From Point to Flow: Enhancing Unsupervised Domain Adaptation with Flow Classification

Lihua Zhou, Mao Ye*, *Senior Member, IEEE,* Nianxin Li, Song Tang, Xu-Qian Fan, Lei Deng, Zhen Lei, *Fellow, IEEE,* and Xiatian Zhu

***Abstract*—Unsupervised domain adaptation aims to transfer knowledge from a labeled source domain to an unlabeled target domain. Existing methods, whether based on distribution matching or self-supervised learning, often focus solely on classifying individual source samples, potentially overlooking discriminative information. To address this limitation, we propose FlowUDA, a novel plugin method that enhances existing UDA frameworks by constructing semantically invariant flows from individual source samples to corresponding target samples, forming cross-domain trajectories. By leveraging a diffusion network guided by ordinary differential equations, FlowUDA ensures these flows preserve the topological structure of the source domain, maintaining their distinguishability. Our method then classifies these flows by sampling points along them and transferring labels from source samples, effectively capturing spatial relationships between domains. In essence, FlowUDA transforms the traditional point-based classification on individual source samples into flow-based classification on flows, allowing the model to learn richer, more discriminative features that bridge the gap between source and target domains. Extensive experiments on standard benchmarks demonstrate that integrating FlowUDA into existing UDA methods leads to notable performance gains, highlighting its effectiveness in addressing domain shift challenges.**

***Index Terms*—Unsupervised domain adaptation, Diffusion, Cross-domain trajectory, Flow classification.**

I. Introduction

DEEP learning has achieved remarkable progress in computer vision [1]–[4], often using large amounts of labeled data. However, it is expensive to label data for every domain. Meanwhile, there exhibits distribution discrepancy between different domains, making the direction use of pre-existing models in new domains less effective [5]. Solving distribution discrepancy with existing annotated data from a source domain for an unlabeled target domain, a.k.a., unsupervised domain adaptation (UDA), is thus useful and critical [6]–[11].

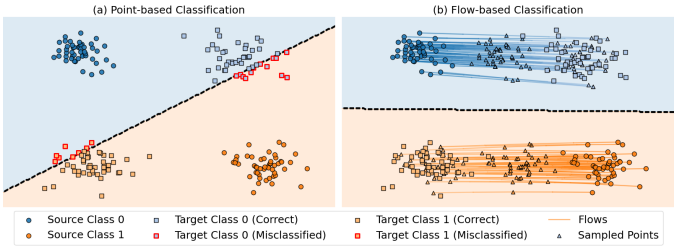


Fig. 1. Point-based classification vs. flow-based classification on real features from the Office-Home dataset (*Art* → *Clipart*, classes *bike* and *clock*). Features are extracted by ResNet-50 and visualized via t-SNE. (a) **Point-based Classification:** Traditional UDA methods classify only source features (circles), leading to poor generalization and misclassifications on target features (squares with red borders). (b) **Flow-based Classification:** FlowUDA constructs category-wise flows from source to target and classifies not only source features but also intermediate points sampled along these flows. This enables better alignment and accurate classification of target samples.

Existing UDA methods primarily adopt either a *distribution alignment* strategy [12]–[16]or rely on *self-supervised learning* [17]–[19]. However, both approaches focus on classifying individual source domain samples, thereby underutilizing the discriminative information inherent in the transitional distribution from the source to the target domain. This reliance solely on source-domain supervision can inadvertently limit the model's ability to generalize effectively to the target domain, potentially degrading its performance (see Fig. 1(a)). Unlike data augmentation techniques that produce static, linear interpolations, our flow constitutes a dynamic, category-consistent trajectory governed by ODEs, preserving topological structure and semantic labels throughout the source-to-target transition.

To address this issue, we build a flow for each source feature using ordinary differential equation (ODE). Each flow represents a trajectory from the source distribution to the target distribution. We theoretically prove that at any given point along these flows, the sampled features maintain the same topological structure as the source domain, ensuring that these constructed flows remain distinguishable. Moreover, we incorporate label information during training, ensuring that the constructed flow retains the same label as the initial

This work was supported by the National Natural Science Foundation of China (62276048), Chengdu Science and Technology Projects (2023-YF06-00009-HZ).

Lihua Zhou, Mao Ye and Nianxin Li are with School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, PR China. E-mail: lihuazhou120@gmail.com, maoye@uestc.edu.cn, linianxin1220@gmail.com

Song Tang is with the Institute of Machine Intelligence (IMI), University of Shanghai for Science and Technology, Shanghai 200093, China. E-mail: steventangsong@gmail.com

Xu-Qian Fan is with College of Information Science and Technoligy, Jinan Univerisity. E-mail: txqfan@jnu.edu.cn

Lei Deng is with Theory Lab, 2012 Labs, Huawei Technologies Co., Ltd. E-mail: deng.lei2@huawei.com

Zhen Lei is with the State Key Laboratory of Multimodel Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100190, China; the School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS), Beijing 100049, China; the Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science and Innovation, Chinese Academy of Sciences, Hong Kong, China. Email: zhen.lei@ia.ac.cn

Xiatian Zhu is with Surrey Institute for People-Centred Artificial Intelligence, CVSSP, University of Surrey, Guildford, UK. E-mail: xiatian.zhu@surrey.ac.uk

* corresponding author.

source features. By leveraging the distinguishability and label consistency of these flows, we shift from static supervision on individual source features to dynamic supervision on evolving trajectories. This transformation enables the model to learn richer semantic relationships between the source and target distributions. The key advantage of our approach lies in its ability to capture and utilize the transitional dynamics between domains, leading to more effective generalization to the target domain, as illustrated in Fig. 1(b).

Based on the above motivation, we propose **FlowUDA**, a novel plugin for unsupervised domain adaptation that transforms traditional point-based classification on source samples into flow-based classification on flows from source distribution to target distribution. FlowUDA consists of two parts. The first part is called ODE flow generation. To construct a category-wise flow connecting the source domain to the target domain, the target samples are first pseudo-labeled and the high confident pseudo-labeled target samples are selected. Based on the labeled source features and pseudo-labeled target features, a diffusion network is learned to construct flows based on ODE with the category consistency constraint. Another part is ODE flow classification. Unlike previous UDA methods constrained by supervision only on source data, FlowUDA classifies the constructed flows. This approach changes the paradigm from classifying static source-domain features to classifying the dynamic trajectories connecting the source and target domains. By doing so, FlowUDA enables spatial classification, capturing richer semantic relationships and transitional dynamics between domains.

Our contributions are summarized below: **(I)** We introduce FlowUDA, a novel plugin for unsupervised domain adaptation, which fundamentally transforms traditional point-based classification on source samples into flow-based classification on flows from source distribution to target distribution. By integrating FlowUDA into existing UDA methods, it enables the model to learn richer semantic relationships and capture the transitional dynamics between domains, effectively performing spatial classification. **(II)** We theoretically prove that at any given point along these flows, the sampled features maintain the same topological structure as the source domain, ensuring distinguishability and label consistency. This guarantees that the learned flows retain critical spatial information. **(III)** Extensive experiments validate that integrating FlowUDA into existing UDA methods leads to superior performance on standard benchmarks, demonstrating its effectiveness in enhancing the capabilities of current approaches.

II. Related Work

Unsupervised Domain Adaptation. UDA aims to transfer knowledge from the labeled source domain to the unlabeled target domain. Existing methods can be roughly divided into two strategies: distribution alignment strategy and self-supervised learning strategy. For distribution alignment strategy, it is hoped to find a domain invariant space where the distributions of source domain and target domain are aligned, which can be further roughly divided into five categories: *statistic moment matching*, *adversarial learning*, *optimal transport*, *bi-classifier adversarial learning* and *adversarial generation*. The methods based on *statistic moment matching* explicitly define a metric function of the distribution discrepancy and minimize it to align the distribution between two domains (*e.g*, DAN [12], CORAL [20] and CAN [13]). The methods based on *adversarial learning* are inspired by the generative adversarial network [21], which introduce a discriminator for domain classification and force the feature extractor to confuse the discriminator to learn domain invariant features (*e.g*, DANN [14], CDAN [15] and ADDA [22]). The methods based on *optimal transport* first connect each source sample and target sample by calculating a coupling matrix and then minimize the distance of these pair-wise connections (*e.g*, DeepJDOT [23], RWOT [24], ETD [25] and DeepHOT [26]). The methods based on *bi-classifier adversarial learning* plays a minimax game between feature extractor and two distinct classifiers during adaptation, which maximize the prediction discrepancy when optimizing two classifiers and minimize the prediction discrepancy when optimizing feature extractor (*e.g*, MCD [27], SWD [28] and CDAL [29]). The methods based on *adversarial generation* introduce a generator and a discriminator, which generate fake data to align the distribution between two domains at pixel-level (*e.g*, CoGAN [30], SimGAN [31] and CycleGAN [32]). In addition, recent work based on distribution alignment has also adopted other strategies. For example, HMA [33] improved the previous alignment from single-space to dual-space; SPA [34] uses graph matching for distribution alignment.

For the self-supervised learning strategy, it uses pseudo-labels or other self-supervision information to construct a training task to improve the generalization ability of the model during the adaptation. For example, ATDOC [18] applies memory mechanism or neighborhood aggregation to pseudo-label target samples. ssUDA [19] performs self-supervised tasks (*e.g*, rotation, flip and patch location predictions) to improve the model generalization. ICON [35] pursues an invariant classifier that is simultaneously consistent with the labels in source domain and the clusters in target domain. MSDTR [36] learns more compact features by a disentanglement and reconstruction process. FixBi [37] introduces a fixed ratio-based mixup to augment multiple intermediate domains between the source and target domain. Different from FixBi that generates intermediate domains samples with mixed categories, which are usually meaningless, FlowUDA uses a diffusion network to generate flow for each source feature with same semantic information.

Before deep learning era, there are also some related works based on manifold flow. GFK [38] constructed geodesic flows with a traditional kernel-based method. Later, [39] proposed to generate intermediate domains by taking a domainness variable as a conditional input, and transfers the images into the intermediate domains. These works are different from contemporary deep learning based domain adaptation methods which try to obtain domain invariant space. The distinguishability of the generated features is still not guaranteed. Other contemporary deep learning works related to flow and new DA tasks include: GGF [40] generates continuous intermediate domains to bridge the gap between source and target. They

employ the Wasserstein gradient flow in Kullback–Leibler divergence to transport samples from the source to the target domain, minimizing distribution discrepancy while preserving label information for gradual domain adaptation. TTTFlow [41] uses an unsupervised Normalizing Flow head to learn the normal distribution of latent features and detect domain shifts in test examples for efficient test-time adaptation. SSF [42] computes a Spline Flow on the Grassmann manifold to model flexible domain shifts. GFlowDA [43] proposes the Generalized Universal Domain Adaptation problem and utilizes generative flow networks for active and comprehensive target label prediction including un- known categories. UGDA [44] introduces the Partial Multi-view Learning task, tackling view-missing challenges in a low-data regime, which offers a relevant perspective on dealing with information scarcity. In contrast to all the aforementioned works, which primarily focus on minimizing distribution discrepancy or modeling domain shift at the point or subspace level, FlowUDA pioneers a novel paradigm: it first constructs a semantically invariant ODE flow for each sample and then performs classification on these cross-domain trajectories, capturing richer and more discriminative information.

Diffusion model. Recently, diffusion model has proposed a completely different framework from previous methods in the generation field. DDPM [45] is a parameterized Markov chain trained using variational inference to produce samples matching the data after finite time. SMLD [46], [47] estimates the score at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. Rectified flow [48] is trained with a simple and scalable unconstrained least squares optimization procedure. Due to the state-of-the-art performance, diffusion model has attracted attention from many fields, such as image editing [49], point cloud completion [50] and stochastic trajectory prediction [51]. Given an input image with user guide in a form of manipulating RGB pixels, SDEdit [49] first adds noise to the input, then subsequently denoises the resulting image through the SDE prior to increase its realism. ILVR [50] guides the generative process in DDPM to generate high-quality images based on a given reference image. MID [51] formulates the trajectory prediction task as a reverse process of motion indeterminacy diffusion, in which progressively discard indeterminacy from all the walkable areas until reaching the desired trajectory.

Recently, the field of domain adaptation has begun to introduce diffusion model. DTS [52] uses CDPM [53] to directly generate high fidelity and diversity samples which follow the distribution of target domain. DATUM [54] leverages text-to-image diffusion models [55] to generate a synthetic target dataset with photo-realistic images that faithfully depicts the style of target domain. DAD [56] proposes Domain-Adaptive Diffusion module accompanied by a mutual learning strategy, which can gradually convert data distribution from the source domain to the target domain while enabling the classification model to learn along the domain transition process. From these works, we can see that most of the current domain adaptation works based on diffusion model usually use the idea of DDPM [45]. The topological structure consistence of this diffusion flow are not theoretically guaranteed as ODE flow. And also they have not evolved from sample feature classification to flow classification.

III. Method

Problem statement. In UDA problem setting, there exist a labeled source domain $D_s = \{(\boldsymbol{x}_i^s, \boldsymbol{y}_i^s)\}_{i=1}^{n_s}$ with n_s labeled samples and a target domain $D_t = \{(\boldsymbol{x}_i^t)\}_{i=1}^{n_t}$ with n_t unlabeled samples. These two domains share the same label space $\{1, 2, \cdots, K\}$, but with different distributions. To solve the distribution shift between these two domains, we aim to train a model including a feature extractor F and a classifier C by using labeled data from the source domain and require it perform well on the unlabeled target domain.

Overview. As shown in Fig.2, FlowUDA consists of two parts: ODE flow generation and ODE flow classification. For ODE flow generation, prior to each epoch, we first assign pseudo-labels to the target samples based on a clustering algorithm [57]. Given a batch of source images $\{\boldsymbol{x}_i^s\}_{i=1}^b$ and target images $\{\boldsymbol{x}_i^t\}_{i=1}^b$, we use a common feature extractor F [1] to map these images to the feature space, $\boldsymbol{f}_i^{s/t} = F(\boldsymbol{x}_i^{s/t})$. Then based on the labels of the source domain and the pseudo-labels of the target domain, a multi-layer linear network s_θ is introduced to solve ordinary differential equation which simulates the flow from the source distribution to the target distribution by category. While for ODE flow classification, through s_θ, we can sample features on the ODE flow respect to each source feature. All the sampled features maintain the category information from source domain because the network s_θ is trained on a category-by-category basis. Finally, the model is retrained based on the sampled features, *i.e*, classifying ODE flow, which allows the trained model to not only identify the source distribution, but also have good generalization ability to other distributions derived from other sampling features.

A. ODE Flow Generation

Pseudo-labeling target domain. To construct a category-wise ODE flow, we first assign pseudo-labels to target samples based on a clustering algorithm [57]. Since the target domain is unlabeled, we cannot compute class-specific cluster centers directly from target samples. Instead, we initialize with source centers, which are semantically meaningful due to available labels. Specifically, in the first step, we sweep the source samples with current feature extractor F before each epoch to calculate source cluster centers $\{\boldsymbol{c}_k^s\}_{k=1}^K$. In the second step, the source cluster centers are initialized as the target cluster centers $\{\boldsymbol{c}_k^t\}_{k=1}^K$. In the third step, each target sample calculates the distance to each target cluster center to get its pseudo-label $\hat{\boldsymbol{y}}_i^t = \arg\min_k \cos(\boldsymbol{f}_i^t, \boldsymbol{c}_k^t)$, where the $\cos(\cdot, \cdot)$ is the cosine distance function. In the fourth step, the target cluster centers can be updated based on the current pseudo-labels. The third and fourth steps are repeated until the algorithm converges. Finally, all target samples can get their pseudo labels $\{\hat{\boldsymbol{y}}_i^t\}_{i=1}^{n_t}$. Then the distance between each target sample and its corresponding cluster center is calculated as $dist_i = 0.5 * (1 - \cos(\boldsymbol{f}_i^t, \boldsymbol{c}_{\hat{\boldsymbol{y}}_i^t}^t))$. To reduce the impact of

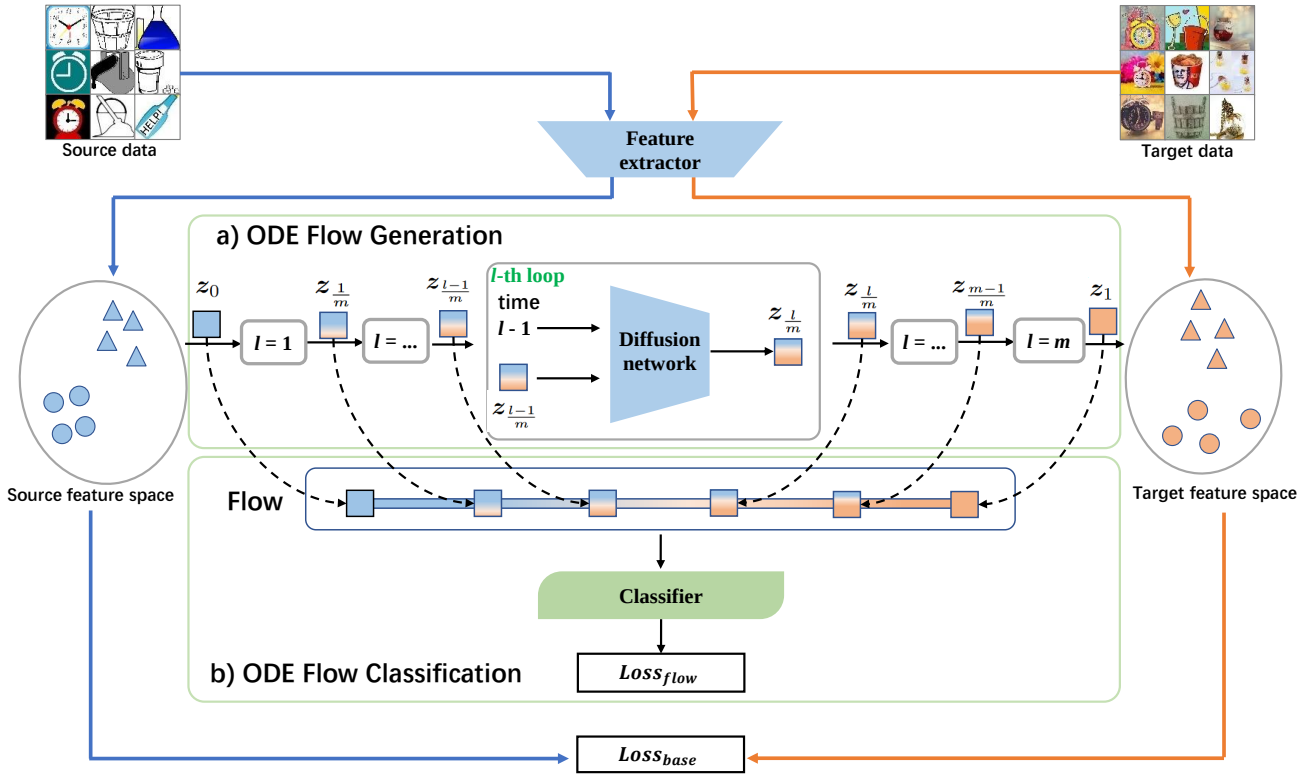


Fig. 2. Overview of the proposed FlowUDA method. (a) ODE flow generation: A diffusion network s_θ is introduced to simulate the ODE flow connecting the source distribution to the target distribution. (b) ODE flow classification: The feature exactor and classifier is trained by classifying features sampled along the ODE flow, optimizing the $Loss_{flow}$. Furthermore, FlowUDA can be seamlessly integrated into other UDA methods as a plugin. By integrating FlowUDA, these methods retain their existing $Loss_{base}$ while replacing their point-based classification with our flow-based classification.

noisy pseudo-labels, we filter out samples with $dist_i > \tau$ and only use high-confidence ones for training s_θ. Here τ is a hyperparameter.

Constructing straight flow. Since our operation is in the feature space, we define the source feature subset as $D^F_{s,k} = \{\boldsymbol{f}_i^{s,k}\}_{i=1}^{n_k^s}$ where $\boldsymbol{f}_i^{s,k}$ means the i-th source feature with label k, and the target feature subset $D^F_{t,k} = \{\boldsymbol{f}_i^{t,k}\}_{i=1}^{n_k^t}$ where $\boldsymbol{f}_i^{t,k}$ means the i-th selected target feature with pseudo-label k. n_k^s and n_k^t are the cardinalities of the corresponding subsets respectively. We further define the source feature set $D_s^F = \bigcup_{k=1}^K D^F_{s,k}$ and the selected target feature set $D_t^F = \bigcup_{k=1}^K D^F_{t,k}$. Based on these definition, we introduce a simpler straight flow $\Psi(l|\boldsymbol{f}_0, \boldsymbol{f}_1)$ for any arbitrary features $\boldsymbol{f}_0$ sampled from $P(D_s^F)$ and $\boldsymbol{f}_1$ sampled from $P(D_t^F)$. This straight flow is defined as follows:

$$\boldsymbol{f}_l = \Psi(l|\boldsymbol{f}_0, \boldsymbol{f}_1) = l * \boldsymbol{f}_1 + (1-l) * \boldsymbol{f}_0 \text{ for } 0 \le l \le 1, \quad (1)$$

where $\boldsymbol{f}_l$ denotes the intermediate feature along a straight flow from source feature $\boldsymbol{f}_0$ to target feature $\boldsymbol{f}_1$, parameterized by time $l \in [0, 1]$.

Constructing ODE Flow. Compared to straight flow, ODE flow has the potential to better preserve the topological structure of data due to its continuous and smooth nature. Specifically, an ODE flow defines a trajectory $\boldsymbol{z}_l = \Phi(l|\boldsymbol{z}_0)$ for $l \in [0, 1]$, starting from a source sample $\boldsymbol{z}_0$ ($\boldsymbol{z}_0 = \boldsymbol{f}_0$ sampled from $P(D_s^F)$) and ending at $\boldsymbol{z}_1$ which follows the distribution $P(D_t^F)$. Here $\boldsymbol{z}_l$ represents a point at time l along the flow, corresponding to features in an intermediate domain. Importantly, this function must be continuous, with $\boldsymbol{z}_0 = \Phi(0|\boldsymbol{z}_0)$ and $\boldsymbol{z}_1 = \Phi(1|\boldsymbol{z}_0)$. To model this ODE flow, we introduce a linear network s_θ to generate velocity fields such that:

$$\frac{\partial \Phi(l|\boldsymbol{z}_0)}{\partial l} = s_\theta(\boldsymbol{z}_l, l). \quad (2)$$

Since obtaining paired samples $\{(\boldsymbol{z}_0, \boldsymbol{z}_1)\}$, which are the two endpoints of an ODE flow, is not feasible, we cannot perform supervised training on the s_θ network. Intuitively, our goal is for the trained s_θ to find $\boldsymbol{z}_1$ such that the pair $\{(\boldsymbol{z}_0, \boldsymbol{z}_1)\}$ minimizes the loss $\mathbb{E}_{(\boldsymbol{z}_0,\boldsymbol{z}_1)\sim(D_s^F, D_t^F)}[c(\boldsymbol{z}_1 - \boldsymbol{z}_0)]$ for any convex distance function $c : R^d \to R$. This encourages the network to learn a transformation that reduces the conversion loss between the source and target distributions.

In Theorem 1, which is proved in Sec. IV, we found if s_θ is trained to fit the diffusion process of straight flow $\Psi(l|\boldsymbol{f}_0, \boldsymbol{f}_1)$, where $\boldsymbol{f}_0$ is randomly sampled from source domain and $\boldsymbol{f}_1$ is randomly sampled from target domain, then the trained network s_θ will reach a target feature $\boldsymbol{z}_1$ at time $l = 1$ for $\boldsymbol{z}_0 = \boldsymbol{f}_0$ such that $\boldsymbol{z}_1$ will achieve the minimum of $\mathbb{E}_{(\boldsymbol{z}_0,\boldsymbol{z}_1)\sim(D_s^F, D_t^F)}[c(\boldsymbol{z}_1 - \boldsymbol{z}_0)]$ for any convex function $c : R^d \to R$.

Based on Theorem 1, at each iteration where $\boldsymbol{f}_0$ and $\boldsymbol{f}_1$ are sampled from D_s^F and D_t^F, and a time information l is further sampled from the uniform distribution over (0, 1), the core part

is training the network s_θ to fit the gradient $\frac{\partial \Psi(l|\boldsymbol{f}_0,\boldsymbol{f}_1)}{\partial l}$ at time l. That is, the network $s_\theta(\boldsymbol{f}_l, l)$ will be trained to minimize following loss function:

$$\mathbb{E}_{(\boldsymbol{f}_0,\boldsymbol{f}_1)\sim(D_s^F,D_t^F)}\mathcal{L}^c\left(s_\theta(\boldsymbol{f}_l,l),\frac{\partial\Psi(l|\boldsymbol{f}_0,\boldsymbol{f}_1)}{\partial l}\right), \quad (3)$$

where $\mathcal{L}^c(\cdot,\cdot)$ represents a consistency constraint such as L_1-Norm or L_2-Norm. Here, L_2-Norm is selected. For network s_θ, it is a multi-layer network with the inputs as the current feature $\boldsymbol{f}_l$ calculated by Eq. (1) with time information l, and the output as the gradient of the current straight flow. From Eq. (1), it follows that:

$$\frac{\partial\Psi(l|\boldsymbol{f}_0,\boldsymbol{f}_1)}{\partial l}=\boldsymbol{f}_1-\boldsymbol{f}_0. \quad (4)$$

In addition, we hope that the learned networks s_θ can be used to implement a diffusion process by category, so the category information k is further input into the s_θ, and the features $\boldsymbol{f}_0, \boldsymbol{f}_1$ are also sampled from the corresponding k-th category subsets. The training loss is defined as follows,

$$Loss_s=\mathbb{E}_{(\boldsymbol{f}_0,\boldsymbol{f}_1)\sim(D_{s,k}^F,D_{t,k}^F)}\|s_\theta(\boldsymbol{f}_l,l,k)-(\boldsymbol{f}_1-\boldsymbol{f}_0)\|^2. \quad (5)$$

Remark: During the training, each $\boldsymbol{f}_0$ will meet different $\boldsymbol{f}_1$ to construct straight flow across epochs due to randomly sampling. Therefore, the trained network s_θ is essentially to fit the expected gradient of the straight flow passing through $\boldsymbol{f}_l$ as required by Theorem 1. Consequently, the trained network s_θ learns more general flow relationships from the source to the target domain. In addition, the network s_θ learns the expected gradient over many random source-target pairs, making it robust to individual mislabeled samples. Moreover, previous works on diffusion-based generation have shown that the generated $\boldsymbol{z}_1$ has the same distribution as $\boldsymbol{f}_1$ [48]. This implies that the generated $\boldsymbol{z}_1$ matches the distribution of the target domain. Therefore, after training, the output of s_θ can be regraded as the gradient of the ODE flow that diffuses feature from the source distribution to the target distribution.

Since the trained s_θ network calculates the gradient at any time, and each feature can find the next feature based on the gradient. That is

$$\boldsymbol{z}_{l+\Delta l}=\boldsymbol{z}_l+\Delta l\ s_\theta(\boldsymbol{z}_l,l,k). \quad (6)$$

By repeating this process, a trajectory from source distribution to the target distribution can be formed given a source feature $\boldsymbol{f}_0$ where $\boldsymbol{z}_0=\boldsymbol{f}_0$, and the trajectory calculated by s_θ is essentially the ODE flow $\Phi(l|\boldsymbol{z}_0)$ we want to find.

B. ODE Flow Classification

Sample generation along ODE flow. In the previous section, we described how to construct an ODE flow for each source feature from source distribution to the target distribution. Theoretically, we can ensure that such ODE flows retain distinguishable characteristics similar to those of the source samples, as proven in Theorem 2 in Sec. IV. Therefore, in this section, our core focus is on describing how we classify these flows.

Essentially, we classify the ODE flows by sampling intermediate points along them and classifying these sampled features. Specifically, for each source feature $\boldsymbol{f}_i^s$ with label $\boldsymbol{y}_i^s$, we generate a sequence of features along the ODE flow starting from $\boldsymbol{z}_0=\boldsymbol{f}_i^s$ at time $l=0$. Let $\Delta l=1/m$. We perform m iterative steps to sample features at times $l=\frac{1}{m},\frac{2}{m},\ldots,1$. The update rule for the l-th step is:

$$\boldsymbol{z}_{\frac{l}{m}}=\boldsymbol{z}_{\frac{l-1}{m}}+\Delta l\cdot s_\theta\left(\boldsymbol{z}_{\frac{l-1}{m}},\frac{l-1}{m},\boldsymbol{y}_i^s\right) \quad (7)$$

where $\boldsymbol{z}_{\frac{l-1}{m}}$ denotes the feature sampled at time $(l-1)/m$, and s_θ is the diffusion network that models the flow dynamics. All sampled features $\{\boldsymbol{z}_{\frac{l}{m}}\}_{l=1}^m$ constitute the set $\boldsymbol{flow}_i^s$, which is used for flow-based classification.

Since the generated ODE flows preserve distinguishability, as proven in Theorem 2, the label information of all generated features along the flow remains unchanged. That is, the set $\boldsymbol{flow}_i^s$ retains the label $\boldsymbol{y}_i^s$, which is identical to that of the source feature $\boldsymbol{f}_i^s$. Therefore, based on the generated samples and their original category labels, we define the flow classification loss as follows:

$$Loss_{flow}=\mathbb{E}_{\boldsymbol{z}\in\boldsymbol{flow}_i^s}\mathcal{L}^{ce}(C(\boldsymbol{z}),\boldsymbol{y}_i^s), \quad (8)$$

where $\mathcal{L}^{ce}(\cdot,\cdot)$ means cross-entropy loss function.

Remark: Although FlowUDA does not explicitly use target domain features for classification, the generated feature $\boldsymbol{z}_1$ by s_θ as the same distribution as the selected target feature, as proven in [48]. This implies that when classifying these flows, the classifier will automatically adapt to the target domain even without using target data.

Embedding ODE flow classification loss. The loss function optimized by existing UDA methods can essentially be divided into two parts: one is the source domain supervision loss, and the other is either the distribution alignment loss for distribution alignment strategies or the auxiliary self-supervised tasks for self-supervised learning strategies. The flow classification loss introduced in this work refines the source domain supervision loss. Therefore, FlowUDA can be seamlessly integrated with existing UDA methods as a plugin. Specifically, at each iteration, apply UDA methods to calculate $Loss_{base}$, such as distribution discrepancy minimization for distribution alignment strategy or auxiliary self-supervised tasks for self-supervised learning strategy. Next, the network s_θ is trained based on Eq. (5). Finally, we calculate $Loss_{flow}$ and combine it with $Loss_{base}$ to optimize the feature extractor F and classifier C based on the following loss

$$\min_{F,C} Loss_{base}+\alpha Loss_{flow}, \quad (9)$$

where α is a trade-off hyperparameter. Specifically, the conditional distribution alignment and pseudo-labeling strategies are used to calculate $Loss_{base}$ in our experiments respectively. The full algorithm is summarized in Algorithm 1.

Remark: Previous UDA methods essentially minimize the distribution alignment loss or auxiliary self-supervised loss, along with point-based classification loss on the source domain samples, expressed as $Loss_{base}+\mathcal{L}^{ce}(C(\boldsymbol{f}_i^s),\boldsymbol{y}_i^s)$, to train the model. In contrast, FlowUDA extends this point-based classification to flow-based classification, i.e., $Loss_{base}+Loss_{flow}$. By leveraging these flows, we generate many intermediate

Algorithm 1 FlowUDA

Input: Source domain $D_s = \{(\boldsymbol{x}_i^s, \boldsymbol{y}_i^s)\}_{i=1}^{n_s}$, target domain $D_t = \{(\boldsymbol{x}_i^t)\}_{i=1}^{n_t}$, the epoch number E, the iteration number N, hyperparameter τ, α, m.
Output: An adapted model.
Procedure:

```
for t = 1:E do
    Update pseudo labels of target domain {y_i^t}_{i=1}^{n_t} and
    filter a part of noise samples based on τ;
    for n = 1:N do
        Calculate Loss_base through any UDA method;
        Sample source and target samples by category;
        Get source features {f_i^s}_{i=1}^b and target features
        {f_i^t}_{i=1}^b based on the feature extractor F;
        Sample time information {l_i}_{i=1}^b from the uniform
        distribution over (0, 1);
        Calculate {f_{i,l_i}^s}_{i=1}^b based on Eq. (1);
        Optimize the network s_θ based on Eq. (5);
        Generate flow features based on Eq. (7) and
        calculate Loss_flow based on Eq. (8);
        Optimize the feature extractor F and classifier C
        based on Eq. (9);
    end for
end for
return Adapted model.
```

domain samples and target samples that retain the source labels. These generated samples carry semantic information from both the target and intermediate domains. Therefore, the trained model not only benefits from the enhanced feature space that preserves better semantic information due to flow distinguishability but also further reduces the error upper bound on the target domain, as proven in Sec. V. This approach ensures that the learned feature space maintains richer and more discriminative semantic information, leading to improved performance on the target domain.

IV. Theoretical Analysis

To ease presentation, we denote the learned ODE flow as $\Phi(l|\boldsymbol{z}_0)$, where $\boldsymbol{z}_0$ is the source feature, $\boldsymbol{z}_l$ is the generated feature at time l based on the diffusion network s_θ, and $\boldsymbol{f}_l$ means the feature calculated by straight flow for $\boldsymbol{f}_0$ and $\boldsymbol{f}_1$ at time l according to Eq. (1).

Theorem 1. Let the network s_θ be trained to fit the diffusion process of straight flow $\Psi(l|\boldsymbol{f}_0, \boldsymbol{f}_1)$, where $\boldsymbol{f}_0$ is randomly sampled from source domain and $\boldsymbol{f}_1$ is randomly sampled from target domain. That is, for $l \in [0, 1]$,

$$s_\theta(\boldsymbol{f}_l, l) = \mathbb{E}_{(\boldsymbol{f}_0, \boldsymbol{f}_1)\sim(D_s^F, D_t^F)}\left[\frac{\partial \Psi(l|\boldsymbol{f}_0, \boldsymbol{f}_1)}{\partial l}\middle| \boldsymbol{f}_l\right]$$

which is the average of the directions of the flows Ψ passing through point $\boldsymbol{f}_l$ at time l, where $\boldsymbol{f}_l$ is given in Eq. (1). Then the trained network s_θ will reach a target feature $\boldsymbol{z}_1$ at time l=1 for $\boldsymbol{z}_0 = \boldsymbol{f}_0$ such that $\boldsymbol{z}_1$ will achieve the minimum of $\mathbb{E}_{(\boldsymbol{z}_0, \boldsymbol{z}_1)\sim(D_s^F, D_t^F)}[c(\boldsymbol{z}_1 - \boldsymbol{z}_0)]$ for any convex function $c : R^d \to R$.

Proof. Suppose $\Phi(l|\boldsymbol{z}_0)$ where $l \in [0, 1]$ is the curve following s_θ which starts at the source feature $\boldsymbol{f}_0$. That is, $d\Phi(l|\boldsymbol{z}_0) = s_\theta dl$ with $\boldsymbol{z}_0 = \boldsymbol{f}_0$. We want to show that, for any $\boldsymbol{f}_1 \in D_t^F$, the inequality $\mathbb{E}[c(\boldsymbol{z}_1 - \boldsymbol{z}_0)] \le \mathbb{E}[c(\boldsymbol{f}_1 - \boldsymbol{f}_0)]$ holds for all convex functions $c : R^d \to R$. Using the fact $\mathbb{E}[(\boldsymbol{f}_1 - \boldsymbol{f}_0)] = \mathbb{E}[\mathbb{E}[(\boldsymbol{f}_1 - \boldsymbol{f}_0)|\boldsymbol{f}_l]]$ and Jensen's inequality, we have

$$\begin{aligned}\mathbb{E}[c(\boldsymbol{f}_1 - \boldsymbol{f}_0)] &= \int_0^1 \mathbb{E}[c(\boldsymbol{f}_1 - \boldsymbol{f}_0)]dl \\ &= \mathbb{E}\left[\int_0^1 \mathbb{E}[c(\boldsymbol{f}_1 - \boldsymbol{f}_0)|\boldsymbol{f}_l]dl\right] \\ &\ge \mathbb{E}\left[\int_0^1 c(\mathbb{E}[\boldsymbol{f}_1 - \boldsymbol{f}_0|\boldsymbol{f}_l])dl\right],\end{aligned}$$

where $\boldsymbol{f}_l$ means the generated feature by Eq. (1). Noting that

$$\mathbb{E}[\boldsymbol{f}_1 - \boldsymbol{f}_0|\boldsymbol{f}_l] = \mathbb{E}[\frac{\partial \Psi(l|\boldsymbol{f}_0, \boldsymbol{f}_1)}{\partial l}|\boldsymbol{f}_l] = s_\theta(\boldsymbol{f}_l, l),$$

by Jensen's inequality, we can get

$$\begin{aligned}\mathbb{E}\left[\int_0^1 c(\mathbb{E}[\boldsymbol{f}_1 - \boldsymbol{f}_0|\boldsymbol{f}_l])dl\right] &= \mathbb{E}\left[\int_0^1 c(s_\theta(\boldsymbol{f}_l, l))dl\right] \\ &\ge \mathbb{E}\left[c\left(\int_0^1 s_\theta(\boldsymbol{f}_l, l)dl\right)\right].\end{aligned}$$

From above, the definition of the ODE flow $\boldsymbol{z}_l$ implies that

$$\begin{aligned}\mathbb{E}[c(\boldsymbol{f}_1 - \boldsymbol{f}_0)] &\ge \mathbb{E}\left[c\left(\int_0^1 s_\theta(\boldsymbol{f}_l, l)dl\right)\right] \\ &= \mathbb{E}[c(\boldsymbol{z}_1 - \boldsymbol{f}_0)] \\ &= \mathbb{E}[c(\boldsymbol{z}_1 - \boldsymbol{z}_0)],\end{aligned}$$

which is the desired result. □

Theorem 2. The generated ODE flows preserve the topological structure of the source domain, maintaining their distinguishability. More precisely, let S represent the source feature space, $\boldsymbol{z}_l$ be the ODE flow, and F_L represent the intermediate feature space at time L, then the ODE flow derives naturally a map

$$\Phi : S \to F_L.$$

Suppose $A \subset S$ is an open subset corresponding to a specific category k in S, then the image of A under Φ, denoted by $B := \Phi(A)$ and corresponding to the same category k, is an open subset in F_L. Furthermore, the boundary of B is the image of the boundary of A under Φ. That is

$$\Phi(\partial A) = \partial B = \partial \Phi(A),$$

where ∂ denotes the boundary operator.

Proof. The last equation holds since $B = \Phi(A)$, so we only need to prove that $\Phi(\partial A) = \partial B$.

Step 1: Prove $\Phi(\partial A) \subset \partial B$.

For $x \in \partial A$, there exists a sequence $\{x_i\} \subset A$ such that $\lim_{i\to\infty} x_i = x$. Noting that Φ is continuous since the ODE flows are continuous with respect to initial data, we have

$$\Phi(x) = \Phi(\lim_{i\to\infty} x_i) = \lim_{i\to\infty} \Phi(x_i).$$

Since $x_i \in A$ and $B = \Phi(A)$, it follows that $\Phi(x_i) \in B$, then $\Phi(x)$ belongs to the closure $\overline{B}$ of B. On the other hand,

$x \notin A$ implies that $\Phi(x) \notin B$ since the ODE flows are disjoint. Hence we can obtain $\Phi(x) \in \partial B$. Therefore $\Phi(\partial A) \subset \partial B$.

Step 2: Prove $\partial B \subset \Phi(\partial A)$.

For $y \in \partial B$, there exists a sequence $\{y_i\} \subset B$ such that $\lim_{i\to\infty} y_i = y$. Since $B = \Phi(A)$, there exists a sequence $\{x_i\} \subset A$ such that $\Phi(x_i) = y_i$. Since Φ^{-1} is also continuous, we can get that $\{x_i\}$ has a limit point x. Suppose $x \in A$, then $\Phi(x) = \lim_{i\to\infty} \Phi(x_i) = \lim_{i\to\infty} y_i = y$, which means $y \in B$, which contradicts the assumption $y \in \partial B$. So $x \in \partial A$ and $\Phi(x) = y$. Hence $y \in \Phi(\partial A)$, then $\partial B \subset \Phi(\partial A)$.

Combining Step 1 and Step 2, we can obtain $\Phi(\partial A) = \partial B$. □

Remark: This means at time $l = L$, the generated feature $\boldsymbol{z}_L$ still distinguishable as the source domain. Therefore, the generated ODE flow, which is composed of these generated features, still distinguishable.

V. Upper Bound Analysis

The model trained by flow classification will further reduce error upper bound on target domain.

For a solution hypothesis $h \in \mathcal{H}$, the expected error on target domain $\mathcal{T}$ is generally bounded as [58]:

$$\mathcal{R}_{\mathcal{T}}(h) \leq \mathcal{R}_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) + \Gamma, \tag{10}$$

where $\mathcal{R}_{\mathcal{S}}(h)$ is the expected error on source domain $\mathcal{S}$, which will be minimized by classifying the start feature of the flow, where these features are essentially source features. $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T})$ represents the domain discrepancy between $\mathcal{S}$ and $\mathcal{T}$, which can be minimized by any unsupervised domain adaptation baseline method. Γ is the shared error across domains defined as $\Gamma = \min_{h\in\mathcal{H}}[\mathcal{E}_{\mathcal{S}}(h, l_{\mathcal{S}}) + \mathcal{E}_{\mathcal{T}}(h, l_{\mathcal{T}})]$ where $l_{\mathcal{S}}$ and $l_{\mathcal{T}}$ are domain-specific real labeling functions, and $\mathcal{E}_{\mathcal{S}/\mathcal{T}}(\cdot)$ measures the disagreement of two labeling functions with respect to a specific domain $\mathcal{S}/\mathcal{T}$.

Under the triangle inequality [58], we have $\mathbb{I}(a, b) \leq \mathbb{I}(a, c) + \mathbb{I}(c, b)$ where $\mathbb{I}(a, b)$ is an indicator function, which equal to 0 when $a = b$, and to 1 otherwise. A simple proof is shown as follows:

- If $c = a$, then $\mathbb{I}(a, c) + \mathbb{I}(c, b) = \mathbb{I}(a, a) + \mathbb{I}(a, b) \geq 0 + \mathbb{I}(a, b) = \mathbb{I}(a, b)$.
- If $c \neq a$, then $\mathbb{I}(a, c) + \mathbb{I}(c, b) = 1 + \mathbb{I}(c, b) \geq 1 \geq \mathbb{I}(a, b)$.

Based on this, we have:

$$\begin{aligned}
\Gamma &= \min_{h\in\mathcal{H}}[\mathcal{E}_{\mathcal{S}}(h, l_{\mathcal{S}}) + \mathcal{E}_{\mathcal{T}}(h, l_{\mathcal{T}})] \\
&= \min_{h\in\mathcal{H}}[\mathbb{E}_{x\sim\mathcal{S}}\mathbb{I}(h(x), l_{\mathcal{S}}(x)) + \mathbb{E}_{x\sim\mathcal{T}}\mathbb{I}(h(x), l_{\mathcal{T}}(x))] \\
&\leq \min_{h\in\mathcal{H}}[\mathbb{E}_{x\sim\mathcal{S}}\mathbb{I}(h(x), l_{\mathcal{S}}(x)) + \mathbb{E}_{x\sim\mathcal{T}}\mathbb{I}(h(x), \hat{l}_{\mathcal{T}}(x)) + \\
&\qquad \mathbb{E}_{x\sim\mathcal{T}}\mathbb{I}(\hat{l}_{\mathcal{T}}(x), l_{\mathcal{T}}(x))] \\
&= \min_{h\in\mathcal{H}}[\mathcal{E}_{\mathcal{S}}(h, l_{\mathcal{S}}) + \mathcal{E}_{\mathcal{T}}(h, \hat{l}_{\mathcal{T}}) + \mathcal{E}_{\mathcal{T}}(\hat{l}_{\mathcal{T}}, l_{\mathcal{T}})],
\end{aligned}$$

where $\hat{l}_{\mathcal{T}}$ is a pseudo-labeling function for domain $\mathcal{T}$. *First*, $\mathcal{E}_{\mathcal{S}}(h, l_{\mathcal{S}})$ quantifies the disagreement between h and $l_{\mathcal{S}}$ on $\mathcal{S}$. When we perform flow classification, the start feature on ODE flow is source feature. By performing supervised learning on source feature, h can be constrained to be close to $l_{\mathcal{S}}$ and hence restrict both disagreement to be small.

For the *Second* and *Third* terms, the disagreement between h and $\hat{l}_{\mathcal{T}}$ on $\mathcal{T}$, $\mathcal{E}_{\mathcal{T}}(h, \hat{l}_{\mathcal{T}})$, will be minimized by classifying the end feature of flow, where these features fit the distribution high-confidence pseudo-labeled target features [48], while the disagreement between $l_{\mathcal{T}}$ and $\hat{l}_{\mathcal{T}}$ on $\mathcal{T}$, $\mathcal{E}_{\mathcal{T}}(l_{\mathcal{T}}, \hat{l}_{\mathcal{T}})$, is minimized by the pseudo-label strategy by inclining to those selected samples with high confident being correctly predicted during the domain adaptation. In addition, compared to traditional pseudo-labeling strategy, more samples with labels are generated from source features based on the learned ODE flow Φ, thus, the accuracy of pseudo-labeling target samples will be improved. The better $\hat{l}_{\mathcal{T}}$ can be achieved, thus the errors of $\mathcal{E}_{\mathcal{T}}(l_{\mathcal{T}}, \hat{l}_{\mathcal{T}})$ and $\mathcal{E}_{\mathcal{T}}(h, \hat{l}_{\mathcal{T}})$ will be further reduced.

Traditional methods usually focus only on the source and target domains when optimizing the upper bound, which may ignore the discriminative information of the intermediate domain as we mentioned in the maintext. Our FlowUDA essentially makes up for that. The details are as follows:

$$\min_{h\in\mathcal{H}}[\mathcal{E}_{\mathcal{S}}(h, l_{\mathcal{S}}) + \mathcal{E}_{\mathcal{T}}(h, l_{\mathcal{T}})] \leq \min_{h\in\mathcal{H}} \sum_{\mathcal{L}=\mathcal{S}}^{\mathcal{T}} \mathcal{E}_{\mathcal{L}}(h, l_{\mathcal{L}})$$

By minimizing each $\mathcal{E}_{\mathcal{L}}(h, l_{\mathcal{L}})$, FlowUDA enhances the model's generalization across the entire feature space. □

VI. Experiment

Datasets. To evaluate the performance enhancement provided by our plugin, we conduct experiments using four standard datasets with existing UDA methods. **Office-31** [72] is a standard dataset, which contains three domains: Amazon (A) downloaded from online website, Webcam (W) and DSLR (D) taken by web camera and digital SLR camera respectively. It contains 4652 images from 31 office environment categories. **Office-Home** [73] is a challenging dataset. It contains 15588 images of 65 categories from 4 different domains: Artistic images (A), Clip-Art images (C), Product images (P) and RealWorld images (R). **Visda-17** [74] is a large-scale dataset, which focus on a 12-category synthesis-to-real object classification task. The source domain contains 152,397 synthetic images and the target domain has 55,388 real object images. **DomainNet** [75] is one of the most challenging datasets in domain adaptation. It contains about 600 thousand images in 345 categories from 6 domains: Clipart (C), Infograph (I), Painting (P), Quickdraw (Q), Real (R) and Sketch (S).

Implementation details. Our experiments are all performed in Pytorch. To verify the robustness of the method, each task is performed 5 times. For fair comparison, the same backbone is adopted with other compared methods. The SGD optimizer is selected to update the model and the CosineAnnealingLR is used to update the learning rate of the SGD optimizer. For hyperparameter τ, it is set as 0.1/0.2/0.1/0.2 in Office-31, Office-Home, Visda-17 and DomainNet respectively. For hyperparameter α, it is set as 0.3 in all datasets respectively. For hyperparameter m, it is set as 5/5/3/5 in Office-31, Office-Home, Visda-17 and DomainNet respectively.

A. Comparison with the State-of-the-Art methods

The performance comparison with state-of-the-art methods on Office-31, Office-Home, VisDA-17, and DomainNet is

TABLE I
COMPARISON WITH THE STATE-OF-THE-ART METHODS ON *Office-31* DATASET. METRIC: CLASSIFICATION ACCURACY (%).

Method	Backbone	A→D	A→W	D→A	D→W	W→A	W→D	*avg*
ResNet-50 [1]	ResNet-50	68.9	68.4	62.5	96.7	60.7	99.3	76.1
ResNet-50+FlowUDA	ResNet-50	83.7±0.2	88.4±0.3	71.0±0.2	98.6±0.1	70.9±0.4	100.0±0.0	85.4
MLS [56]	ResNet-50	-	-	-	-	-	-	80.1
DTS [52]	ResNet-50	95.4	94.7	77.8	98.4	78.0	**100.0**	90.6
COT [9]	ResNet-50	96.1	96.5	76.7	99.1	77.4	**100.0**	91.0
DeepHOT [26]	ResNet-50	92.8	95.7	74.2	99.3	75.1	99.3	89.5
ALDA [59]	ResNet-50	94.0	95.6	72.2	97.7	72.5	**100.0**	88.7
CaCo [60]	ResNet-50	91.7	89.7	73.1	98.4	72.8	**100.0**	87.6
SUDA [61]	ResNet-50	91.2	90.8	72.2	98.7	71.4	**100.0**	87.4
DMAL [62]	ResNet-50	89.1	88.4	71.8	99.2	70.8	**100.0**	86.7
NSM [63]	ResNet-50	93.8	95.4	75.6	98.8	76.3	99.5	89.9
PSE	ResNet-50	92.7±0.4	90.4±0.2	74.2±0.2	97.5±0.1	72.1±0.2	99.6±0.1	87.8
PSE+FlowUDA	ResNet-50	94.4±0.2	92.5±0.3	75.8±0.1	99.1±0.1	75.3±0.2	99.8±0.0	89.5
CAN [13]	ResNet-50	95.0	94.5	78.0	99.1	77.0	99.8	90.6
CAN+FlowUDA	ResNet-50	96.0±0.2	96.0±0.2	78.6±0.3	99.4±0.1	77.4±0.1	**100.0±0.0**	91.2
ATDOC [18]	ResNet-50	94.4	94.5	75.6	98.9	75.2	99.6	89.7
ATDOC+FlowUDA	ResNet-50	95.2±0.2	95.1±0.1	76.6±0.2	99.2±0.1	75.7±0.3	**100.0±0.0**	90.3
HMA [33]	ResNet-50	92.4	95.1	73.7	99.2	72.8	**100.0**	88.9
HMA+FlowUDA	ResNet-50	93.8±0.3	96.0±0.3	75.2±0.2	99.2±0.1	73.9±0.1	**100.0±0.0**	89.7
GSDE [64]	ResNet-50	96.7	96.9	78.3	98.8	79.2	**100.0**	91.7
GSDE+FlowUDA	ResNet-50	97.7±0.3	97.6±0.1	79.1±0.3	99.8±0.2	80.3±0.2	**100.0±0.0**	92.4
CDTrans [65]	DeiT-base	97.0	96.7	81.1	99.0	81.9	**100.0**	92.6
CDTrans++FlowUDA	DeiT-base	**98.1±0.1**	97.4±0.3	**82.6±0.1**	99.8±0.0	82.7±0.3	**100.0±0.0**	93.4
PMTrans [66]	DeiT-base	96.5	99.0	81.4	99.4	82.1	**100.0**	93.1
PMTrans+FlowUDA	DeiT-base	97.5±0.3	**99.8±0.1**	82.3±0.4	**100.0±0.0**	**83.4±0.3**	**100.0±0.0**	**93.8**

TABLE II
COMPARISONS WITH THE STATE-OF-THE-ART METHODS ON *Office-Home* DATASET. METRIC: CLASSIFICATION ACCURACY (%).

Method	Backbone	A→C	A→P	A→R	C→A	C→P	C→R	P→A	P→C	P→R	R→A	R→C	R→P	*avg*
ResNet-50 [1]	ResNet-50	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
ResNet-50+FlowUDA	ResNet-50	43.9±0.3	72.7±0.2	76.3±0.3	61.7±0.2	69.0±0.3	74.1±0.2	60.3±0.4	43.7±0.3	77.0±0.1	65.8±0.4	45.0±0.3	75.8±0.2	63.8
MDD+IA [67]	ResNet-50	56.2	77.9	79.2	64.4	73.1	74.4	64.2	54.2	79.9	71.2	58.1	83.1	69.5
MetaAlign [68]	ResNet-50	59.3	76.0	80.2	65.7	74.7	75.1	65.7	56.5	81.6	74.1	61.1	85.2	71.3
MLS [56]	ResNet-50	-	-	-	-	-	-	-	-	-	-	-	-	52.4
COT [9]	ResNet-50	57.6	75.2	83.2	67.8	76.2	75.7	65.4	56.2	82.4	75.1	60.7	84.7	71.7
DeepHOT [26]	ResNet-50	57.0	77.3	81.9	66.5	77.4	78.0	63.8	54.8	81.5	73.4	60.0	84.5	71.3
ALDA [59]	ResNet-50	53.7	70.1	76.4	60.2	72.6	71.5	56.8	51.9	77.1	70.2	56.3	82.1	66.6
EIDCO [69]	ResNet-50	58.1	76.8	80.0	65.1	75.4	74.4	65.6	58.3	82.2	75.0	62.8	84.7	71.5
DMAL [62]	ResNet-50	48.1	70.6	76.6	60.8	67.7	68.8	62.5	51.2	78.1	73.3	54.0	81.0	66.1
NSM [63]	ResNet-50	56.8	76.5	80.3	66.6	75.6	75.9	63.6	53.9	81.0	73.3	57.9	83.8	70.4
PSE	ResNet-50	45.4±0.3	74.4±0.3	77.5±0.2	63.7±0.4	71.5±0.2	74.8±0.2	62.8±0.3	44.6±0.4	79.8±0.2	66.4±0.3	46.3±0.4	78.2±0.2	65.5
PSE+FlowUDA	ResNet-50	47.9±0.3	76.3±0.2	79.8±0.3	66.1±0.3	73.2±0.2	76.4±0.2	64.3±0.4	46.5±0.4	81.0±0.2	68.1±0.2	49.6±0.3	79.2±0.1	67.4
CAN [13]	ResNet-50	58.7	78.1	82.1	67.4	75.7	78.1	67.2	54.2	82.5	73.4	60.9	83.5	71.8
CAN+FlowUDA	ResNet-50	59.8±0.2	79.2±0.3	82.7±0.1	69.2±0.2	76.9±0.3	78.5±0.2	69.7±0.2	56.2±0.3	83.1±0.1	74.4±0.3	61.5±0.3	84.1±0.2	72.9
ATDOC [18]	ResNet-50	58.3	78.8	82.3	69.4	78.2	78.2	67.1	56.0	82.7	72.0	58.2	85.5	72.2
ATDOC+FlowUDA	ResNet-50	59.6±0.3	79.6±0.2	84.4±0.2	70.7±0.2	79.3±0.2	79.5±0.1	68.6±0.4	56.8±0.2	83.7±0.2	74.1±0.2	59.8±0.3	86.8±0.1	73.6
HMA [33]	ResNet-50	58.7	78.1	81.6	67.4	75.8	78.1	66.8	54.2	82.5	73.4	59.7	83.5	71.7
HMA+FlowUDA	ResNet-50	60.0±0.3	79.1±0.3	83.0±0.1	68.6±0.2	77.1±0.2	78.9±0.2	68.1±0.2	55.6±0.3	83.5±0.1	74.2±0.3	60.9±0.3	85.5±0.1	72.9
GSDE [64]	ResNet-50	57.8	80.2	81.9	71.3	78.9	80.5	67.4	57.2	84.0	76.1	62.5	85.7	73.6
GSDE+FlowUDA	ResNet-50	58.9±0.3	81.3±0.3	83.0±0.4	72.8±0.2	80.5±0.2	81.8±0.1	69.1±0.2	58.8±0.3	85.4±0.3	77.0±0.2	64.3±0.2	87.3±0.3	75.0
CDTrans [65]	DeiT-base	68.8	85.0	86.9	81.5	87.1	87.3	79.6	63.3	88.2	82.0	66.0	90.6	80.5
CDTrans++FlowUDA	DeiT-base	70.4±0.3	86.4±0.3	88.3±0.2	83.4±0.2	**89.2±0.2**	**89.2±0.2**	**81.4±0.3**	65.0±0.2	89.6±0.4	83.5±0.2	67.8±0.1	92.5±0.4	82.2
PMTrans [66]	DeiT-base	71.8	87.3	88.3	83.0	87.7	87.8	78.5	67.4	89.3	81.7	70.7	92.0	82.1
PMTrans+FlowUDA	DeiT-base	**73.4±0.2**	**89.0±0.3**	**89.6±0.2**	**84.7±0.2**	**89.2±0.3**	89.0±0.3	79.9±0.4	**68.9±0.2**	**91.1±0.2**	**83.6±0.3**	**72.1±0.3**	**93.4±0.2**	**83.7**

shown in Tables I–IV, respectively. We integrate FlowUDA into various base methods, including ResNet, PSE, CAN [13], ATDOC [18], GSDE [64], CDTrans [65], and PMTrans [66], demonstrating its plug-and-play compatibility across diverse UDA frameworks and backbones.

From the results, we observe the following: 1).FlowUDA consistently improves performance across all datasets and base methods. For example, on Office-31, PSE+FlowUDA, CAN+FlowUDA, and GSDE+FlowUDA achieve +1.7%, +0.6%, and +0.7% gains over their respective baselines; similar improvements are observed on Office-Home (+1.9%, +1.1%, +1.4%). 2). When integrated with strong recent meth-

TABLE III
COMPARISON WITH THE STATE-OF-THE-ART METHODS ON *Visda-17* DATASET. METRIC: PER-CLASS CLASSIFICATION ACCURACY (%).

Method	Backbone	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	*avg*
ResNet-101 [1]	ResNet-101	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
ResNet-101+FlowUDA	ResNet-101	72.1	70.2	65.2	69.3	85.5	72.3	88.6	61.9	85.9	63.1	80.0	37.7	71.0
BCDM [70]	ResNet-101	95.1	87.6	81.2	73.2	92.7	95.4	86.9	82.5	95.1	84.8	88.1	39.5	83.4
DWL [71]	ResNet-101	90.7	80.2	86.1	67.6	92.4	81.5	86.8	78.0	90.6	57.1	85.6	28.7	77.1
MLS [56]	ResNet-101	-	-	-	-	-	-	-	-	-	-	-	-	68.1
DTS [52]	ResNet-101	96.4	87.2	82.2	78.2	94.6	96.8	88.7	82.6	92.4	93.8	86.6	57.8	86.4
COT [9]	ResNet-101	96.9	89.6	84.2	74.1	96.4	96.5	88.6	82.0	96.0	94.1	85.1	62.1	87.1
DeepHOT [26]	ResNet-101	-	-	-	-	-	-	-	-	-	-	-	-	73.5
ALDA [59]	ResNet-101	93.8	74.1	82.4	69.4	90.6	87.2	89.0	67.6	93.4	76.1	87.7	22.2	77.8
CaCo [60]	ResNet-101	90.4	80.7	78.8	57.0	88.9	87.0	81.3	79.4	88.7	88.1	86.8	63.9	80.9
SUDA [61]	ResNet-101	88.3	79.3	66.2	64.7	87.4	80.1	85.9	78.3	86.3	87.5	78.8	**74.5**	79.8
EIDCo [69]	ResNet-101	-	-	-	-	-	-	-	-	-	-	-	-	83.8
NSM [63]	ResNet-101	93.8	81.8	83.0	66.0	93.2	81.8	89.0	78.5	92.1	77.1	89.7	38.3	80.4
PSE	ResNet-101	92.6	72.5	62.0	73.2	93.6	77.2	89.0	66.9	86.1	71.2	81.5	39.9	75.5
PSE+FlowUDA	ResNet-101	96.8	74.8	65.9	74.5	94.8	88.2	91.9	75.9	89.6	74.4	83.3	48.3	79.9
CAN [13]	ResNet-101	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
CAN+FlowUDA	ResNet-101	97.5	90.5	85.4	75.1	98.1	97.5	91.3	83.8	97.4	**96.6**	87.8	61.3	88.5
ATDOC [18]	ResNet-101	93.7	83.0	76.9	58.7	89.7	95.1	84.4	71.4	89.4	80.0	86.7	55.1	80.3
ATDOC+FlowUDA	ResNet-101	94.8	84.4	80.1	60.4	93.9	96.2	85.2	76.8	92.7	87.6	88.6	58.2	83.2
HMA [33]	ResNet-101	88.3	71.2	85.1	66.4	86.3	79.3	88.8	87.6	83.9	79.3	83.4	46.2	78.8
HMA+FlowUDA	ResNet-101	97.0	83.5	85.2	59.4	91.1	94.6	82.0	76.5	86.3	91.7	85.4	51.1	82.0
CDTrans [65]	DeiT-base	97.1	90.5	82.4	77.5	96.6	96.1	93.6	88.6	97.9	86.9	90.3	62.8	88.4
CDTrans++FlowUDA	DeiT-base	97.5	91.6	83.5	77.3	97.2	96.9	**95.9**	**90.0**	**98.1**	88.0	90.2	63.3	**89.1**
PMTrans [66]	DeiT-base	98.2	92.2	88.1	77.0	**97.4**	95.8	94.0	72.1	97.1	95.2	**94.6**	51.0	87.7
PMTrans+FlowUDA	DeiT-base	**98.4**	**94.5**	**90.8**	**79.1**	97.0	**97.4**	95.6	75.1	97.7	95.2	93.6	50.4	88.7

TABLE IV
COMPARISONS WITH THE STATE-OF-THE-ART METHODS ON *DomainNet* DATASET. METRIC: CLASSIFICATION ACCURACY (%); BACKBONE: RESNET-50. FOR EACH CROSS-DOMAIN PAIR, THE SOURCE/TARGET DOMAINS ARE SPECIFIED IN THE CORRESPONDING ROW/COLUMN FIELDS.

ResNet	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	14.2	29.6	9.5	43.8	34.3	26.3
inf	21.8	-	23.2	2.3	40.6	20.8	21.7
pnt	24.1	15.0	-	4.6	45.0	29.0	23.5
qdr	12.2	1.5	4.9	-	5.6	5.7	6.0
rel	32.1	17.0	36.7	3.6	-	26.2	23.1
skt	30.4	11.3	27.8	3.4	32.9	-	21.2
Avg.	24.1	11.8	24.4	4.7	33.6	23.2	20.3

MCD	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	15.4	25.5	3.3	44.6	31.2	24.0
inf	24.1	-	24.0	1.6	35.2	19.7	20.9
pnt	31.1	14.8	-	1.7	48.1	22.8	23.7
qdr	8.5	2.1	4.6	-	7.9	7.1	6.0
rel	39.4	17.8	41.2	1.5	-	25.2	25.0
skt	37.3	12.6	27.2	4.1	34.5	-	23.1
Avg.	28.1	12.5	24.5	2.4	34.1	21.2	20.5

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

SWD	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	14.7	31.9	10.1	45.3	36.5	27.7
inf	22.9	-	24.2	2.5	33.2	21.3	20.0
pnt	33.6	15.3	-	4.4	46.1	30.7	26.0
qdr	15.5	2.2	6.4	-	11.1	10.2	9.1
rel	41.2	18.1	44.2	4.6	-	31.6	27.9
skt	44.2	15.2	37.3	10.3	44.7	-	30.3
Avg.	31.5	13.1	28.8	6.4	36.1	26.1	23.6

CAN	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	17.3	38.4	8.6	53.2	39.7	31.4
inf	33.5	-	34.2	4.7	51.2	26.7	30.1
pnt	39.9	14.5	-	8.2	59.4	33.7	31.1
qdr	25.9	3.0	10.8	-	13.7	14.9	13.7
rel	52.4	16.9	46.3	3.9	-	41.9	32.3
skt	53.9	17.5	45.9	15.5	57.6	-	38.1
Avg.	41.1	13.8	35.1	8.2	47.0	31.4	29.5

CAN+FlowUDA	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	27.8	49.5	12.2	64.4	50.0	40.8
inf	40.0	-	39.5	8.8	62.8	31.4	36.5
pnt	47.7	18.8	-	16.5	64.7	39.3	37.3
qdr	32.5	8.8	24.0	-	21.6	22.5	21.9
rel	58.2	20.2	55.8	10.8	-	49.2	38.8
skt	61.8	23.3	51.2	18.2	61.5	-	43.2
Avg.	48.0	19.8	44	13.3	55	38.5	**36.4**

ods, FlowUDA achieves competitive or state-of-the-art results. Notably, PMTrans+FlowUDA obtains 93.8% on Office-31 and 83.7% on Office-Home, while GSDE+FlowUDA reaches 92.4% on Office-31 and 75.0% on Office-Home, outperforming many existing approaches. 3). The effectiveness of FlowUDA is independent of the base method or backbone architecture, as it brings consistent gains on both CNN-based and Transformer-based models, confirming its general applicability. 4). As expected, stronger base methods yield higher absolute performance (e.g., CAN+FlowUDA $>$ PSE+FlowUDA), but the relative improvement from FlowUDA remains stable, highlighting that our flow-based classification paradigm complements existing UDA strategies rather than replacing them.

These results validate that transforming point-based classification into flow-based classification is an effective and general-purpose enhancement for unsupervised domain adaptation.

B. Further Experiments

Visualization of feature distribution by t-SNE. To intuitively understand the proposed FlowUDA, we use t-SNE [76] to visualize the classification results on the task C→A in Office-home dataset based on two different methods: PSE and CAN. As shown in Fig. 3, from left to right, the images represent the visualization results of the PSE, PSE+FlowUDA, CAN and CAN+FlowUDA respectively. From Fig. 3, it can be seen that when FlowUDA is used, a huge improvement can be obtained compared with the baseline. It shows that the feature

Fig. 3. Feature distribution visualization using t-SNE. The results are based on the task C→A in Office-home dataset.

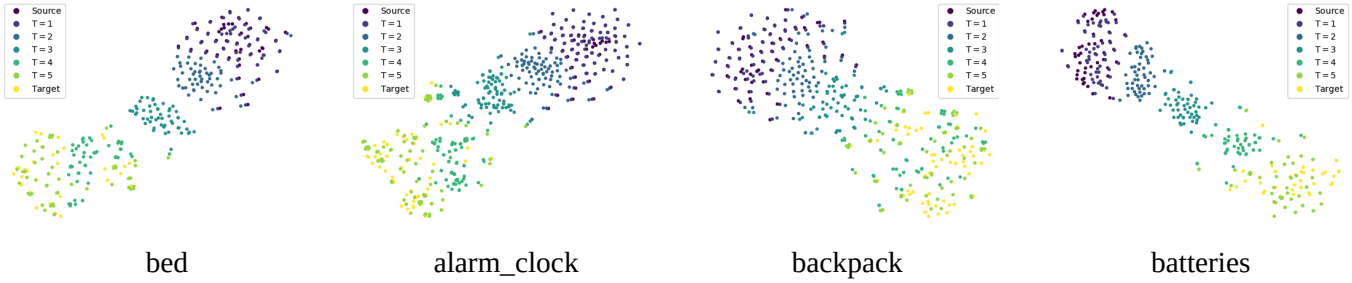


Fig. 4. Visualization of ODE flow using t-SNE. The results are based on the task C→A in Office-home dataset. Four categories as alarm clock, backpack, batteries, and bed are shown. *Zoom in for best view.*

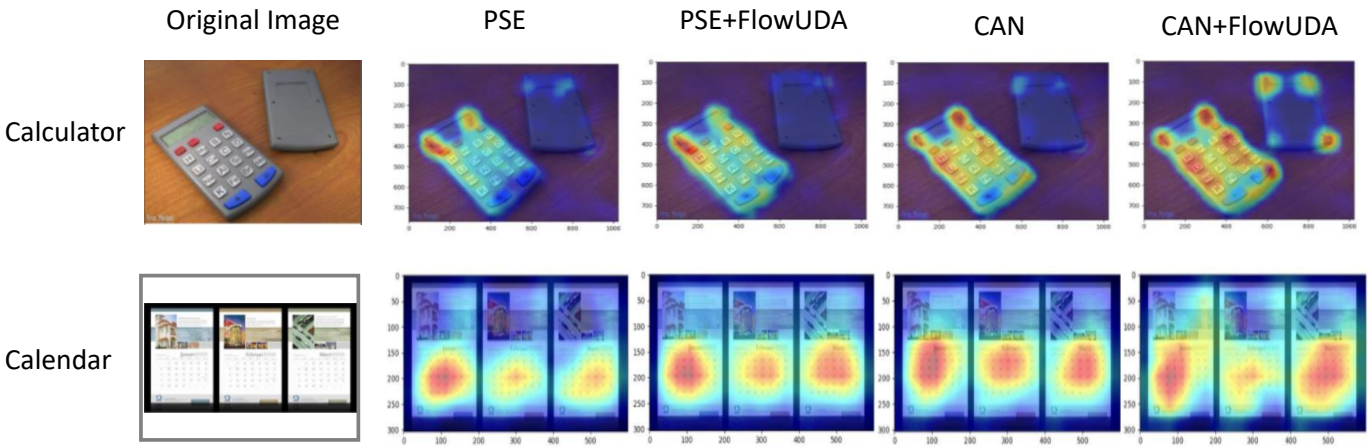


Fig. 5. Grad-CAM visualization of different methods on Office-Home task $C \rightarrow A$. Compared to PSE and CAN, our FlowUDA generates more focused and object-aligned attention maps, indicating stronger feature discrimination and reduced distraction from background noise.

space is more discriminate, thus more compact clusters can be obtained.

Visualization of ODE flow by t-SNE. To intuitively understand the ODE flow we generated, we visualize it using t-SNE. To better show how the source feature being transformed to the target domain, specifically, only the s_θ network is trained and the backbone (feature extractor and classifier) is frozen, so that the domain shift exists. We conduct experiments on the task C→A in Office-home dataset. A series of generated features based on s_θ network connecting the source domain to the target domain are shown. The diffusion flows of four categories as alarm clock, backpack, batteries, and bed are shown in Fig.4. It can be clearly seen that the generated features are constantly approaching the target domain along straight line like flow. Meanwhile, the end generated samples of ODE flows are embedded in the corresponding category clusters in the target domain.

Attention visualization by CAM. We further make visualization analysis by Grad-CAM [77] on Office-home task $C \rightarrow A$ in Fig.5. We randomly select 4 categories. For each category, one image is randomly selected for activation mapping visualization. A CAM heatmap can reflect the classification capability by visualizing the attention of the model to different regions of the input image, and the higher heat values in corresponding regions indicate more effective extraction of key features from the image. It is evident that, the attentions of PSE+FlowUDA and CAN+FlowUDA are more large compared with PSE and CAN respectively and

concentrate on the corresponding objects.

TABLE V
COMPARISON OF MIXUP, STRAIGHT FLOW, AND FLOWUDA ON *Office-31, Office-Home, and VisDA-17*.

Method	Office-31	Office-Home	VisDA-17
PSE+mixup	88.6	66.6	77.4
PSE+straight flow	88.8	66.9	78.1
PSE+FlowUDA	89.5	67.4	79.9
CAN+mixup	90.8	72.4	87.9
CAN+straight flow	91.0	72.6	88.2
CAN+FlowUDA	91.2	72.9	88.5

Comparison with mixup and straight flow. We compare three feature interpolation strategies under the same experimental protocol: (1) *mixup*, which generate m samples by sampling m different $\lambda \sim \text{Beta}(\alpha, \alpha)$; (2) *straight flow*, which uses linear interpolation $\boldsymbol{z}_{\frac{l}{m}} = (1 - \frac{l}{m}) * \boldsymbol{f}^s + \frac{l}{m} * \boldsymbol{f}^t, \quad l \in \{1, 2, \ldots, m\}$ to produce m intermediate points; and (3) *FlowUDA*, which learns an ODE flow via a diffusion network s_θ and samples m points along it. As shown in Table V, straight flow consistently outperforms mixup, and FlowUDA further improves over straight flow across all datasets and base methods. This demonstrates two key advantages of our approach: (1) Structured trajectory modeling (even with fixed linear paths) is more effective than random pairwise mixup; (2) Learning adaptive flows via s_θ captures richer domain-shift dynamics, leading to better generalization.

TABLE VI
THE EFFECTS OF SAMPLING ON OFFICE-31. RS: RANDOM SAMPLING, RSC: RANDOM SAMPLING BY CATEGORY USING PSEUDO-LABEL

Layer number	A→D	A→W	D→A	W→A
Resnet	68.9	68.4	62.5	60.7
FlowUDA (RS)	79.5	82.4	67.6	68.1
FlowUDA (RSC)	83.7	88.4	71.0	70.9

Discussion about sampling. In FlowUDA, we use pseudo-labels to construct class-wise straight flows to train the s_θ. In this experiment, we also compared the training of the network s_θ without using pseudo labels where we randomly sampled source domain and target domain samples in all categories. The results are shown in the Table VI. What can be found is that compared with the baseline, the effect can be improved even without using pseudo labels. This shows that through random sampling, the trained network s_θ can simulate the diffusion process from the source distribution to the target distribution. At the same time, the best experimental results can be obtained by using category-by-category training. This is because by using pseudo labels, the training difficulty of the network s_θ is greatly reduced, so better results are obtained. Although pseudo-labels may contain noise, experimental results show they provide training positive gains overall.

Discussion on consistency constraint. As mentioned in main text, $\mathcal{L}^c(\cdot, \cdot)$ is a consistency constraint between the output of s_θ and the gradient of the flow and L_2-Norm is selected. To evaluate the effect of this loss function selection, we further test L_1-Norm and cross entropy on Office-31 and Office-home by using the method CAN+FlowUDA. As shown in Table VII, the performance of FlowUDA is marginally affected by the loss function selection, suggesting the stability and flexibility of our model.

TABLE VII
THE EFFECTS OF USING DIFFERENT CONSISTENCY LOSS FUNCTIONS ON *Office-31* AND *Office-home*. CE: CROSS ENTROPY; L_1: L_1-NORM; L_2: L_2-NORM.

Loss	A→D	A→W	D→A	D→W	W→A	W→D	Office-home
CE	95.7	95.9	78.6	99.4	77.5	100.0	72.7
L_1	95.4	95.3	78.4	99.3	77.2	100.0	72.4
L_2	96.0	96.0	78.6	99.4	77.4	100.0	72.9

TABLE VIII
THE PERFORMANCE OF CLASSIFICATION ACCURACY (%) ON SAMPLING NUMBER m BASED ON *Office-Home* DATASET.

Method	1	2	3	4	5	6
PSE+FlowUDA	65.9	66.3	66.8	67.2	67.4	67.5
CAN+FlowUDA	72.2	72.4	72.5	72.6	72.8	72.8

Discussion about the sampling number m. In FlowUDA, for each source feature, the m features are sampled along the diffusion ODE flow. Therefore, the effect on the sampling number is performed on the Office-Home dataset. The results are shown in Table VIII. It can be observed that the performance can gradually become better as the sampling number increases, which is consistent with the common sense that when there are more samples, the model should be learned better. However, considering the computational cost, we finally set m as 5,5 and 3 in Office-31, Office-Home and Visda-17 respectively.

TABLE IX
THE EFFECTS WITH DIFFERENT LAYER NUMBER ON OFFICE-31 (FIRST TWO ROWS) AND OFFICE-HOME (LAST TWO ROWS).

Layer number	1	2	3	4	5
PSE+FlowUDA	84.4	86.2	88.0	88.9	89.5
CAN+FlowUDA	88.9	89.9	90.3	90.7	91.2
PSE+FlowUDA	63.2	64.8	65.8	66.6	67.4
CAN+FlowUDA	69.9	71.0	71.8	72.6	72.9

The size of the network s_θ. s_θ is a multi-layer neural networks used to approximate the gradient of ODE flow. Each layer is composed of a linear layer, batch norm and activation function. The input to the first layer has the dimension of $d + 2$, where d is the feature dimension, and the remaining dimensions are used for category and time respectively. Here $d = 2048$ due to the Resnet backbone. For simplicity, the remaining layers of the s_θ network are all d-dimensional input and d-dimensional output. In this experiment, the effect of layer number is discussed, which is shown in Table IX. It can be found that as the layer number increases, the better the performance is. The reason is that the network also becomes stronger to fit the approximation function, resulting in better results. Of course, before the learning ability is saturated, more blocks will definitely have better learning

ability, but considering the computational overhead, we finally choose 5 layers. In addition, when the layer number increases from 1 to 2, the performances of both CAN+FlowUDA and PSE+FlowUDA are also greatly improved. This is because when layer number is 1, the network can only simulate linear function, while when it is 2, it can start learning non-linear function.

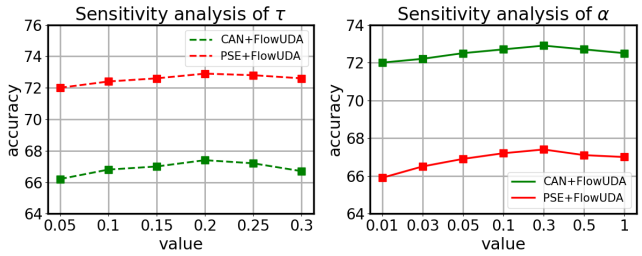


Fig. 6. Sensitivity analysis of τ and α on Office-Home.

Parameter analysis of τ and α. There exist two hyperparameters τ and α in FlowUDA. τ is a hyperparameter used to help filter a part of noisy target samples. The impact on performance is discussed on Office-Home dataset. The results are shown in Fig. 6. It can be found that as τ increases, the performance will first increase and then decrease. This is because when τ is small, a small number of target samples are selected to train s_θ, which results in the learned s_θ not being able to handle the distribution transition between two domains very well. When τ is too large, too many noise samples are selected to train s_θ, which causes noise generated samples. But overall, it can be observed from Fig. 6, there exists a performance stable region which is robust to the value of τ.

The hyperparameter α is used for the trade-off between $Loss_{base}$ and $Loss_{flow}$. A sensitivity analysis is performed on α to analyze the performance robustness of FlowUDA. We conduct experiments with PSE+FlowUDA and CAN+FlowUDA on Office-Home dataset. The average performances are shown in Fig. 6. It can be found that as α becomes larger, the average performance of both methods first increases and then decreases. There exists a range of 0.1-0.5, in which the performance is stable. It shows that the hyperparameter α is not sensitive.

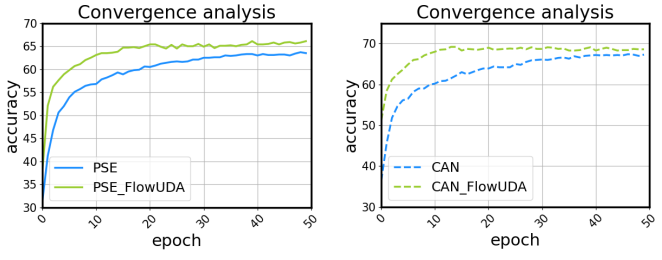


Fig. 7. Convergence curves on C→A task in Office-home dataset.

Convergence analysis. How fast the model converges is an issue we need to consider, this experiment discusses the convergence performances between PSE and PSE+FlowUDA and between CAN and CAN+FlowUDA. The experiments are performed on the C→A task in Office-home dataset, which is shown in Fig.7. What can be found is that when FlowUDA is used, the original method not only performs better in performance, but also converges faster.

TABLE X
ABLATION STUDY ON PSEUDO-LABEL ASSIGNMENT STRATEGY. "TARGET-ONLY" USES ONLY THE UPDATED TARGET CLUSTER CENTERS; "SOURCE+TARGET" USES BOTH SOURCE AND TARGET CENTERS.

Method	Office-31	Office-Home	VisDA-17
Source+Target	90.8	72.2	87.5
Target-only (Ours)	**91.2**	**72.9**	**88.5**

Ablation study on pseudo-label assignment strategy. To validate our design choice in pseudo-label assignment, we conduct an ablation study comparing two strategies for computing sample-to-center similarity: (1) using only the updated target cluster centers (our method), and (2) using a combination of source and target cluster centers (e.g., averaging the two centers before computing distance). As shown in Table X, using only the adaptive target centers consistently yields better performance across all three datasets. This is because the updated target centers better reflect the target feature distribution, while incorporating the original source centers reintroduces domain-shift bias that has already been mitigated through K-means iteration. These results confirm that our strategy of relying solely on the latest target cluster centers is both principled and effective.

TABLE XI
COMPUTATIONAL COST COMPARISON ON OFFICE-31 (A→D).

Method	GPU Memory (MB)	Time (s)
CAN	5,722 / 2104	42 / 1.2
CAN+FlowUDA	6,310 / 2,104	53 /1.2

Computational Cost Analysis. We evaluate the computational overhead of FlowUDA on the Office-31 A→D task. As shown in Table XI, CAN+FlowUDA introduces moderate additional cost during training (6,310 MB GPU memory and 53 s per epoch) compared to CAN (5,722 MB and 42 s), due to the auxiliary diffusion network s_θ and flow sampling. Importantly, during inference, the diffusion network s_θ is discarded, and only the adapted feature extractor F and classifier C are used. Therefore, FlowUDA incurs zero additional inference cost compared to the base method. This is reflected in the identical inference memory (2,104 MB) and latency (1.2 s) between CAN and CAN+FlowUDA.

VII. CONCLUSION

In this work, we proposed a novel method for unsupervised domain adaptation, named as FlowUDA. Inspired by the idea of diffusion models, we trained a network which constructs a flow from the source distribution to the target distribution based on an ODE equation. Then the source model is retrained to be able to classify this feature flow, which is different from the previous UDA method of feature classification. Our approach is simple yet effective, and most importantly, it can be easily integrated into the existing UDA methods as a plug-in to further improve performance. Thorough performance

comparison and experimental analysis on multiple datasets verified the effectiveness and robustness of FlowUDA.

REFERENCES

[1] 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.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[3] S. T. Krishna and H. K. Kalluri, "Deep learning and transfer learning approaches for image classification," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 7, no. 5S4, pp. 427–432, 2019.

[4] B. Zhao, J. Feng, X. Wu, and S. Yan, "A survey on deep learning-based fine-grained object classification and semantic segmentation," *International Journal of Automation and Computing*, vol. 14, no. 2, pp. 119–135, 2017.

[5] 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.

[6] Y. Zuo, H. Yao, L. Zhuang, and C. Xu, "Margin-based adversarial joint alignment domain adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 4, pp. 2057–2067, 2022.

[7] L. Zhang, P. Wang, W. Wei, H. Lu, C. Shen, A. van den Hengel, and Y. Zhang, "Unsupervised domain adaptation using robust class-wise matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 5, pp. 1339–1349, 2019.

[8] X. Xu, H. He, H. Zhang, Y. Xu, and S. He, "Unsupervised domain adaptation via importance sampling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 12, pp. 4688–4699, 2020.

[9] Y. Liu, Z. Zhou, and B. Sun, "Cot: Unsupervised domain adaptation with clustering and optimal transport," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 19 998–20 007.

[10] M. Meng, Z. Wu, T. Liang, J. Yu, and J. Wu, "Exploring fine-grained cluster structure knowledge for unsupervised domain adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 8, pp. 5481–5494, 2022.

[11] L. Zhou, S. Xiao, M. Ye, X. Zhu, and S. Li, "Adaptive mutual learning for unsupervised domain adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 11, pp. 6622–6634, 2023.

[12] 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.

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

[14] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[15] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *NeurIPS*, 2018.

[16] Q. Tian, Y. Zhu, H. Sun, S. Chen, and H. Yin, "Unsupervised domain adaptation through dynamically aligning both the feature and label spaces," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[17] G. French, M. Mackiewicz, and M. Fisher, "Self-ensembling for visual domain adaptation," in *International Conference on Learning Representations*, 2018.

[18] J. Liang, D. Hu, and J. Feng, "Domain adaptation with auxiliary target domain-oriented classifier," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 632–16 642.

[19] Y. Sun, E. Tzeng, T. Darrell, and A. A. Efros, "Unsupervised domain adaptation through self-supervision," *arXiv preprint arXiv:1909.11825*, 2019.

[20] B. Sun, J. Feng, and K. Saenko, "Correlation alignment for unsupervised domain adaptation," in *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 153–171.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014.

[22] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7167–7176.

[23] B. B. Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty, "Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 447–463.

[24] R. Xu, P. Liu, L. Wang, C. Chen, and J. Wang, "Reliable weighted optimal transport for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4394–4403.

[25] M. Li, Y.-M. Zhai, Y.-W. Luo, P.-F. Ge, and C.-X. Ren, "Enhanced transport distance for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 936–13 944.

[26] Y. Xu, G. Wen, Y. Hu, and P. Yang, "Modeling hierarchical structural distance for unsupervised domain adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, 2024.

[27] 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.

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

[29] L. Zhou, M. Ye, X. Zhu, S. Li, and Y. Liu, "Class discriminative adversarial learning for unsupervised domain adaptation," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 4318–4326.

[30] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 469–477.

[31] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2107–2116.

[32] 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.

[33] L. Zhou, M. Ye, X. Zhu, S. Xiao, X.-Q. Fan, and F. Neri, "Homeomorphism alignment for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 18 699–18 710.

[34] Z. Xiao, H. Wang, Y. Jin, L. Feng, G. Chen, F. Huang, and J. Zhao, "Spa: A graph spectral alignment perspective for domain adaptation," in *NeurIPS*, 2023.

[35] Z. Yue, Q. Sun, and H. Zhang, "Make the u in uda matter: Invariant consistency learning for unsupervised domain adaptation," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[36] L. Zhou, M. Ye, D. Zhang, C. Zhu, and L. Ji, "Prototype-based multisource domain adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 5308–5320, 2021.

[37] J. Na, H. Jung, H. J. Chang, and W. Hwang, "Fixbi: Bridging domain spaces for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 1094–1103.

[38] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 2066–2073.

[39] R. Gong, W. Li, Y. Chen, and L. V. Gool, "Dlow: Domain flow for adaptation and generalization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2477–2486.

[40] Z. Zhuang, Y. Zhang, and Y. Wei, "Gradual domain adaptation via gradient flow," in *The Twelfth International Conference on Learning Representations*, 2024.

[41] D. Osowiechi, G. A. V. Hakim, M. Noori, M. Cheraghalikhani, I. Ben Ayed, and C. Desrosiers, "Tttflow: Unsupervised test-time training with normalizing flow," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 2126–2134.

[42] R. Caseiro, J. F. Henriques, P. Martins, and J. Batista, "Beyond the shortest path: Unsupervised domain adaptation by sampling subspaces along the spline flow," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3846–3854.

[43] D. Zhu, Y. Li, Y. Shao, J. Hao, F. Wu, K. Kuang, J. Xiao, and C. Wu, "Generalized universal domain adaptation with generative flow networks," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 8304–8315.

[44] Y. Zhou, Y. Guo, S. Hao, R. Hong, and J. Luo, "Few-shot partial multi-view learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 10, pp. 11 824–11 841, 2023.

[45] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 6840–6851.

[46] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

[47] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2020.

[48] X. Liu, C. Gong *et al.*, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *The Eleventh International Conference on Learning Representations*, 2022.

[49] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," in *International Conference on Learning Representations*, 2021.

[50] J. Choi, S. Kim, Y. Jeong, Y. Gwon, and S. Yoon, "Ilvr: Conditioning method for denoising diffusion probabilistic models," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2021, pp. 14 347–14 356.

[51] T. Gu, G. Chen, J. Li, C. Lin, Y. Rao, J. Zhou, and J. Lu, "Stochastic trajectory prediction via motion indeterminacy diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 113–17 122.

[52] Y. Zhang, S. Chen, Y. Zhang, and J. Lu, "Diffusion-based target sampler for unsupervised domain adaptation," *arXiv preprint arXiv:2303.12724*, 2023.

[53] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8162–8171.

[54] Y. Benigmim, S. Roy, S. Essid, V. Kalogeiton, and S. Lathuilière, "One-shot unsupervised domain adaptation with personalized diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 698–708.

[55] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[56] D. Peng, Q. Ke, Y. Lei, and J. Liu, "Unsupervised domain adaptation via domain-adaptive diffusion," *arXiv preprint arXiv:2308.13893*, 2023.

[57] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 3, pp. 433–439, 1999.

[58] 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.

[59] M. Chen, S. Zhao, H. Liu, and D. Cai, "Adversarial-learned loss for domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 3521–3528.

[60] J. Huang, D. Guan, A. Xiao, S. Lu, and L. Shao, "Category contrast for unsupervised domain adaptation in visual tasks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1203–1214.

[61] J. Zhang, J. Huang, Z. Tian, and S. Lu, "Spectral unsupervised domain adaptation for visual recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9829–9840.

[62] J. Huang, N. Xiao, and L. Zhang, "Balancing transferability and discriminability for unsupervised domain adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5807–5814, 2022.

[63] M. Wang, S. Wang, X. Yang, J. Yuan, and W. Zhang, "Equity in unsupervised domain adaptation by nuclear norm maximization," *IEEE Transactions on Circuits and Systems for Video Technology*, 2024.

[64] T. Westfechtel, H.-W. Yeh, D. Zhang, and T. Harada, "Gradual source domain expansion for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2024, pp. 1946–1955.

[65] T. Xu, W. Chen, F. Wang, H. Li, R. Jin *et al.*, "Cdtrans: Cross-domain transformer for unsupervised domain adaptation," in *International Conference on Learning Representations*, 2022.

[66] J. Zhu, H. Bai, and L. Wang, "Patch-mix transformer for unsupervised domain adaptation: A game perspective," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 3561–3571.

[67] X. Jiang, Q. Lao, S. Matwin, and M. Havaei, "Implicit class-conditioned domain alignment for unsupervised domain adaptation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4816–4827.

[68] G. Wei, C. Lan, W. Zeng, and Z. Chen, "Metaalign: Coordinating domain alignment and classification for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 643–16 653.

[69] Y. Zhang, Z. Wang, J. Li, J. Zhuang, and Z. Lin, "Towards effective instance discrimination contrastive loss for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 388–11 399.

[70] 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.

[71] N. Xiao and L. Zhang, "Dynamic weighted learning for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 242–15 251.

[72] 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.

[73] 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.

[74] 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.

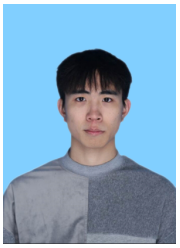
[75] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1406–1415.

[76] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[77] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

Lihua Zhou received the Ph.D. degree from University of Electronic Science and Technology of China in 2024. Currently, he is pursuing postdoctoral research at the Centre for Artificial Intelligence and Robotics, Chinese Academy of Sciences. His current research interests include machine learning, computer vision, and transfer learning.

Mao Ye received the B.S. degree from Sichuan Normal University, Chengdu, China, in 1995, and the M.S degree from University of Electronic Science and Technology of China, Chengdu, China, in 1998 and Ph.D. degree from Chinese University of Hong Kong, China, in 2002, all in mathematics. He has been a short-time visiting scholar at University of Queensland, and University of Pennsylvania. He is currently a professor and director of CVLab with University of Electronic Science and Technology of China, Chengdu, China. His research interests include machine learning and computer vision.



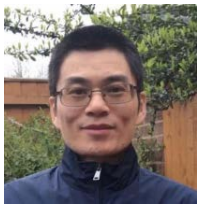
Nianxin Li received the M.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2024. He is currently pursuing the Ph.D. degree with the University of Electronic Science and Technology of China, Chengdu, China. His current research interests include machine learning, computer vision, and transfer learning.

Song Tang received the M.S. degree in system engineering from the Chongqing University of Posts and Communications, Chongqing, China, in 2013, and the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2017. From 2017 to 2019, he was an Assistant Researcher with the Department of Informatik, Universit at Hamburg, Hamburg, Germany. He is currently an Associate Professor with the Institute of Machine Intelligence, University of Shanghai for Science and Technology. His current research interests include transfer learning, reinforcement learning, and computer vision.

Xu-Qian Fan received the M.Phil. and Ph.D. degrees in Mathematics from the Chinese University of Hong Kong in 2000 and 2004 respectively. He is currently an associate professor of Department of Mathematics, Jinan University, Guangzhou, China. His research interests include applied probability and geometric analysis. He is interested in the problems that are motivated by machine learning, partial differential equations, quantum computation, Riemannian geometry, and stochastic differential equations.

Lei Deng received the Ph.D. degree from the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, in 2017. He was an assistant professor in School of Electrical Engineering & Intelligentization, Dongguan University of Technology. His research interests are timely network communications, intelligent transportation system, and spectral-energy efficiency in wireless networks.

Zhen Lei received the B.S. degree in automation from the University of Science and Technology of China in 2005, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences in 2010, where he is currently a Professor. He has published over 200 papers in international journals and conferences with 31000+ citations in Google Scholar and H-index 82. His research interests are computer vision, pattern recognition, image processing, and face recognition. He is the winner of the 2019 IAPR Young Biometrics Investigator Award. He was the Program Co-Chair of IJCB2023, was the Competition Co-Chair of IJCB2022, has served as the area chair for several conferences, and is an Associate Editor of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON BIOMETRICS, BEHAVIOR, AND IDENTITY SCIENCE, Pattern Recognition, Neurocomputing, and IET Computer Vision. He is a Fellow of IEEE, IAPR and AAIA.



Xiatian Zhu is a Senior Lecturer with Surrey Institute for People-Centred Artificial Intelligence, and Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford, UK. He received his Ph.D. degree from the Queen Mary University of London. He won the Sullivan Doctoral Thesis Prize 2016. He was a research scientist at Samsung AI Centre, Cambridge, UK. His research interests include computer vision, and machine learning.