahimi, S.: Contrastive test-time adaptation. In: CVPR, pp. 295–305 (2022)
3. Chen, L., et al.: Reusing the task-specific classifier as a discriminator: discriminator-free adversarial domain adaptation. In: CVPR, pp. 7181–7190 (2022)
4. Ding, N., Xu, Y., Tang, Y., Xu, C., Wang, Y., Tao, D.: Source-free domain adaptation via distribution estimation. In: CVPR, pp. 7212–7222 (2022)
5. Fu, B., Cao, Z., Wang, J., Long, M.: Transferable query selection for active domain adaptation. In: CVPR, pp. 7272–7281 (2021)
6. Gomes, R., Krause, A., Perona, P.: Discriminative clustering by regularized information maximization. In: NeurIPS (2010)
7. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022). https://www.gurobi.com
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
9. Huang, J., Guan, D., Xiao, A., Lu, S.: Model adaptation: historical contrastive learning for unsupervised domain adaptation without source data. In: NeurIPS (2021)
10. Kayhan, O.S., Gemert, J.C.V.: On translation invariance in CNNs: convolutional layers can exploit absolute spatial location. In: CVPR, pp. 14274–14285 (2020)
11. Kim, T., Kim, C.: Attract, perturb, and explore: learning a feature alignment network for semi-supervised domain adaptation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12359, pp. 591–607. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58568-6_35
12. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
13. LeCun, Y., et al.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
14. Lefort, R., Fablet, R., Boucher, J.-M.: Weakly supervised classification of objects in images using soft random forests. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 185–198. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15561-1_14
15. Lengyel, A., Garg, S., Milford, M., van Gemert, J.C.: Zero-shot day-night domain adaptation with a physics prior. In: ICCV, pp. 4399–4409 (2021)
16. Li, B., Wang, Y., Zhang, S., Li, D., Keutzer, K., Darrell, T., Zhao, H.: Learning invariant representations and risks for semi-supervised domain adaptation. In: CVPR, pp. 1104–1113 (2021)
17. Li, R., Cheong, L.F., Tan, R.T.: Heavy rain image restoration: Integrating physics model and conditional adversarial learning. In: CVPR, pp. 1633–1642 (2019)
18. Liang, J., Hu, D., Feng, J.: Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In: ICML, pp. 6028–6039 (2020)

19. Liang, J., Hu, D., Feng, J., He, R.: Dine: domain adaptation from single and multiple black-box predictors. In: CVPR, pp. 8003–8013 (2022)
20. Lin, Y., Pintea, S.L., van Gemert, J.C.: Deep Hough-transform line priors. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12367, pp. 323–340. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58542-6_20
21. Lipton, Z., Wang, Y.X., Smola, A.: Detecting and correcting for label shift with black box predictors. In: ICML, pp. 3122–3130 (2018)
22. Liu, X., et al.: Adversarial unsupervised domain adaptation with conditional and label shift: infer, align and iterate. In: ICCV, pp. 10367–10376 (2021)
23. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR, pp. 3431–3440 (2015)
24. Mann, G.S., McCallum, A.: Simple, robust, scalable semi-supervised learning via expectation regularization. In: ICML, pp. 593–600 (2007)
25. Pan, S.J., Yang, Q.: A survey on transfer learning. TKDE **22**(10), 1345–1359 (2009)
26. Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., Wang, B.: Moment matching for multi-source domain adaptation. In: ICCV, pp. 1406–1415 (2019)
27. Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., Saenko, K.: Visda: the visual domain adaptation challenge. arXiv preprint arXiv:1710.06924 (2017)
28. Prabhu, V., Chandrasekaran, A., Saenko, K., Hoffman, J.: Active domain adaptation via clustering uncertainty-weighted embeddings. In: ICCV, pp. 8505–8514 (2021)
29. Prabhu, V., Khare, S., Kartik, D., Hoffman, J.: Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. In: CVPR. pp. 8558–8567 (2021)
30. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NeurIPS (2015)
31. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 213–226. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15561-1_16
32. Saito, K., Kim, D., Sclaroff, S., Darrell, T., Saenko, K.: Semi-supervised domain adaptation via minimax entropy. In: ICCV, pp. 8050–8058 (2019)
33. Schapire, R.E., Rochery, M., Rahim, M., Gupta, N.: Incorporating prior knowledge into boosting. In: ICML, pp. 538–545 (2002)
34. Shi, Y., Sha, F.: Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In: ICML (2012)
35. Sun, T., Lu, C., Zhang, T., Ling, H.: Safe self-refinement for transformer-based domain adaptation. In: CVPR, pp. 7191–7200 (2022)
36. Tan, S., Peng, X., Saenko, K.: Class-imbalanced domain adaptation: an empirical odyssey. In: Bartoli, A., Fusiello, A. (eds.) ECCV 2020. LNCS, vol. 12535, pp. 585–602. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-66415-2_38
37. Urban, G., et al.: Do deep convolutional nets really need to be deep and convolutional? In: ICLR (2017)
38. Wang, M., Deng, W.: Deep visual domain adaptation: a survey. Neurocomputing **312**, 135–153 (2018)
39. Wang, Z., Lyu, S., Schalk, G., Ji, Q.: Learning with target prior. In: NeurIPS (2012)
40. Wilson, G., Cook, D.J.: A survey of unsupervised deep domain adaptation. ACM TIST **11**(5), 1–46 (2020)

41. Wolsey, L.A.: Integer Programming. Wiley, New York (2020)
42. Xie, B., Yuan, L., Li, S., Liu, C.H., Cheng, X., Wang, G.: Active learning for domain adaptation: an energy-based approach. In: AAAI, pp. 8708–8716 (2022)
43. Zhang, H., Zhang, Y., Jia, K., Zhang, L.: Unsupervised domain adaptation of black-box source models. arXiv preprint arXiv:2101.02839 (2021)
44. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: beyond empirical risk minimization. In: ICLR (2018)
45. Zhang, Y., Liu, T., Long, M., Jordan, M.: Bridging theory and algorithm for domain adaptation. In: ICML, pp. 7404–7413 (2019)
46. Zhu, X., Ghahramani, Z., Lafferty, J.D.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML, pp. 912–919 (2003)