Divergence-agnostic Unsupervised Domain Adaptation by Adversarial Attacks

Jingjing Li, *Member, IEEE,* Zhekai Du, Lei Zhu, *Member, IEEE,* Zhengming Ding, *Member, IEEE,* Ke Lu, and Heng Tao Shen, *Senior Member, IEEE, Fellow, ACM*

Abstract—Conventional machine learning algorithms suffer the problem that the model trained on existing data fails to generalize well to the data sampled from other distributions. To tackle this issue, unsupervised domain adaptation (UDA) transfers the knowledge learned from a well-labeled source domain to a different but related target domain where labeled data is unavailable. The majority of existing UDA methods assume that data from the source domain and the target domain are available and complete during training. Thus, the divergence between the two domains can be formulated and minimized. In this paper, we consider a more practical yet challenging UDA setting where either the source domain data or the target domain data are unknown. Conventional UDA methods would fail this setting since the domain divergence is agnostic due to the absence of the source data or the target data. Technically, we investigate UDA from a novel view — adversarial attack — and tackle the divergence-agnostic adaptive learning problem in a unified framework. Specifically, we first report the motivation of our approach by investigating the inherent relationship between UDA and adversarial attacks. Then we elaborately design adversarial examples to attack the training model and harness these adversarial examples. We argue that the generalization ability of the model would be significantly improved if it can defend against our attack, so as to improve the performance on the target domain. Theoretically, we analyze the generalization bound for our method based on domain adaptation theories. Extensive experimental results on multiple UDA benchmarks under conventional, source-absent and target-absent UDA settings verify that our method is able to achieve a favorable performance compared with previous ones. Notably, this work extends the scope of both domain adaptation and adversarial attack, and expected to inspire more ideas in the community.

Index Terms—Unsupervised domain adaptation, transfer learning, adversarial attacks, domain generalization, model adaptation.

1 Introduction

Deep neural networks that trained on massive amount of labeled data continue to be the most promising method for a variety of machine learning applications and problems. Although these methods have achieved remarkable success, they work under the assumption that the training set and the test set should be independent and identically distributed (i.i.d.). Unfortunately, this assumption hardly holds in many real-world applications due to various factors such as illumination, resolution and corruption, etc. As a result, the model trained on the source domain will suffer significant performance degradation when tested on novel samples from other domains with different data distributions, which is known as domain shift [1]. To alleviate this problem, a rich line of unsupervised domain adaptation (UDA) methods have been proposed to learn a model that can generalize well to the target domain with labeled source data and unlabeled target data [2], [3], [4].

Most existing UDA methods are designed to learn domain-invariant features by minimizing the domain divergence, which can be measured by giving the sampled data from both domains. This line of works can be roughly grouped into two categories: metric learning [5] and adversarial learning [6]. Specifically, the methods in the first category use various distance metrics (e.g., MMD) to quantify distribution discrepancy and then optimize these metrics to align different domains. Adversarial learning methods apply a domain discriminator to distinguish whether a sample comes from the source or the target domain, and encourage the feature generation network to extract domain-invariant representations via a two-player mini-max game.

Despite the great progress in UDA, existing methods, either metric learning-based ones or adversarial learning-based ones, all require labeled source data and unlabeled target data during training. However, this assumption can be too ideal in practical scenarios for two reasons: (1) The source domain data are not always available in many real-world applications, due to the data privacy issue, e.g., medical data, decentralization issue, e.g., federal learning, and computation resource restrictions, etc. (2) In some real-world scenarios, we cannot access the target domain data in advance due to dynamically changing environments or the rareness of target samples. For instance, changes in illumination, background or shooting angle will lead to a different target distribution. However, we cannot identify these factors in advance until the model is deployed, making it intractable to collect target samples for training. Conventional UDA methods cannot handle these scenarios since the domain divergence is agnostic. In this regard, there is a strong incentive to develop UDA methods that can handle these domain divergence-agnostic scenarios.

In this paper, we investigate a more general and practical UDA setting where the source data or the target data can be unknown during learning. Conventionally, we can refer to these two cases as source-free unsupervised domain adaptation (SF-UDA) [7], [8] and domain generalization (DG) [9], [10], respectively, which have been studied separately in

- *Jingjing Li, Zhekai Du, Ke Lu and Heng Tao Shen are with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Lei Zhu is with Shandong Normal University, Zhengming Ding is with Tulane University.*
  E-mail: lijin117@yeah.net

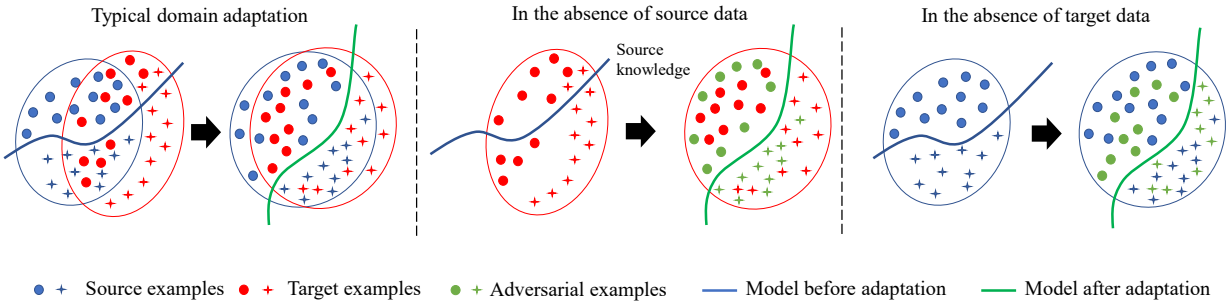


Fig. 1. A comparison among typical domain adaptation (left), source data-absent domain adaptation (middle) and target data-absent domain adaptation (right). Typical domain adaptation methods fail to handle the source or target data-absent cases. Our method explicitly explores unseen distributions related to the source domain(s) to help the model generate to the target domain.

the literature. In this work, we regard each of them as a special case in UDA and try to develop a general UDA framework that can handle these cases as well as conventional UDA simultaneously. Specifically, we introduce adversarial attack [11] into domain adaptation and prove that attack-proof models have favorable effects on improving the generalization ability of transfer learning methods. At first, let us revisit the essence of UDA. In UDA, the target samples have the same semantic information as the source samples yet with different data distributions. The model trained on the source domain tends to mis-classify the target samples due to the violation of the i.i.d. assumption. On the other hand, adversarial examples refer to a set of perturbed inputs that are almost indistinguishable from original data and yet mis-classified by the network. The common principle behind target samples and adversarial samples is that they both take advantage of the i.i.d. assumption. From this point, all mis-classified samples in the target domain can be regarded as visually limitless, naturally occurring adversarial samples. Intuitively, when we tackle UDA, we are essentially fighting with a generator which produces the domain perturbation to attack the source model. Thus we argue exploring the unseen distribution which is related with the source domain helps improve the generalization ability on the target domain, and this directly motivate our idea — UDA by adversarial attacks. More specifically, we propose a generative framework which is supposed to generate smooth and diverse adversarial samples to explore the unseen distribution, then the model is expected to harness these adversarial samples to improve the robustness and the generalization ability, thus boosting the performance on the target domain, as shown in Fig. 1. The tricky part is how to generate adversarial samples. On the one hand, directly deploying existing adversarial examples is not optimal for UDA since our goal is diversity rather pure attacks. On the other hand, adding random Gaussian noises [12] to add diversity is prone to deteriorate the original information thus also does not meet our purpose. Based on the observation that generating adversarial examples via neural networks is promising to guarantee the diversity and generalization [13] due to the randomness and generalization of deep neural networks [14], [15], we employ a neural network to generate perturbations. Specifically, the smoothness refers to a sample and the samples around it should belong to the same category, thus extending the classification boundaries and enabling a more robust classifier for the target samples, which can be controlled by adding a regularization term when generating perturbations, through which the semantic information could be preserved.

Moreover, it is worth noting that our framework can tolerate the absence of either source or target domain data, since generating adversarial examples only requires to access the data from one domain and the output of the training model (even no strict requirement for ground-truth labels). We will explain how our method deal with these divergence-agnostic UDA settings in Section 3. The main contributions of this work are summarized as follows.

- We consider a more general and practical domain adaptation problem, where either the source data or the target data can be absent during learning, and propose a new perspective to handle this general UDA problem by adversarial attacks. To the best of our knowledge, adversarial attack is widely investigated in terms of security and defense. This work is expected to inspire new ideas to extend the scope of both adversarial attack and domain adaptation.
- We propose a framework that produces diverse and smooth adversarial examples to enhance the generalization ability of the training model, which helps the source model generalize to the target domain. Several auxiliary regularization items are employed to further improve the adaptation performance.
- Extensive experiments over multiple UDA benchmarks show that our method achieves superior or comparable results over current state-of-the-art methods in both source data-absent and target data-absent UDA scenarios. This further demonstrates the versatility of our proposed method.

2 Related Work

2.1 Unsupervised Domain Adaptation

Typical unsupervised domain adaptation. The essence of unsupervised domain adaptation is to challenge the domain shift problem [16], which violates the i.i.d. assumption in machine learning. The early works of UDA use the instance-based adaptation strategy [17], [18]. These methods assume that a part of source data can be reused for learning the target model and reduce the discrepancy by inferring the resampling weight of the source instances in a non-parametric way. More recent works use feature-based strategies to align feature distributions between domains by learning a shared representation space [19], [20], [21]. As

deep learning becomes more prevalent, these feature-based methods have gained prominence in various visual tasks like image classification [22], face recognition [23], semantic segmentation [24] and object detection [25] thanks to the powerful feature extraction ability of deep neural networks (DNN) [26].

Current DNN based UDA methods pursue distribution alignment by learning invariant representations between domains, these approaches can be roughly grouped into two categories: metric learning-based adaptation and adversarial learning-based adaptation. Metric learning-based methods try to mitigate the distribution gap directly by minimizing a statistic criterion [27], [28]. Among them, maximum mean discrepancy (MMD) [28] is a widely used criterion which measures the mean discrepancy between two domains in the unit ball of a reproducing kernel Hilbert space (RKHS). For instance, deep adaptation networks (DAN) [5] align the distributions by minimizing the multi-kernel MMD between two domains and the accuracy error on source samples simultaneously. Joint Adaptation Networks (JAN) [29] extends DAN by using a joint maximum mean discrepancy to align the cross-domain distributions. Furthermore, other MMD variants [30] are also used in domain adaptation for a better measure of domain discrepancy. On the other hand, adversarial learning-based methods [6], [31] leverage an additional domain discriminator to distinguish the domain label for a sample, the feature extraction network is encouraged to minimize the Proxy $\mathcal{A}$-distance or $\mathcal{H}$-divergence between domains by fooling the domain discriminator in a GAN-based [32] framework. The most representative work is Domain Adversarial Neural Network (DANN) [6]. Instead of aligning marginal distributions, cycle-consistent conditional adversarial transfer networks [33] uses a cycle-consistent loss and a domain discriminator conditioned on the classifier prediction for a category-conditioned alignment. In addition, other variants of GAN such as CoGAN [34], CycleGAN [35] and DualGAN [36] also have been successfully applied to DA scenarios. Nevertheless, all the above methods require both source data and target data to be accessible during adaptation, thus fails in domain divergence-agnostic UDA scenarios.

UDA without source data. A recent branch of UDA assumes the absence of source data during training due to some realistic reasons, which can be referred to as source-free unsupervised domain adaptation (SFUDA) [37] or model adaptation [7]. These methods aim to improve the model performance on the target domain based on unlabeled target samples and a pre-trained source model. As an early work, USFDA [38] was developed to tackle open-set and partial UDA without access to source data. However, it requires the pre-training model to be equipped with a specifically designed module, which is unrealistic in many scenarios. For close-set SFUDA, existing methods can be roughly divided into two categories, one is self-training-based methods, another is generation-based methods. For the first category, SHOT [8] fine-tunes the source-trained model to the target domain based on information maximization. Meanwhile, pseudo labels obtained by weighted deep clustering are used for self-supervision. Analogously, PrDA [39] fine-tunes a pre-trained source model to the target domain by iteratively performing prototype updating and pseudo labeling based on prediction entropy. BAIT [40] leverages the idea of diverse classifiers in typical UDA community and makes it applicable for SFDA via entropy mini-max. SFDA [37] proposes to use the samples with low entropy to refine the pseudo labels for training. MCS [41] discover the pseudo labels of target data based on the statistical properties per-class calculated on source features. As for generation-based methods, 3C-GAN [7] generates target-style samples and leverages other regularizations to moderately retrain the source model. Hou et al. [42] convert the target-style images to the source-style based on the mean and variance stored in BN layers in the source model. Different from these models, we tackle this problem from a new perspective, i.e., adversarial attacks, which extends the scope of the seen distribution to improve the generation ability of the model.

UDA without target data is conventionally referred to as domain generalization (DG) [9] in the literature. The goal of DG is to exploit the knowledge from multiple seen source domains to learn a model that performs well on any unseen target domain. The research line of existing can be roughly divided into three categories: classifier-based [43], [44], feature-based [45], [46] and augmentation-based [47], [48] methods. Classifier-based methods usually train a domain-specific classifier for each source domain, then the sub-classifiers are combined into a classifier to realize domain generalization. Feature-based methods aim to learn the domain-invariant representations across source domains. Among these methods, Li *et al.* [45] develop an end-to-end learning framework for DG via a low-rank parameterized CNN model. Li *et al.* [49] resort to MMD constrained adversarial autoencoders to learn an aligned distribution across source domains, and match it to an arbitrary prior distribution via adversarial learning. Matsuura *et al.* [50] propose a method which divides samples into multiple latent domains via clustering, and train the domain-invariant feature extractor via adversarial learning. Augmentation-based methods usually augment source domains through adversarial perturbation and self-supervised learning. For instance, Shankar *et al.* [47] presented a domain perturbation strategy to perturb the input data based on the gradient. And Volpi *et al.* [48] generated the adversarial perturbations according to fictitious target distributions within a certain Wasserstein distance from the source. Recently, some methods also resort to self-supervised learning, i.e., various auxiliary tasks such as jigsaw puzzles [51], [52] and self-challenging [53], to improve generalization ability. We argue that existing methods in the first two categories are not robust and limited to the seen distributions since they focus on the invariant representation across source domains. Our method belongs to the augmentation-based ones. However, different from [47], [48], we explore the unseen target domain explicitly through a neural network.

2.2 Adversarial Attacks

Adversarial attacks aim to deliberately generate adversarial examples which are mis-classified by a DNN model via adding small magnitude perturbations to original inputs, which is first introduced by Szegedy *et al.* [54]. To mitigate this pitfall of DNN, many researchers suggest generating

adversarial examples and incorporating them into the training process to make the model resistant to these small perturbations, which is called adversarial training [11], [55], [56]. Adversarial training is closely related to our work since the main goal of both is to learn a model robust to perturbations in the input. Different from adversarial training which targets at imperceptibly perturbed samples, we focus on the samples with larger perturbations, which are out of source distributions.

Recently, a number of attack methods to generate adversarial examples have been proposed. As an early work, Goodfellow *et al.* propose two typical gradient-based attack methods called Fast Gradient Sign Method (FGSM) and Fast Gradient Method [11], which generate perturbations based on the gradient of the cost function with respect to the input data. Later, to solve the linear hypothesis problem in FGSM and FGM, Madry *et al.* [12] propose Projected Gradient descent (PGD) to achieve a stronger adversary, which is a very classic method and set a strong basis for many recent algorithms [57], [58]. Recently, several GAN-based methods have been proposed to generate adversarial examples by a feed-forward network rather than an optimization procedure [59], [60]. These methods empirically achieve higher attack success rate and generate more diverse adversarial examples [13].

3 METHOD

In this paper, we propose to address domain Adaptation by Adversarial Attacks (AAA). Specifically, we challenge the problem where the source data or the target data can be absent when learning the model. We aim to train a discriminative and robust model by generating adversarial examples and harnessing them. In this section, we first introduce our approach in the conventional UDA setting and then extend it to source data-absent and target data-absent UDA protocols.

3.1 Notations and Definitions

Let $\mathcal{X}$, $\mathcal{Z}$, $\mathcal{Y}$ be the input, feature and label spaces, respectively. In conventional UDA, we have a source domain with the joint distribution $\mathcal{D}_s$ supported by n_s labeled source samples $\left\{x_s^i, y_s^i\right\}_{i=1}^{n_s}$, and a target domain with n_t unlabeled target samples $\left\{x_t^i\right\}_{i=1}^{n_t}$ from the joint distribution $\mathcal{D}_t$. $\mathcal{D}_s$ and $\mathcal{D}_t$ are defined on $\mathcal{X} \times \mathcal{Y}$ and share the label space $\mathcal{Y}$ which consists of M discrete labels $\{1, 2, 3, \ldots, M\}$. The goal of UDA is to minimize the discrepancy between the source and target features coupled with the source risk so that the model trained by the source domain can also predict the labels $\left\{y_t^i\right\}_{i=1}^{n_t}$ of the samples in the target domain accurately. To this end, we formulate our model into a feature extractor $F : \mathcal{X} \to \mathcal{Z}$ that maps the input to the feature space, and a classifier $C : \mathcal{Z} \to \mathcal{Y}$ which outputs final predictions based on the features. In addition, our framework introduces a generator G conditioned on randomly sampled inputs for producing diverse and smooth perturbations. By incorporating these augmented examples for training, our model is prone to be smoother at each data point [61], which is generally preferred for robust classification. For clarity, G, F and C are parameterized by θ_G, θ_F and θ_C, respectively.

3.2 Domain Adaptation by Adversarial Training

The main idea of adversarial training is that the prediction for an input sample should be resistant to small perturbations. It is commonly accepted that adversarial training can improve the robustness for deep models [62]. Recently, a notable research in [63] claims that a robust model obtained by adversarial training tends to transfer better to other domains based on the observation that robust model is more likely to comprehend human-aligned features (e.g., adversarial training is like a natural learning paradigm of humans since they often learn a concept through many similar-looking examples). Alternatively, Shafahi *et al.* [64] suggested *robust transfer learning* and pointed out that robust feature extractors can be preserved in robust networks for different domains. Meanwhile, the accuracy on clean samples can also be preserved. Thus there is a great potential to leverage the idea of adversarial training to the field of domain adaptation, where the target distribution $\mathcal{D}_t$ takes the form of an unseen black-box adversary. In this paper, we consider the general principle of adversarial training which yields the following worst-case problem around the training distribution $\mathcal{P}_0$:

$$\min_{\theta_f} \sup_{\mathcal{P}: D(\mathcal{P}, \mathcal{P}_0) \leq \epsilon} \mathbb{E}_{(x^i, y^i) \sim \mathcal{P}}[\ell\left(f\left(x^i\right), y^i\right)], \tag{1}$$

where $f = F \circ C$ is the neural network, $D\left(\mathcal{P}, \mathcal{P}_0\right)$ is a distance metric, $(x^i, y^i) \sim \mathcal{P}$ is the target data when source data is absent and vice versa, ℓ represents the loss function (cross-entropy loss in this paper). Recently, ℓ_∞-bounded attracts got a lot of attention [65]. In this paper, we use ℓ_∞-bounded distance as an implement of D. For each data point x, the ℓ_∞-bound can be defined as

$$\mathcal{B}_\epsilon[x] = \left\{x' \mid d_{inf}\left(x, x'\right)_\infty \leq \epsilon\right\}, \tag{2}$$

where $d_{inf}\left(x, x'\right) = \left\|x - x'\right\|_\infty$ and $\epsilon > 0$ is the perturbation magnitude. The adversarial example $\widetilde{x}$ can be constructed by $\widetilde{x} = x + \epsilon\left(G\left(x\right) / \left\|G\left(x\right)\right\|_2\right)$. Once we solved the worst-case problem (1), it is safe to say that our model could achieve a robust performance on distributions that are ϵ away from the original distribution $\mathcal{P}_0$.

Generating Adversarial Examples. Since the purpose of adversarial examples is to confuse the classification model f, we encourage the generator learning towards the following goal:

$$\max_{\theta_G} \mathcal{L}_{adv} = \frac{1}{n_s} \sum_{i=1}^{n_s} \ell\left(f\left(x_s^i + \epsilon\left(G\left(x_s^i\right) / \left\|G\left(x_s^i\right)\right\|_2\right)\right), y_s^i\right). \tag{3}$$

In addition, we control the magnitude of the perturbations, so that target distributions satisfying $D\left(\mathcal{P}, \mathcal{P}_0\right) \leq \epsilon$ can be regard as realistic covariate shifts that preserve the original semantic information of the source data. Specifically, we add the following regularization term,

$$\min_{\theta_G} \mathcal{L}_{reg} = \frac{1}{n_s} \sum_{i=1}^{n_s} \left\|G\left(x_s^i\right)\right\|_2. \tag{4}$$

Then we define the following learning objective for G,

$$\min_{\theta_G} \mathcal{L}_{gen} = -\mathcal{L}_{adv} + \lambda_{reg} \mathcal{L}_{reg} \tag{5}$$

where λ_{reg} is a trade-off hyper-parameter.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3109287, IEEE Transactions on Pattern Analysis and Machine Intelligence

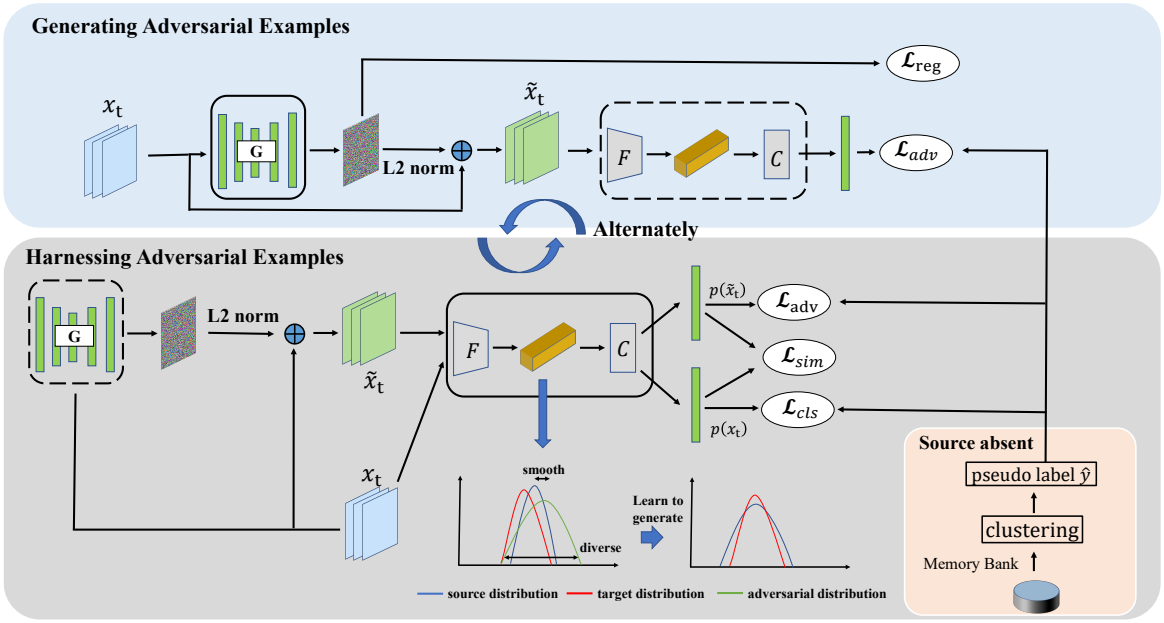


Fig. 2. The framework of our proposed method. Our framework consists of two alternative steps: generating adversarial examples and harnessing adversarial examples. In the first step, we employ an generator G to generate smooth and diverse perturbations for training samples x_t (we use target data when source data is absent and vice versa). In the second step, we learn from the generated examples to generalize better to the target domain. Some additional regularizations are used to stabilize the training. These two steps are performed alternatively until convergence. The models in dashline are frozen. We use pseudo labels for target samples by clustering when source data is not accessible. Best viewed in color.

Harnessing Adversarial Examples. Formally, the general domain adaptation principle can be formulated as

$$\min_{\theta_F,\theta_C} \mathcal{E}_s(F,C) + \lambda_{dis} dist_{\mathcal{D}_s \leftrightarrow \mathcal{D}_t} \tag{6}$$

where λ_{dis} is a hyper-parameter, $\mathcal{E}_s(F,C)$ is the source risk, $dist_{\mathcal{D}_s \leftrightarrow \mathcal{D}_t}$ denotes the distribution distance between the source and target domains, which is minimized through adversarial learning [6] or moment matching [5] in previous work. In the typical UDA setting, we first optimize θ_F and θ_C simultaneously by minimizing the $\mathcal{E}_s(F,C)$ on the labeled source data,

$$\min_{\theta_F,\theta_C} \mathcal{L}_{cls} = \frac{1}{n_s} \sum_{i=1}^{n_s} \ell\left(C\left(F\left(x_s^i\right)\right), y_s^i\right). \tag{7}$$

In this paper, we propose to improve the generalization ability of the model by learning from adversarial examples. Specifically, we minimize the cross-entropy loss on the adversarial examples that generated by G in the previous step,

$$\min_{\theta_F,\theta_C} \mathcal{L}_{adv} = \frac{1}{n_s} \sum_{i=1}^{n_s} \ell\left(f\left(x_s^i + \epsilon\left(G\left(x_s^i\right) / \left\|G\left(x_s^i\right)\right\|_2\right)\right), y_s^i\right). \tag{8}$$

Eq. (8) only considers the similarity between the predictions of each sample and its adversarial examples within the corresponding ground truth category. To further explore the intrinsic relationship between the original sample and the adversarial sample, thus minimizing the cross-category ambiguity, we introduce $\mathcal{L}_{sim}$, which estimates the prediction discrepancy in a more diverse space. Formally,

$$\min_{\theta_F,\theta_C} \mathcal{L}_{sim} = \frac{1}{n_s} \sum_{i=1}^{n_s} \Gamma\left(p_s^i, \widetilde{p}_s^i\right), \tag{9}$$

where p_s^i and $\widetilde{p}_s^i$ are the probabilistic predictions for the original sample x_s^i and the adversarial sample $\widetilde{x}_s^i$, respectively. Γ is the classifier determinacy disparity [66] which is expressed as

$$\Gamma\left(p_1, p_2\right) = \sum_{i,j=1}^{C} A_{ij} - \sum_{i=1}^{C} A_{ii} = \sum_{i \neq j}^{C} A_{ij}. \tag{10}$$

Here $A = p_1 p_2^\top$ is the prediction correlation matrix of $K \times K$. Obviously, elements on the diagonal represent the prediction consistency for each category, thus $\Gamma(\cdot,\cdot)$ measures the ambiguity between two prediction. Minimizing $\mathcal{L}_{sim}$ encourages the classifier to make a more consistent prediction compared to minimizing Eq. (8) merely. For instance, the discrepancy of the cross-entropy loss between $p_1 = [0.6, 0.3, 0.1]$ and $p_2 = [0.6, 0.1, 0.3]$ is zero since it only considers the position on the ground-truth category, while we cannot conclude they are good predictions since the ambiguity exists. $\mathcal{L}_{sim}$ is able to find this pitfall and reduce cross-category ambiguity. Finally, the learning objective of the model f to harness adversarial examples in UDA can be formulated as,

$$\min_{\theta_F,\theta_C} \mathcal{L}_{cls} + \lambda_{dis} dist_{\mathcal{D}_s \leftrightarrow \mathcal{D}_t} + \lambda_{adv} \mathcal{L}_{adv} + \lambda_{sim} \mathcal{L}_{sim}, \tag{11}$$

where λ_{adv} and λ_{sim} are the trade-off hyper-parameters. The overall pipeline of our framework can be seen in Fig. 2. As pointed out by Engstrom *et al.* [67], the feature representations obtained by adversarial training are approximately invertible, so as to move the image towards the well-chosen direction in latent space that allows high-level human-understandable feature manipulation in pixel space. This makes the features of our model hold semantic information and thus more transferable.

3.3 Robust Transfer in the Absence of Source Data

We explain how our proposed AAA approaches to SFUDA, where the UDA problem becomes how to exploit the knowl-

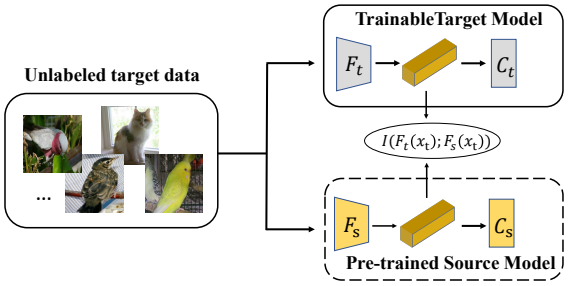


Fig. 3. Illumination of our mutual information maximization.

edge from the pre-trained model and utilize a set of unlabeled target data. Here we freeze the parameters of the source model as we suppose it has encoded the knowledge learned from the source domain, and fork a trainable target model from it for fine-tuning to the target domain. We denote the frozen source model and the trainable target model by $f_s = F_s \circ C_s$ and $f_t = F_t \circ C_t$ for disambiguation.

Utilizing unlabeled data by adversarial training. Intuitively, the model trained on the source domain actually learns a decision boundary supported by the source data. In SFUDA, our purpose is to adjust the decision boundary to make it suitable and robust for the target data. In this regard, we modify Eq. (7) and Eq. (8) by replacing the source data x_s^i with the target data x_t^i, and use the estimated pseudo label $\hat{y}^i$ instead of the ground-truth label y_s^i for self-supervision.

Since clustering based pseudo labels [8], [22] have been proven to be simple yet effective in previous UDA works, inspired by [8], [22], we maintain a memory bank to store the feature representations of all target samples output by current F_t, then perform a weighted k-means clustering as used in [22] to refine pseudo labels. As an alternative framework, we update pseudo labels every certain steps, i.e., we freeze the target model f_t and get current representations, then perform clustering to update $\{\hat{y}^i\}_{i=1}^{n_t}$. Based on updated pseudo labels, we fine-tune f_t by minimizing the loss we defined through back-propagation. This EM algorithm-like strategy will continue to improve the Evidence Lower Bound (ELBO) [68] of the target data.

Exploiting source knowledge by mutual information (MI) maximization. In addition to pseudo label-based adversarial training for target samples. We further propose mutual information maximization between features encoded by f_s and f_t for the same sample to exploit source knowledge from f_s, as shown in Fig. 3. We suppose that f_s contains both domain-specific and domain-shared information. Maximizing the mutual information (domain-shared information) helps the model to exploit domain-invariant information about the input samples, thus preserving the source knowledge. From another perspective, this can be regard as a regularization which prevents the adapted model from changing too significantly and thus stabilizes the training, similar idea has been used in [7].

Formally, the MI measures the Kullback-Leibler (KL) divergence between the joint distribution and the product of marginal distributions of two variables X and Y, i.e,

$$\begin{aligned} I(X,Y) &= \sum_{y\in Y}\sum_{x\in X} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) \\ &= KL(p(x,y)\|p(x)p(y)). \end{aligned} \tag{12}$$

Here X and Y are the the distributions over feature encodings produced by sampling $x = F_s(x_t^i)$ and $y = F_t(x_t^i)$, respectively. It is worth noting that directly estimating mutual information is intractable due to highly complex data distributions. As an alternative, previous studies estimate the mutual information by using various kinds of lower-bounds [69], [70]. In this paper, we choose the Noise-Contrastive Estimation (a.k.a infoNCE) [70] which derived from contrastive learning [71] since our goal is to improve the MI rather than to obtain the exact value. Then the MI can be rewritten as,

$$\begin{aligned} &I(X;Y) = I\left(F_s\left(X_t\right);F_t\left(X_t\right)\right) \\ &\geq \mathbb{E}_{\mathbb{P}}\left[T\left(F_s\left(x_t^i\right),F_t\left(x_t^i\right)\right) - \mathbb{E}_{\tilde{\mathbb{P}}}\left[\log\textstyle\sum_{\hat{x}_t^i} e^{T\left(F_s\left(\hat{x}_t^i\right),F_t\left(x_t^i\right)\right)}\right]\right] \end{aligned} \tag{13}$$

where $\hat{x}_t^i$ is an input data sampled from distribution $\tilde{\mathbb{P}} = \mathbb{P}$. $T(\cdot)$ is a score function implemented by inner product in this work. The motivation is that although the model has been adapted, each $f_t(x_t^i)$ can also distinguish its corresponding mirror feature $f_s(x_t^i)$ from other samples. On one hand, we preserve the source knowledge by MI maximization. On the other hand, this contrastive learning term helps to learn instance-level discriminative features. At last, the learning objective of f_t for source data-absent UDA can be given by,

$$\min_{\theta_{F_t},\theta_{C_t}} \mathcal{L}_{adv} + \lambda_{sim}\mathcal{L}_{sim} - \lambda_{mi} I\left(F_s\left(X_t\right);F_t\left(X_t\right)\right) + \lambda_{cls}\mathcal{L}_{cls}. \tag{14}$$

3.4 Robust Transfer in the Absence of Target Data

We also extend our idea to UDA without target data. The majority of existing studies assume there should be multiple source domains at hand for generalization. For a fair comparison, we follow this setting and divide our source distribution into K disjoint ones, $\mathcal{D}_s = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$. The samples in each source domain are denoted as $\left\{\left(x_i^k, y_i^k\right)\right\}_{i=1}^{n_s^k}$, where n_s^k is the number of samples in $\mathcal{D}_k$.

We modify $\mathcal{L}_{cls}$ and $\mathcal{L}_{adv}$ by summing up and averaging the loss over source domains, i.e.,

$$\mathcal{L}_{cls} = \frac{1}{m}\sum_{k=1}^{K}\sum_{i=1}^{n_s^k} \ell\left(f\left(\left(x_i^k\right), y_i^k\right)\right), \tag{15}$$

$$\mathcal{L}_{adv} = \frac{1}{m}\sum_{k=1}^{K}\sum_{i=1}^{n_s^k} \ell\left(f\left(x_i^k + \epsilon\left(G\left(x_i^k\right)/\left\|G\left(x_i^k\right)\right\|_2\right)\right), y_i^k\right), \tag{16}$$

where $m = \sum_{k=1}^{K}\sum_{i=1}^{N_k}$. The learning objective of f is,

$$\min_{\theta_F,\theta_C} \mathcal{L}_{cls} + \lambda_{adv}\mathcal{L}_{adv} + \lambda_{sim}\mathcal{L}_{sim}. \tag{17}$$

3.5 Overall Training Strategy

Now we have employed our method in the typical, source data-absent and target data-absent UDA, respectively. These three UDA settings share the same objective for generator G (i.e., Eq. (5)) and slightly differ in the learning objective of F and C, namely, Eq.(11), Eq.(14) and Eq.(17) for typical, source data-absent and target data-absent UDA, respectively. The training of the generator G and the classification models F and C are carried out alternatively until convergence.

4 Theoretical Analysis

In this section, we draw a connection between our method and the theory of domain adaptation [72], [73], which gives the generalization bound of the expected error on the target samples. We denote the sampled data from multiple sources as $\mathcal{S}_S = \{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$ and corresponding adversarial samples as $\mathcal{S}_{adv} = \{\mathcal{S}_{K+1}, \ldots, \mathcal{S}_{2K}\}$. The numbers of samples in each datasets are $N_1, \ldots, N_K, N_{K+1}, \ldots, N_{2K}$, respectively, with total number m. We then construct our training set as $\mathcal{S}_{S^+} = \mathcal{S}_S + \mathcal{S}_{adv}$. Similarly, let $\mathcal{P}_{S^+} = \mathcal{P}_S + \mathcal{P}_{adv} = \{\mathcal{P}_1, \ldots, \mathcal{P}_K, \mathcal{P}_{K+1}, \ldots, \mathcal{P}_{2K}\}$ be the feature distributions of $\mathcal{S}_{S^+}$, and $\mathcal{P}_T$ be the target feature distribution. The $\mathcal{H}\Delta\mathcal{H}$-distance [72], [74] is widely used in domain adaptation to define the discrepancy between two distributions $\mathcal{P}_S$ and $\mathcal{P}_T$ w.r.t. a hypothesis set $\mathcal{H}$. Formally,

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{P}_S, \mathcal{P}_T) = 2 \sup_{h_1, h_2 \in \mathcal{H}} \left| P_{\mathbf{x}\sim\mathcal{P}_S}\left[h_1(\mathbf{x}) \neq h_2(\mathbf{x})\right] - P_{\mathbf{x}\sim\mathcal{P}_T}\left[h_1(\mathbf{x}) \neq h_2(\mathbf{x})\right] \right|. \tag{18}$$

In our method, we consider $2K$ training domains. We use vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{2K})$ to denote domain weights with $\sum_{i=1}^{2K} \alpha_i = 1$. Then given the hypothesis h, the empirical risk over multiple domains can be expressed as

$$\hat{\epsilon}_{\boldsymbol{\alpha}}(h) = \sum_{i=1}^{N} \alpha_i \epsilon_i(h) = \sum_{i=1}^{N} \frac{\alpha_i}{N_i} \sum_{x \in \mathcal{S}_i} \left| h(x) - f_i(x) \right|, \tag{19}$$

where f_i is the ground-truth labeling function of $\mathcal{S}_i$. Based on above definitions, Ben-David *et al.* [72] gave the VC-dimension-based expected error bound on the target samples as following,

Theorem 1. Let $\mathcal{H}$ be a hypothesis space of VC dimension d. For each $j \in \{1, \ldots, N\}$, let S_j be a labeled sample of size $\beta_j m$ generated by drawing $\beta_j m$ points from $\mathcal{D}_j$ and labeling them according to f_j. If $\hat{h} \in \mathcal{H}$ is the empirical minimizer of $\hat{\epsilon}_\alpha(h)$ for a fixed weight vector $\boldsymbol{\alpha}$ on these samples and $h_T^* = \min_{h\in\mathcal{H}} \epsilon_{\mathcal{T}}(h)$ is the target error minimizer, then for any $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\epsilon_{\mathcal{T}}(\hat{h}) \leq \epsilon_{\mathcal{T}}(h_T^*) + 4\sqrt{\left(\sum_{j=1}^{N} \frac{\alpha_j^2}{\beta_j}\right)\left(\frac{d\log(2m) - \log(\delta))}{2m}\right)} + 2\gamma_\alpha + d_{\mathcal{H}\Delta\mathcal{H}}\left(\mathcal{P}_\alpha, \mathcal{P}_T\right), \tag{20}$$

where $\gamma_{\boldsymbol{\alpha}} = \min_h \{\epsilon_{\mathcal{T}}(h) + \epsilon_{\boldsymbol{\alpha}}(h)\}$. $d_{\mathcal{H}\Delta\mathcal{H}}\left(\mathcal{P}_\alpha, \mathcal{P}_T\right)$ measures the divergence between the target distribution and the α-weighted mixture of source distributions. [75] points out this kind of $\mathcal{H}\Delta\mathcal{H}$-distance could provide a significantly tighter bound than the pair-wise $\mathcal{H}\Delta\mathcal{H}$-distance between each source-target pair (see Theorem 4 in [72]).

Theorem 1 shows that the upper-bound of the target error of the learned hypothesis depends on the divergence between the target domain and the adversarial examples-augmented source domain(s). And [76] points out that the error of the ideal joint hypothesis (i.e., $\gamma_{\boldsymbol{\alpha}}$) is associated with the feature discriminability. This is the direct motivation of our idea. In the typical UDA ($K = 1$), although we can optimize $\mathcal{H}\Delta\mathcal{H}$-distance between the source domain and the target domain directly, our method can further minimize the upper-bound by improving the feature discriminability, since adversarial training makes the model smooth at each data point [61]. In the target-absent UDA, if we regard the target data as a special kind of adversarial attack, the learned model by AAA has guaranteed generalization bounds w.r.t. the target domain data if it is not much "different" from the training data [77]. In addition, we explicitly explore the unseen distribution by adversarial examples, which further reduces the "difference" and minimizes the upper-bound. For source-absent UDA, we do not directly use the source-learned hypothesis to measures the target risk as in Eq. (20), instead, we fine-tune the learned source minimizer with the unlabeled target data. We analyze the upper-bound with the following theory,

Theorem 2. [72] Let $\mathcal{H}$ be the hypothesis class. Given two domains $\mathcal{S}$ and $\mathcal{T}$, we have

$$\forall h \in \mathcal{H}, \epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) + \lambda, \tag{21}$$

where $\epsilon_{\mathcal{S}}(h)$ is the expected error on the source samples, $\lambda = \min_h \{\epsilon_{\mathcal{S}}(h) + \epsilon_{\mathcal{T}}(h)\}$ is the shared expected error.

Theorem 2 shows the target risk also depends on the source risk with the shared hypothesis h. Inspired by [7], [78] which aims to learn a related but different model for each domain, we use MI maximization as a regularization to make the target model related with the source pre-trained model, so as to preserve the source knowledge encoded in it. In other words, this can preserve the first term in Eq.(21) which has been minimized in the pre-trained source model. In addition, adversarial training makes the target features more discriminative at category-level, and MI maximization (the instance-wise contrastive learning loss) improves the discriminability at instance-level, thus the third term can also be minimized.

5 Experiments

5.1 Setup

We perform extensive evaluations of our proposed method in three UDA scenarios (i.e., typical UDA, source data-absent UDA and target data-absent UDA) over four publicly available visual datasets.

Office-31 [89] is a standard visual benchmark widely adopted by domain adaptation methods. It is constructed by images from three distinct domains which share 31 categories: Amazon (**A**), Webcam (**W**) and DSLR (**D**). There are 2817, 795, 498 images involved in each domain, respectively.

Office-Home [90] is a challenging UDA dataset which contains four distinct domains: Artistic (**Ar**), Clipart (**Cl**), Product (**Pr**), and Real-World (**Rw**). These four domains share 65 categories of objects and contain 15,500 images totally.

VisDA-2017 [91] is a challenging large-scale dataset for 2017 visual domain adaptation challenge which focuses on the simulation-to-reality adaptation. It consists of over 280 thousand images across 12 categories. We use the training set as our source domain that contains 152,397 synthetic images generated by rendering 3D models, and use the validation set as our target domain which has 55,388 real images collected from the Microsoft COCO dataset.

TABLE 1
Accuracy(%) on **VisDA-2017** for typical unsupervised domain adaptation (ResNet-101). The best results are highlighted in **bold**.

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Avg.
ResNet [79]	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
DAN [5]	87.1	63.0	76.5	42.0	90.3	42.9	85.9	53.1	49.7	36.3	85.8	20.7	61.6
DANN [6]	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
MCD [80]	87.0	60.9	83.7	64.0	88.9	79.6	84.7	76.9	88.6	40.3	83.0	25.8	71.9
ADR [81]	87.8	79.5	83.7	65.3	92.3	61.8	88.9	73.2	87.8	60.0	85.5	32.3	74.8
SWD [82]	90.8	82.5	81.7	70.5	91.7	69.5	86.3	77.5	87.4	63.6	85.6	29.2	76.4
CDAN+E [31]	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.9
CDAN+BSP [83]	92.4	61.0	81.0	57.5	89.0	80.6	90.1	77.0	84.2	77.9	82.1	38.4	75.9
AFN [84]	93.6	61.3	84.1	70.6	94.1	79.0	**91.8**	79.6	89.9	55.6	89.0	24.4	76.1
BNM [85]	89.6	61.5	76.9	55.0	89.3	69.1	81.3	65.5	90.0	47.3	**89.1**	30.1	70.4
STAR [86]	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 [22]	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
AAA+CAN (Ours)	**97.8**	86.4	**84.9**	71.9	97.3	**97.5**	91.0	**82.6**	**97.3**	**96.3**	88.9	58.4	**87.5**

TABLE 2
Accuracy(%) on **Office-31** for source-free unsupervised domain adaptation (ResNet-50). "SF" stands for source free and the best results are highlighted in **bold**.

Method (Source → Target)	SF	A → W	D → W	W → D	A → D	D → A	W → A	Avg
ResNet [79]		68.4±0.2	96.7±0.1	99.3±0.1	68.9±0.2	62.5±0.3	60.7±0.3	76.1
DAN [5]		80.5±0.4	97.1±0.2	99.6±0.1	78.6±0.2	63.6±0.3	62.8±0.2	80.4
RTN [87]		84.5±0.2	96.8±0.1	99.4±0.1	77.5±0.3	66.2±0.2	64.8±0.3	81.6
DANN [6]		82.0±0.4	96.9±0.2	99.1±0.1	79.7±0.4	68.2±0.4	67.4±0.5	82.2
MinEnt [88]	✗	86.8±0.2	**98.6**±0.1	**100.0**±.0	88.7±0.3	67.2±0.5	63.4±0.4	84.1
MCD [80]		88.6±0.2	98.5±0.1	**100.0**±.0	92.2±0.2	69.5±0.1	69.7±0.3	86.5
CDAN+E [31]		94.1±0.1	**98.6**±0.1	**100.0**±.0	92.9±0.2	71.0±0.3	69.3±0.3	87.7
AFN [84]		90.1±0.8	**98.6**±0.2	99.8±.0	90.7±0.5	73.0±0.2	70.2±0.3	87.1
BNM [85]		91.5±.0	98.5±.0	**100.0**±.0	90.3±.0	70.9±.0	71.6±.0	87.1
SHOT [8]		90.1 ± 0.2	98.4 ± 0.3	99.9 ± 0.1	94.0 ± 0.1	74.7 ±0.2	74.3±0.1	88.6
SFDA [37]	✓	91.1±0.3	98.2±0.3	99.5±0.2	92.2±0.2	71.0±0.2	71.2±0.2	87.2
AAA (Ours)		**94.2**±0.3	98.1±0.2	99.8±0.2	**95.6**±0.4	**75.6**±0.2	**76.0**±0.3	**89.9**

PACS [45] is a recently proposed specially for domain generalization. PACS consists of four domains: Photo (P), Art painting (A), Cartoon (C) and Sketch(S), which share seven common categories: dog, elephant, giraffe, guitar, horse and person.

Since training procedures of existing UDA methods heavily depend on domain divergence, our method can realize its potential value better when the domain divergence is unknown. For this consideration, our evaluation mainly focuses on divergence-agnostic scenarios. For typical UDA, we embed our AAA into the state-of-the-art typical UDA method CAN [22], we show our AAA is able to further improve the adaptation performance.

5.2 Implementation Details

We choose PyTorch [92] to implement our AAA framework, and NIVIDIA GeForce RTX 2080 Ti GPU is employed as our hardware platform. We repeat each experiment three times and report the mean accuracy. The network architecture and experimental settings are detailed as follows.

Network Architecture. For the typical UDA, we follow the network architectures (including the backbone and the classifier) with CAN [22], namely, we use the pre-trained ResNet-50 or ResNet-101 [79] as the backbone to extract features from raw images, and replace the last linear layer with a task-specific layer which has M output units. For source-free UDA, we employ the same architectures as used in SHOT [8], i.e., we replace the last layer of the backbone with a bottleneck layer with 256 units and add a three-layer FC with 1000 units of the second layer. For domain generalization, we evaluate multiple backbones such as AlexNet [26] and ResNet for a fair comparison with previous works. The classifier is a linear layer with M output units. As for perturbation generator G, we adopt a similar architecture in image-to-image translation [35]. Specifically, we use 3 convolutional layers, 4 residual blocks [79] and 3 deconvolutional layers as the encoder, the feature extractor and the decoder, respectively.

Experimental Settings. For all experiments, we adopt the Adam [93] to optimize the generator G with learning rate 0.0001 and epsilon $1e^{-8}$. The feature extractor F and classifier C are optimized with the SGD optimizer. For typical UDA, we use the same hyper-parameters as reported in CAN [22]. For source-free UDA, The learning rate η of F_t is set to 0.01 except 0.001 for VisDA-2017, and the learning rate of C_t is 10 times lower than that of F_t. The batch size is set to 32 for Office-31 and 64 for VisDA-2017 and Office-Home. For domain generalization, the learning rate is 0.001 for AlexNet and 0.004 for ResNet. We set the batch size as 64 for all tasks. Following [8], [31], we adopt a learning rate scheduler $\eta_p = \eta \cdot (1 + 10 \cdot p)^{-0.75}$ for all optimizer with p the learning progress.

TABLE 3
Accuracy(%) on **Office-Home** for source-free unsupervised domain adaptation (ResNet-50). The best results are highlighted in **bold**.

Method	SF	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg
ResNet [79]	✗	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DANN [6]		45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
DAN [5]		43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
CDAN+E [31]		50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
CDAN+BSP [83]		52.0	68.6	76.1	58.0	70.3	70.2	58.6	50.2	77.6	72.2	**59.3**	81.9	66.3
AFN [84]		52.0	71.7	76.3	64.2	69.9	71.9	63.7	51.4	77.1	70.9	57.1	81.5	67.3
SFDA [37]	✓	48.4	73.4	76.9	64.3	69.8	71.7	62.7	45.3	76.6	69.8	50.5	79.0	65.7
SHOT [8]		55.3	77.7	81.6	**66.5**	78.1	78.2	67.2	52.0	80.7	72.1	58.6	83.2	70.9
AAA (Ours)		**56.7**	**78.3**	**82.1**	66.4	**78.5**	**79.4**	**67.6**	**53.5**	**81.6**	**74.5**	58.4	**84.1**	**71.8**

TABLE 4
Accuracy(%) on **VisDA-2017** for source-free unsupervised domain adaptation (ResNet-101). The best results are highlighted in **bold**.

Method	SF	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Avg.
ResNet [79]	✗	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
DAN [5]		87.1	63.0	76.5	42.0	90.3	42.9	85.9	53.1	49.7	36.3	85.8	20.7	61.6
DANN [6]		81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
MCD [80]		87.0	60.9	83.7	64.0	88.9	79.6	84.7	76.9	88.6	40.3	83.0	25.8	71.9
SWD [82]		90.8	82.5	81.7	70.5	91.7	69.5	86.3	77.5	87.4	63.6	85.6	29.2	76.4
CDAN+E [31]		85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.9
AFN [84]		93.6	61.3	84.1	70.6	94.1	79.0	**91.8**	79.6	89.9	55.6	89.0	24.4	76.1
BNM [85]		89.6	61.5	76.9	55.0	89.3	69.1	81.3	65.5	90.0	47.3	**89.1**	30.1	70.4
SHOT [8]	✓	94.3	**88.5**	80.1	57.3	93.1	94.9	80.7	80.3	**91.5**	89.1	86.3	58.2	82.9
3C-GAN [7]		**94.8**	73.4	68.8	**74.8**	93.1	**95.4**	88.6	**84.7**	89.1	84.7	83.5	48.1	81.6
SFDA [37]		86.9	81.7	**84.6**	63.9	93.1	91.4	86.6	71.9	84.5	58.2	74.5	42.7	76.7
AAA (Ours)		94.4	85.9	74.9	60.2	**96.0**	93.5	87.8	80.8	90.2	**92.0**	86.6	**68.3**	**84.2**

For other hyper-parameters, we utilize $\epsilon = 1$, $\lambda_{adv} = 0.1$ for typical UDA. For source-free UDA, we set $\epsilon = 10$, $\lambda_{mi} = 0.01$ and $\lambda_{cls} = 0.3$. We use $\lambda_{sim} = 0$ for SFUDA since we empirically find $\mathcal{L}_{sim}$ has marginal influence in this setting. For domain generalization, we use $\epsilon = 10$ and $\lambda_{adv} = 0.1$. Besides, the weights of $\mathcal{L}_{sim}$ and $\mathcal{L}_{adv}$ are the same for typical UDA and DG, and λ_{reg} for generator G is empirically fixed as 0.1 for all tasks.

As for MI estimation, we use the features output by the last convolutional layer of the backbone network. We resize the raw images to $224 \times 224 \times 3$ and adopt random horizontal flip strategy during training. We do not use any data augmentation such as flip or ten-crop [31] during validation.

For domain generation, our training procedure strictly follows the protocol in the community, i.e., we train our model on the training set, then choose the model which achieves the highest accuracy on the validation set for testing on the unseen target domain. The dataset divisions of **PACS** and **Office-Home** are the same as [51]. For **Office-31**, we randomly split 10% data from the training set as the validation set.

5.3 Results of Typical UDA

The classification results on **VisDA-2017** are shown in Table 1. We follow previous works [7], [80] and employ ResNet-101 as our backbone module. The results of compared baselines are directly cited from previous papers wherever available. It is favorable that our method outperforms other methods by large margins. Specifically, our method achieve the best performance in 6 out of 12 categories: *plane, bus, knife, person, plant, sktbrd*. Besides, our method brings an improvement up to 35.1% in terms of the averaged accuracy compared with the source-only model (i.e., ResNet). compared with our baseline, our method also boosts the averaged accuracy of **CAN** [22] by 0.3%. We guess the relatively small improvement may result from two reasons: (1) **CAN** is a very advanced method in typical UDA which already achieves an objectively high accuracy, thus leading to less room for improvement. (2) The performance gain in typical UDA mainly benefits from the minimization of the domain divergence $dist_{\mathcal{D}_s \leftrightarrow \mathcal{D}_t}$ since the λ in Eq.(21) is expected to be small. However, the results show that our AAA is also helpful to improve the typical UDA methods.

TABLE 5
Accuracy(%) on **Office-31** for domain generalization (ResNet-18). $\mathfrak{R}$ stands for all the rest domains and keeps the same meaning in following tables. The best results are highlighted in **bold**.

Method	$\mathfrak{R} \rightarrow$ A	$\mathfrak{R} \rightarrow$ D	$\mathfrak{R} \rightarrow$ W	Avg.
DeepAll [51]	54.4±0.2	98.2±0.2	93.3±0.3	82.0±0.2
RSC [94]	52.4±0.3	98.3±0.1	94.3±0.2	81.7±0.2
AAA (Ours)	**55.7±0.2**	**98.6±0.2**	**94.8±0.3**	**83.0±0.2**

5.4 Results of Source Data-absent UDA

We further evaluate our method on three datasets including **Office-31, Office-Home** and **VisDA-2017** under the SFUDA setting. We adopt the ResNet-50 backbone as the feature extractor for **Office-31** and **Office-Home**, and employ the ResNet-101 for **VisDA-2017**. The maximum epoch number is empirically set as 20 for all datasets and the results of them are reported in Table 2, 3 and 4. We compare our method

TABLE 6
Domain generalization results on **PACS** using AlexNet and ResNet backbone. The best results are highlighted in **bold**.

Method	Backbone	$\mathfrak{R} \rightarrow$ P	$\mathfrak{R} \rightarrow$ A	$\mathfrak{R} \rightarrow$ C	$\mathfrak{R} \rightarrow$ S	Avg.
DeepAll [51]	AlexNet	89.98	66.68	69.41	60.02	71.52
Epi-FCR [46]	AlexNet	86.1	64.7	72.3	65.0	72.0
JiGen [51]	AlexNet	89.00	67.63	71.71	65.18	73.38
MMLD [50]	AlexNet	88.98	69.27	**72.83**	66.44	74.38
MASF [95]	AlexNet	90.68	70.35	72.46	67.33	75.21
DG-ER [10]	AlexNet	89.92	71.34	70.29	71.15	75.67
EISNet [52]	AlexNet	**91.20**	70.38	71.59	70.25	75.86
AAA (Ours)	AlexNet	89.8±0.39	**71.51±0.33**	71.09±0.33	**72.55±0.64**	**76.13±0.23**
DeepAll [51]	ResNet18	96.28	78.96	73.93	70.59	79.94
JiGen [51]	ResNet18	96.03	79.42	75.25	71.35	80.51
MASF [95]	ResNet18	94.99	80.29	**77.17**	71.69	81.03
DG-ER [10]	ResNet18	**96.65**	80.70	76.40	71.77	81.38
Epi-FCR [46]	ResNet18	93.90	82.10	77.00	73.00	81.50
EISNet [52]	ResNet18	95.83	81.89	76.44	74.33	82.15
AAA (Ours)	ResNet18	95.93±0.53	**82.52±0.42**	77.05±1.05	**77.65±0.66**	**83.29±0.37**
DeepAll [51]	ResNet50	97.66	86.20	78.70	70.63	83.29
MASF [95]	ResNet50	95.01	82.89	80.49	72.29	82.67
DG-ER [10]	ResNet50	**98.25**	**87.51**	79.31	76.30	85.34
EISNet [52]	ResNet50	97.11	86.64	**81.53**	78.07	85.84
AAA (Ours)	ResNet50	97.54±0.21	86.72±0.39	79.95±0.52	**82.26±0.30**	**86.62±0.27**

TABLE 7
Domain generalization results on **Office-Home** using ResNet18 backbone. The best results are highlighted in **bold**.

Method	$\mathfrak{R} \rightarrow$ Ar	$\mathfrak{R} \rightarrow$ Cl	$\mathfrak{R} \rightarrow$ Pr	$\mathfrak{R} \rightarrow$ Rw	Avg.
DeepAll [51]	52.15	45.86	70.86	73.15	60.51
D-SAM [96]	58.03	44.37	69.22	71.45	60.77
JiGen [51]	53.04	47.51	71.47	72.79	61.20
RSC [94]	58.42	47.90	71.63	74.54	63.12
AAA (Ours)	**59.81±0.22**	**51.79±0.31**	**73.49±0.22**	**75.20±0.18**	**65.07±0.09**

with both source-accessible and source-free UDA methods [1]. Table 2 shows that although we do not access the source data, our method still outperforms other UDA methods on **Office-31** dataset. Specifically, our AAA achieves the highest accuracy on three tasks: **A** $\rightarrow$ **D**, **A** $\rightarrow$ **W** and **D** $\rightarrow$ **A**. For other difficult tasks (e.g., **W** $\rightarrow$ **A**), our method performs a little worse than [7]. A possible reason is that the samples in domain **W** are significantly less than that in **A**, the limited diversity of the source domain adds the difficulty for training a robust model that generalizes well to the target domain. Nevertheless, our method reaches the highest averaged accuracy in 10 out of 12 tasks.

Table 3 reports the results on **Office-HOME** dataset. It is obvious that our method achieves performance surpassing or comparable to other state-of-the-arts with the averaged accuracy 71.8%. The results show that our AAA is beneficial to boost the adaptation capability in all tasks compared with the source-only model and obtain the highest performance in 6 tasks. Comparing with the source-free method SFDA [37], we achieve an extra gain of 6.1% on averaged accuracy.

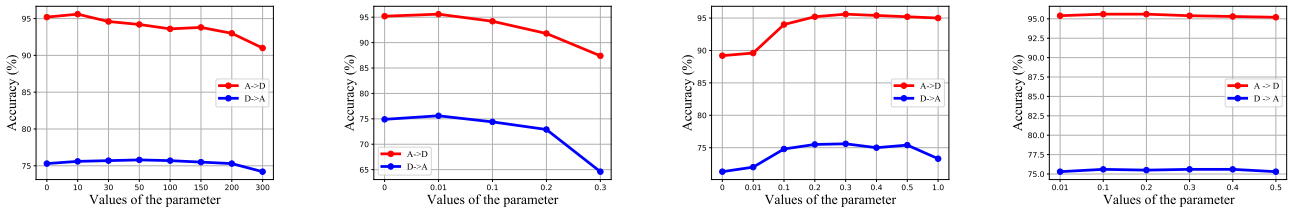
For the large-scale datasets **VisDA-2017**, our AAA also outperforms other popular UDA methods, advancing the averaged accuracy from 82.9% (SHOT) to 84.2%. It is desirable that AAA significantly boosts the accuracy of the most difficult category *truck* to 68.3%, surpassing the second-best method SHOT by 10.1%. For other tasks such as *horse*, *person* and *sktbrd*, our method also gets desirable improvements comparing with the second-best methods by 2.9%, 3.9% and 2.9%, respectively.

These results reveal an interesting observation that the algorithms for SFUDA have the ability to achieve better performance than algorithms for typical DA. This may be because many typical UDA methods are prone to improve the transferability at the price of the discriminability of target features [83], thus we believe there is an optimal balance between $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ and λ in Eq. (21). Although we cannot optimize $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S},\mathcal{T})$ directly in SFUDA, we enhance the discriminability of the target features to optimize λ by adversarial attacks and defense.

5.5 Results of Target Data-absent UDA

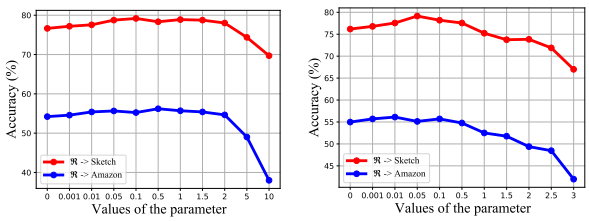
For target data-absent UDA (i.e., domain generalization), we report our results as well as previous state-of-the-arts on **Office-31**, **PACS** and **Office-HOME** in Table 5, 6, and 7, respectively. For all experiments in this setting, we specify a domain as the unseen target domain and use all rest

1. We use the authors' publicly available code for SHOT [8] to reproduce the baseline results. However, we were unable to reproduce the results reported in [8] after extensive runs. Thus we report the best performance we found for SHOT.



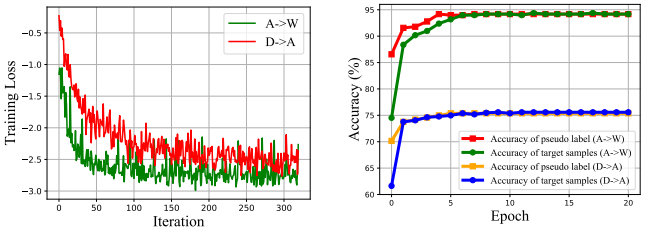
(a) ϵ (SFUDA) (b) λ_{mi} (SFUDA) (c) λ_{cls} (SFUDA) (d) λ_{reg} (SFUDA)

Fig. 4. The hyper-parameter sensitivity analysis w.r.t (a) ϵ (b) λ_{mi} (c) λ_{cls} and (d) λ_{reg} in our method. Task $A \rightarrow D$ and $D \rightarrow A$ on **Office-31** under the SFUDA setting are chosen for evaluation.



(a) λ_{sim} (DG) (b) λ_{adv} (DG)

Fig. 5. The hyper-parameter sensitivity analysis w.r.t λ_{sim} (a) and λ_{adv} (b) in our method. We choose tasks $\mathfrak{R} \rightarrow S$ on **PACS** and task $\mathfrak{R} \rightarrow A$ on **Office-31** under the DG setting as examples. Both of them are base on ResNet-18 backbone.



(a) Training Loss (SFUDA) (b) Test Accuracy (SFUDA)

Fig. 6. The training process of our model on **Office-31**. We take A→W and D→A on Office-31 as examples.

domains as source domains. These results are obtained after 30 epochs of training. Similar to the observations in above experiments, our method achieves state-of-the-art performance on all three datasets. For **Office-31**, we evaluate our method based on ResNet-18 backbone, and compare it with the recent state-of-the art RSC [94] as well as the widely used baseline DeepAll [51], which simply mixes all source domains as one combined domain and train the model with the standard cross-entropy loss. According to the results, our method outperforms other two baseline by a large marginal. Comparing the results reported in previous works, we can find that in the domain generalization community, even 1% improvement on averaged accuracy is challenging. Our method is definitely effective and sets a strong baseline for further studies.

For **PACS**, we apply our AAA to AlexNet, ResNet-18 and ResNet-50 to evaluate our method. Table 6 lists the accuracy for the target data. Obviously, our methods can reach the highest averaged accuracy with all three backbones and performs the best when employing ResNet-50 backbone. For the relatively difficult task $\mathfrak{R} \rightarrow$ S, our method brings significantly higher accuracy (i.e. 72.55%, 77.65% and 82.26%) than other popular DG approaches, surpassing the second-best ones by 2.3%, 3.32% and 4.19% with different backbones, respectively.

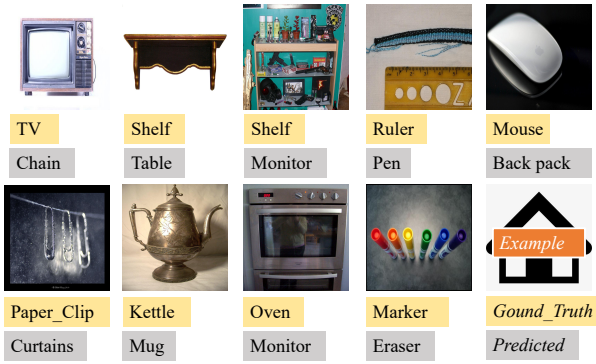
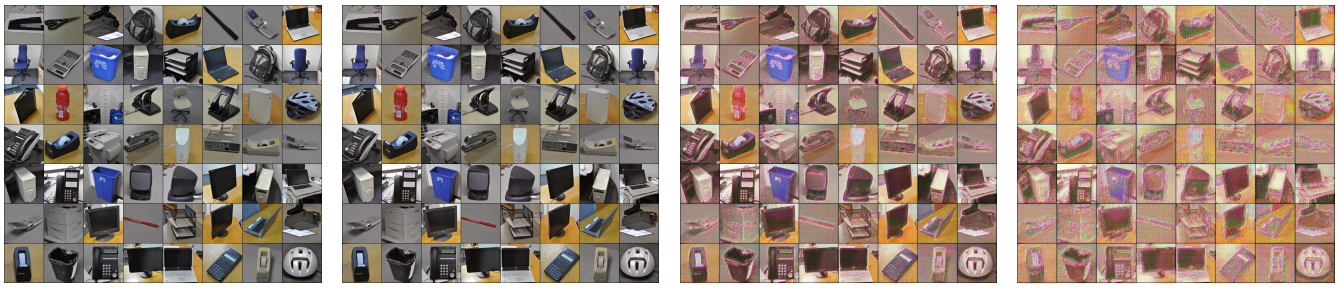


Fig. 7. The qualitative results on **Office-Home** for DG task $R \rightarrow A$. The yellow label is the ground-truth and the gray label is the predicted label.

Comparing the accuracy in these three datasets, we have the observation that our method performs much better on the large dataset **Office-Home**, with 1.95% improvement on averaged accuracy compared with the second-best one. As shown in Table 7, we boost the accuracy on domain *Clipart* from 47.90% to 51.79%. Please note that comparing with the other two datasets, **Office-Home** has much more categories and samples, thus making generalization more challenging. The superior performance of our method on such a large-scale dataset further verify the effectiveness of our method.

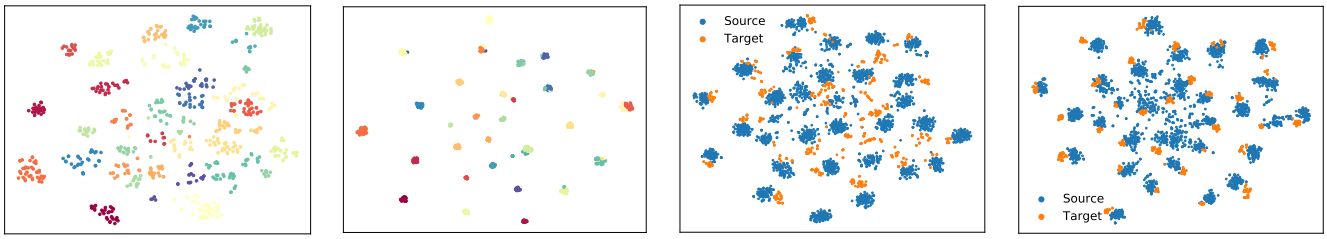
5.6 Model Analysis

Parameter Sensitivity. We check the sensitivities of all hyper-parameters in our method, including ϵ, λ_{cls}, λ_{mi}, λ_{sim}, λ_{reg} and λ_{adv}, under the SFUDA setting and the DG setting. The results are reported in Fig. 4 and Fig. 5, respectively. According to Fig. 4(a), the accuracy first increase and then decrease as ϵ varies from 0 to 300. This may be because too large ϵ would deteriorate the original information of the image, leading to unstable training. Fig. 4(b) indicates that a small λ_{mi} (e.g. 0.01) tends to have a better performance. When $\lambda_{mi} > 0.1$, the accuracy would suffer an obvious decline. The possible reason is that there are inherent differences between the source domain and the target domain, overemphasis on maximizing mutual information will lead to negative transfer. Fig. 4(c) shows that when $\lambda_{cls} < 0.3$, the self-supervised loss can significantly



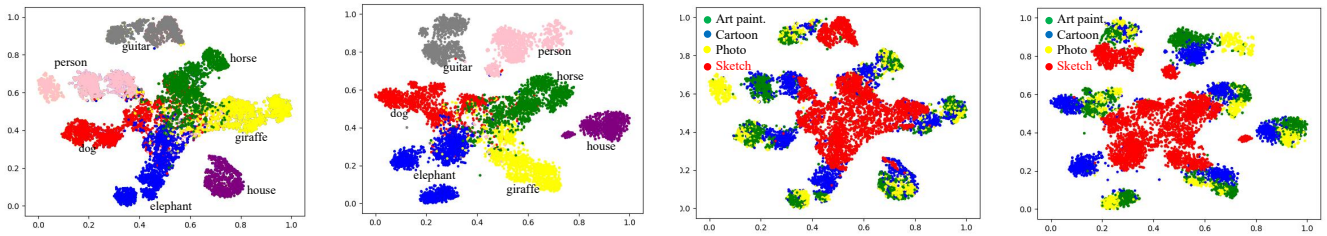
(a) Original (b) $\epsilon = 10$ (c) $\epsilon = 100$ (d) $\epsilon = 200$

Fig. 8. Comparison of the original images with adversarial images generated by our method on **Office-31**. The magnitude ϵ is switched to 0, 10, 100 and 200, respectively.



(a) Source Model (Classes) (b) Adapted Model (Classes) (c) Source Model (Domains) (d) Adapted Model (Domains)

Fig. 9. The t-SNE features from the last bottleneck layer of our ResNet backbone before and after adaptation for SFUDA task. We take A→W on **Office-31** as an example. Different colors represent different classes in (a), (b) and represent different domains in (c), (d). It is worth noting that source data are not available in our setting. We visualize the source samples as auxiliary information for the sake of a better understanding in (c) and (d). One should focus on the target samples in this figure. Best viewed in color.



(a) EISNet (Classes) (b) Ours (Classes) (c) EISNet (Domains) (d) Ours (Domains)

Fig. 10. The t-SNE features learned by EISNet and our method for DG task $\mathfrak{R} \rightarrow S$ on **PACS**. We provide both the class-level feature in (a), (b) and domain-level feature in (c) and (d) for better understanding. Best viewed in color.

boost the performance in SFUDA, thus a proper λ_{cls} is important for our method. Instead, a large λ_{cls} will lead to negative transfer since the pseudo labels are still noisy after clustering. In addition, Fig. 4(d) validates that our model performs uniformly in a wide range of values. This shows that our model is not sensitive to λ_{reg}. From Fig. 5(a) and 5(b), we can see that our model is robust to the variations of λ_{sim} and λ_{adv} when $\lambda_{sim} < 2$ and $\lambda_{adv} < 0.5$. The accuracy increase steadily when λ_{sim} varies from 0 to 0.1, this demonstrates the important role of $\mathcal{L}_{sim}$ in exploring the relationship between original data and adversarial data, which is beneficial to promote the adaptation capability of the model. Generally, the parameters in our method are not sensitive when they are in proper ranges.

Training Stability. We report the training loss and the test accuracy of our method during the optimization process in Fig. 6(a) and 6(b) on SFUDA task $A \rightarrow W$ and $D \rightarrow A$. Fig. 6(a) shows that in both tasks, the training loss can rapidly decrease to a small value after 100 iterations. They keep decreasing smoothly and converge after about 300 iterations. In Fig. 6(b), we can see that our method can quickly converge after nearly 10 epochs on both tasks. We also report the accuracy curves of pseudo labels in Fig. 6(b). It is obvious these clustering-based pseudo labels can boost the accuracy of nave pseudo labels significantly at the beginning. As the training goes on, the accuracy of pseudo labels is also increase. Thus although the pseudo labels are wrongly predicted at the first epoch, it is possible to be refined in the later training process.

Significance test of the improvement. To further verify the quantitative results achieved by our model, we perform a significant test, namely McNemars Test [97] to present the significant improvement brought by our AAA contrast to the strong baseline method SHOT. In precise, we first obtained the SHOT model via the authors' publicly available code and test it on four tasks on office-31 that still have room for improvement. The results for McNemars Test are shown in Table 8. For the calculation of p-value, we assume the test

TABLE 8
Results of the significance test over four relatively hard tasks on **Office-31**. We employ McNemars test to compare the model trained by SHOT and our AAA to show the significant improvement brought by our method.

A->W	SHOT Correct	SHOT Incorrect	A->D	SHOT Correct	SHOT Incorrect
AAA Correct	712	37	AAA Correct	461	21
AAA Incorrect	5	41	AAA Incorrect	7	9
p-value	0.00001		p-value	0.01254	
D->A	SHOT Correct	SHOT Incorrect	W->A	SHOT Correct	SHOT Incorrect
AAA Correct	2084	48	AAA Correct	2060	76
AAA Incorrect	22	663	AAA Incorrect	31	650
p-value	0.00255		p-value	0.00002	

TABLE 9
Ablation study on **Office-31** with the SFUDA setting. The w/, w/o and r.p is short for with, without and random perturbation, respectively.

Method	A→W	D→W	W→D	A→D	D→A	W→A	Avg
w/o $\mathcal{L}_{\text{adv}}$	90.4	97.1	**100.0**	95.2	74.9	**76.0**	88.9
w/o MI	92.3	97.4	99.8	95.4	75.2	75.9	89.3
w/o $\mathcal{L}_{\text{cls}}$	91.1	97.1	99.8	89.2	71.3	70.4	86.5
w/ r.p.	90.2	97.4	**100**	95.4	74.3	75.8	88.9
Full Model	**94.2**	**98.1**	99.8	**95.6**	**75.6**	**76.0**	**89.9**

TABLE 10
Comparison between different adversarial attack strategies for domain generalization task on **PACS** using AlexNet.

Method	Photo	Art.	Cartoon	Sketch	Avg.
PGD [12]	89.40	67.48	71.12	67.68	73.85
AAA (ours)	**89.80**	**71.51**	**71.09**	**72.55**	**76.13**

statistic follows a Chi-Squared distribution with 1 degree of freedom if each cell in the contingency table used in the calculation has a count of at least 25, otherwise a binomial distribution is assumed. Following common practices, we set the significance threshold as 0.05, which means the improvement is significant (reject Null Hypothesis) if p-value $<= 0.05$ or marginal (accept Null Hypothesis) otherwise. Table 8 shows that on all tested tasks, two models have a quite different proportion of errors on the test set. The small p-values validate that our formulation has a significant effect on improving the performance against SHOT.

Ablation Study. To take a further step into different components within our method. We report the results of 5 AAA variations on **Office-31** under the SFUDA setting in Table 9. Specifically, we remove $\mathcal{L}_{adv}$, MI and $\mathcal{L}_{cls}$ from the full model respectively. The results indicate that each component is beneficial to promote the adaptation performance, and they work better together. Besides, we investigate the influence of our elaborate perturbations by comparing with the random perturbations, which are random Gaussian noises with the same magnitude (i.e., 10). It shows that result of random perturbations based AAA is almost equal to the one without adversarial training (i.e. w/o $\mathcal{L}_{adv}$). This empirically verifies the effectiveness of our perturbations.

Qualitative Results. To make an intuitive understanding of the results, we randomly select some samples in **Office-Home** for DG task $\mathfrak{R} \rightarrow A$ that are wrongly classified by RSC [94] while correctly classified by our method in Fig. 7. It can be seen that our method can correctly predict both the easy ones like *TV, Mouse* and hard ones like *shelf, paper clip*, etc.

We also compare the original images with perturbed images in Fig. 5.5. We can see that when $\epsilon < 10$, the perturbations are almost imperceptible. When $\epsilon = 100$, the images begin to be noisy, and when $\epsilon = 200$, the images have been seriously indistinct, the original semantic information has almost been lost. This can explain why our method works poorly in large perturbations.

Feature Visualization. We provide the t-SNE [98] based feature learned by our method for both SFUDA task ($A \rightarrow W$ on **Office-31**) and DG task ($\mathfrak{R} \rightarrow S$ on **Office-Home**) in Fig. 9 and Fig. 10, respectively. As shown in Fig. 9(a) and 9(c), the target features present a disorganized distribution when source-only pre-trained model is applied. In Fig. 9(b) and 9(d), it is clear that our method improves the discriminability of the target data. Instead, the discriminability of the source data seem to decrease, since handling source data is not our task in SFUDA. Our goal is to adapt the model gradually to the target domain, which can be verified in Fig. 9(d). Similarly, the feature representations for DG task $\mathfrak{R} \rightarrow S$ in Fig. 10(a) and 10(b) demonstrate that our method can learn more discriminative features compared with EISNet [52]. From Fig. 10(c) and 10(d), we can find that our method tries to explore the latent distributions with the known distributions instead of merely pursue invariant representations across multiple sources domains, which justifies our motivation.

Effect of Different Adversarial Attack Strategies. We further compare our method with other classic adversarial attack methods in Table 10. We choose the representative multi-step gradient-based method PGD [12] as the baseline. From Table 10, we can see that our AAA significantly outperforms PGD for domain generalization. Although they have some similarities, our motivation is to explore the unseen distributions related with the seen ones rather than purely attacks. The randomness and generalization ability of DNNs nicely meet our demand and provide diverse and smooth adversarial samples for generalizing to the target domain.

6 Conclusions

In this paper, we propose a general adaptation method termed domain Adaptation by Adversarial Attacks (AAA)

to handle a challenging yet practical setting for unsupervised domain adaptation where the source data or the target data can be absent during training, which provides a new perspective to tackle domain shift problem. Specifically, we reveal the latent distributions related to the training distribution by adding diverse and smooth perturbations. The model is expected to learn discriminative and transferable feature representations and achieve domain adaptation by learning from these adversarial samples. In addition, We employ several regularization terms to promote the adaptation performance. For source data-absent UDA, we further present a mutual information maximization strategy to exploit knowledge from the source pre-trained model. These techniques enable our method to perform well in divergence-agnostic UDA as well as the typical UDA. Extensive experiments on 4 popular visual datasets demonstrate that our AAA is able to achieve comparable or better performance than other state-of-the-arts for all these three UDA scenarios.

REFERENCES

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[2] J. Li, E. Chen, Z. Ding, L. Zhu, K. Lu, and H. T. Shen, "Maximum density divergence for domain adaptation," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[3] M. Shao, D. Kit, and Y. Fu, "Generalized transfer subspace learning through low-rank constraint," *International Journal of Computer Vision*, vol. 109, no. 1-2, pp. 74–93, 2014.

[4] Z. Ding, S. Ming, and Y. Fu, "Latent low-rank transfer subspace learning for missing modality recognition," in *Twenty-eighth AAAI conference on artificial intelligence*, 2014.

[5] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105.

[6] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.

[7] R. Li, Q. Jiao, W. Cao, H.-S. Wong, and S. Wu, "Model adaptation: Unsupervised domain adaptation without source data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9641–9650.

[8] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6028–6039.

[9] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation," in *International Conference on Machine Learning*. PMLR, 2013, pp. 10–18.

[10] S. Zhao, M. Gong, T. Liu, H. Fu, and D. Tao, "Domain generalization via entropy regularization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[12] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[13] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," *arXiv preprint arXiv:1703.09387*, 2017.

[14] S. Scardapane and D. Wang, "Randomness in neural networks: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 2, p. e1200, 2017.

[15] D. Sarkar, "Randomness in generalization ability: a source to improve it," *IEEE transactions on neural networks*, vol. 7, no. 3, pp. 676–685, 1996.

[16] T. Tommasi, M. Lanzi, P. Russo, and B. Caputo, "Learning the roots of visual domain shift," in *European Conference on Computer Vision*. Springer, 2016, pp. 475–482.

[17] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 5, pp. 770–787, 2009.

[18] R. Wang, M. Utiyama, L. Liu, K. Chen, and E. Sumita, "Instance weighting for neural machine translation domain adaptation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1482–1488.

[19] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Transfer independently together: A generalized framework for domain adaptation," *IEEE transactions on cybernetics*, vol. 49, no. 6, pp. 2144–2155, 2018.

[20] J. Li, M. Jing, K. Lu, L. Zhu, and H. T. Shen, "Locality preserving joint transfer for domain adaptation," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6103–6115, 2019.

[21] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Heterogeneous domain adaptation through progressive alignment," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, pp. 1381–1391, 2018.

[22] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4893–4902.

[23] K. Sohn, S. Liu, G. Zhong, X. Yu, M.-H. Yang, and M. Chandraker, "Unsupervised domain adaptation for face recognition in unlabeled videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3210–3218.

[24] Y. Zou, Z. Yu, B. Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 289–305.

[25] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5001–5009.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[27] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (cmd) for domain-invariant representation learning," *arXiv preprint arXiv:1702.08811*, 2017.

[28] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[29] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *International conference on machine learning*. PMLR, 2017, pp. 2208–2217.

[30] S. Li, C. H. Liu, Q. Lin, Q. Wen, L. Su, G. Huang, and Z. Ding, "Deep residual correction network for partial domain adaptation," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[31] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Advances in Neural Information Processing Systems*, 2018, pp. 1640–1650.

[32] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[33] J. Li, E. Chen, Z. Ding, L. Zhu, K. Lu, and Z. Huang, "Cycle-consistent conditional adversarial transfer networks," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 747–755.

[34] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," *arXiv preprint arXiv:1606.07536*, 2016.

[35] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[36] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2849–2857.

[37] Y. Kim, S. Hong, D. Cho, H. Park, and P. Panda, "Domain adaptation without source data," *arXiv preprint arXiv:2007.01524*, 2020.

[38] J. N. Kundu, N. Venkat, R. V. Babu *et al.*, "Universal source-free domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4544–4553.

[39] Y. Kim, D. Cho, P. Panda, and S. Hong, "Progressive domain adaptation from a source pre-trained model," *arXiv preprint arXiv:2007.01524*, 2020.

[40] S. Yang, Y. Wang, J. van de Weijer, L. Herranz, and S. Jui, "Unsupervised domain adaptation without source data by casting a bait," *arXiv preprint arXiv:2010.12427*, 2020.

[41] J. Liang, R. He, Z. Sun, and T. Tan, "Distant supervised centroid shift: A simple and efficient approach to visual domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2975–2984.

[42] Y. Hou and L. Zheng, "Source free domain adaptation with image translation," *arXiv preprint arXiv:2008.07514*, 2020.

[43] L. Duan, D. Xu, and I. W.-H. Tsang, "Domain adaptation from multiple sources: A domain-dependent regularization approach," *IEEE Transactions on neural networks and learning systems*, vol. 23, no. 3, pp. 504–518, 2012.

[44] L. Niu, W. Li, and D. Xu, "Multi-view domain generalization for visual recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4193–4201.

[45] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5542–5550.

[46] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales, "Episodic training for domain generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1446–1455.

[47] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi, "Generalizing across domains via cross-gradient training," *arXiv preprint arXiv:1804.10745*, 2018.

[48] R. Volpi, H. Namkoong, O. Sener, J. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," *arXiv preprint arXiv:1805.12018*, 2018.

[49] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5400–5409.

[50] T. Matsuura and T. Harada, "Domain generalization using a mixture of multiple latent domains," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 749–11 756.

[51] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2229–2238.

[52] S. Wang, L. Yu, C. Li, C.-W. Fu, and P.-A. Heng, "Learning from extrinsic and intrinsic supervisions for domain generalization," in *European Conference on Computer Vision*. Springer, 2020, pp. 159–176.

[53] Z. Huang, H. Wang, E. P. Xing, and D. Huang, "Self-challenging improves cross-domain generalization," in *ECCV*, 2020.

[54] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[55] C. Heinze-Deml and N. Meinshausen, "Conditional variance penalties and domain shift robustness," *Machine Learning*, pp. 1–46, 2020.

[56] A. Sinha, H. Namkoong, and J. Duchi, "Certifiable distributional robustness with principled adversarial training," *arXiv preprint arXiv:1710.10571*, vol. 2, 2017.

[57] F. Farnia, J. M. Zhang, and D. Tse, "Generalizable adversarial training via spectral normalization," *arXiv preprint arXiv:1811.07457*, 2018.

[58] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," *arXiv preprint arXiv:2001.03994*, 2020.

[59] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *arXiv preprint arXiv:1801.02610*, 2018.

[60] X. Liu and C.-J. Hsieh, "Rob-gan: Generator, discriminator, and adversarial attacker," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 234–11 243.

[61] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1979–1993, 2018.

[62] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu, "On the convergence and robustness of adversarial training." in *ICML*, vol. 1, 2019, p. 2.

[63] F. Utrera, E. Kravitz, N. B. Erichson, R. Khanna, and M. W. Mahoney, "Adversarially-trained deep nets transfer better," 2021.

[64] A. Shafahi, P. Saadatpanah, C. Zhu, A. Ghiasi, C. Studer, D. Jacobs, and T. Goldstein, "Adversarially robust transfer learning," 2020.

[65] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *arXiv preprint arXiv:1810.00069*, 2018.

[66] S. Li, F. Lv, B. Xie, C. H. Liu, J. Liang, and C. Qin, "Bi-classifier determinacy maximization for unsupervised domain adaptation," *arXiv preprint arXiv:2012.06995*, 2020.

[67] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, B. Tran, and A. Madry, "Adversarial robustness as a prior for learned representations," *arXiv preprint arXiv:1906.00945*, 2019.

[68] R. Ranganath, S. Gerrish, and D. Blei, "Black box variational inference," in *Artificial intelligence and statistics*. PMLR, 2014, pp. 814–822.

[69] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *International Conference on Machine Learning*. PMLR, 2018, pp. 531–540.

[70] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[71] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[72] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010.

[73] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," *arXiv preprint arXiv:0902.3430*, 2009.

[74] C. Cortes and M. Mohri, "Domain adaptation in regression," in *International Conference on Algorithmic Learning Theory*. Springer, 2011, pp. 308–323.

[75] S. Sun, H. Shi, and Y. Wu, "A survey of multi-source domain adaptation," *Information Fusion*, vol. 24, pp. 84–92, 2015.

[76] C. Chen, Z. Zheng, X. Ding, Y. Huang, and Q. Dou, "Harmonizing transferability and discriminability for adapting object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8869–8878.

[77] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[78] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 4, pp. 801–814, 2018.

[79] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[80] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3723–3732.

[81] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Adversarial dropout regularization," *arXiv preprint arXiv:1711.01575*, 2017.

[82] C.-Y. Lee, T. Batra, M. H. Baig, and D. Ulbricht, "Sliced wasserstein discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 285–10 295.

[83] X. Chen, S. Wang, M. Long, and J. Wang, "Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation," in *International conference on machine learning*. PMLR, 2019, pp. 1081–1090.

[84] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1426–1435.

[85] S. Cui, S. Wang, J. Zhuo, L. Li, Q. Huang, and Q. Tian, "Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3941–3950.

[86] Z. Lu, Y. Yang, X. Zhu, C. Liu, Y.-Z. Song, and T. Xiang, "Stochastic classifiers for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9111–9120.

[87] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," *arXiv preprint arXiv:1602.04433*, 2016.
[88] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in neural information processing systems*, 2005, pp. 529–536.
[89] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European conference on computer vision*. Springer, 2010, pp. 213–226.
[90] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5018–5027.
[91] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "Visda: The visual domain adaptation challenge," *arXiv preprint arXiv:1710.06924*, 2017.
[92] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.
[93] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[94] Z. Huang, H. Wang, E. P. Xing, and D. Huang, "Self-challenging improves cross-domain generalization," *arXiv preprint arXiv:2007.02454*, vol. 2, 2020.
[95] Q. Dou, D. C. de Castro, K. Kamnitsas, and B. Glocker, "Domain generalization via model-agnostic learning of semantic features," in *Advances in Neural Information Processing Systems*, 2019, pp. 6450–6461.
[96] A. DInnocente and B. Caputo, "Domain generalization with domain-specific aggregation modules," in *German Conference on Pattern Recognition*. Springer, 2018, pp. 187–198.
[97] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.
[98] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.