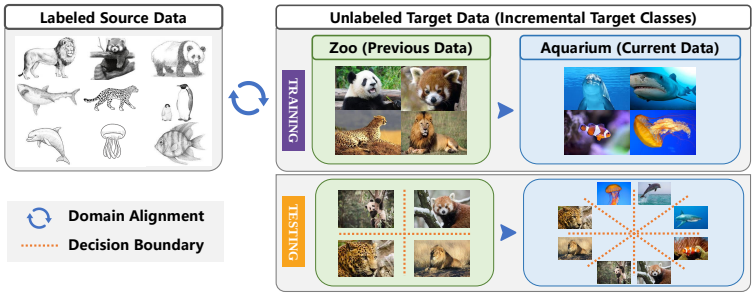

Fig. 1. An illustration of Class-incremental Unsupervised Domain Adaptation (CI-UDA), where labeled source data are accessible all the time, while unlabeled target data come online class-incrementally. When new target classes arrive, CI-UDA seeks to align the new target data to the source domain and retain the knowledge of previous target data since previous target data are unavailable.

shift between the source and target domains [35,42,55,58,60]. To deal with this, existing UDA methods conduct domain alignment either by domain-invariant feature learning [10,56] or by image transformation [14,41].

Most existing UDA methods assume the availability of all target data in advance. However, in practice, target data often come in a streaming manner with different categories [21,31,50]. For example, a practical scenario is to transfer the knowledge of sketch images for real-world animal recognition as shown in Fig. 1, where plenty of labeled sketches are easily collected in advanced but unlabeled real-world images come incrementally (*e.g.,* the images of the land animals in the zoo come first, followed by the sea animals in the aquarium). In such a scenario, it is more appropriate to adapt the source model with the target images observed so far (from partial animal classes) instead of waiting for the images of all animals to be available, which can be more memory-efficient and time-efficient. That is, the model needs to be first adapted with the target images from land animals, and then with the images from sea animals. Note that when adapting the source model with sea animals, the previous samples of land animals are unavailable for saving the data storage cost. In this scenario, existing UDA methods that assume all target classes to be available in advance tend to fail. To address this, we explore a new and practical task, called *Class-Incremental Unsupervised Domain Adaptation* (CI-UDA), where the labeled source samples are available all the time, but the unlabeled target samples come incrementally and only partial target classes are available at a time.

CI-UDA has two characteristics: 1) the target categories at the current time step are never seen before and only occupy a subspace of the source label space; 2) the target samples of previously seen categories will be unavailable for later adaptation. As a result, besides the common challenge of domain shifts in UDA [10,44], CI-UDA poses two new challenges. The first is how to detect the shared classes between source and target domains in each time step. Since only a portion of target data is available at each time step, the label space of

the target domain *is inconsistent with and is partial of* the source label space at each step, which makes domain alignment difficult. The second is how to alleviate catastrophic forgetting [47] of the old-class knowledge when learning new target classes. Since previous target samples are unavailable, later adaptation with new target classes results in knowledge forgetting of previous classes.

In CI-UDA, the key is to continually conduct domain adaptation in the absence of previous target samples. To deal with knowledge forgetting, a recent work [36] has shown that storing image prototypes for previous classes helps to retain knowledge. In addition, feature prototypes can also be used for domain alignment [32]. In other words, label prototypes open an opportunity for handling all challenges, simultaneously. However, a simple combination of existing methods [32,35,36] is not feasible for CI-UDA, since obtaining image prototypes for knowledge retaining [36] requires data labels but the target domain in CI-UDA is totally unlabeled. Moreover, feature prototypes [32,35] cannot update the feature extractor, so simply detecting them is unable to overcome the knowledge forgetting issue of the feature extractor in CI-UDA.

To better handle CI-UDA, we develop a new Prototype-guided Continual Adaptation (ProCA) method. To be specific, ProCA presents two solution strategies: 1) a label prototype identification strategy: we identify target label prototypes by detecting the shared class between source and target domains. Note that identifying label prototypes is challenging due to the inconsistent class space between the source and target domains. Therefore, detecting the shared classes is important, but is unfortunately difficult due to the absence of target labels. To overcome this, we dig into the difference between the shared classes and source private classes, and empirically observe (c.f. Fig. 3) that the cumulative probabilities of the shared classes are often higher than those of source private classes. Following this finding, we exploit the cumulative probabilities of target samples to detect the shared classes, and use the detected shared classes to identify target label prototypes. 2) a prototype-based alignment and replay strategy: based on the identified label prototypes, we conduct domain adaptation by aligning each target label prototype to the source center with the same class, and overcome catastrophic forgetting by enforcing the model to retain knowledge carried by the label prototypes learned from previous categories.

Extensive experiments on three benchmark datasets (*i.e.,* Office-31-CI, Office-Home-CI and ImageNet-Caltech-CI) demonstrate that ProCA is capable of handling CI-UDA. Moreover, we empirically show that ProCA can be used to improve existing partial UDA methods for tackling CI-UDA, which verifies the applicability of our method.

We summarize the main contributions of this paper as follows:

- We study a new yet difficult problem, called Class-incremental Unsupervised Domain Adaptation (CI-UDA), where unlabeled target samples come incrementally and only partial target classes are available at a time. Compared to vanilla UDA, CI-UDA does not assume all target data to be known in advance, and thus opens the opportunity for tackling more practical UDA scenarios in the wild.

– We propose a novel ProCA to handle CI-UDA. By innovatively identifying target label prototypes, ProCA is able to alleviate both domain discrepancies via prototype-based alignment and catastrophic forgetting via prototype-based knowledge replay. Moreover, ProCA can be applied to enhance existing partial domain adaptation methods to overcome CI-UDA.

2 Related Work

We first review the literature of unsupervised domain adaptation, including closed-set unsupervised domain adaptation, partial domain adaptation and continual domain adaptation. After that, we discuss a more relevant task, *i.e.,* class-incremental domain adaptation. Due to the page limit, we provide the literature of universal domain adaptation [53] and the difference between our ProCA and existing methods [1,2,3,32,35,36,47] in the supplementary (see Appendix A).

2.1 Unsupervised Domain Adaptation

Closed-set unsupervised domain adaptation (UDA). The goal of UDA [8,23,29,30,52,56] is to improve the model performance on the unlabeled target domain based on a label-rich relevant source domain. In this field, the most common task is closed-set UDA [33] which assumes that source and target domains share the same set of classes. Existing UDA methods have shown great progress in alleviating domain shifts by matching high-order moments of distributions [4,18,45], by learning domain-invariant features in an adversarial manner [10,15,40,56], or by image transformation via generative adversarial models [14,41,48]. Recently, OP-GAN [49] combines UDA with self-supervised learning, involving a self-supervised module to enforce the image content consistency.

Partial domain adaptation (PDA). Compared to closed-set UDA, PDA [1] assumes that the target label set is a subset of the source label set instead of restricting the same label set. In general, PDA aims to transfer a deep model trained from a big labeled source domain to a small unlabeled target domain. To handle the inconsistent label space, most existing methods assign class-level [1] or instance-level [2] transferability weights for source samples. To reduce negative transfer caused by source private classes, BA^{3}US [27] augments the target domain to conduct balanced adversarial alignment, while DPDAN [16] aligns the positive part of the source domain to the target domain by decomposing the source domain distribution into two parts.

Continual domain adaptation (CDA). Different from the above tasks, CDA [43] assumes that more than one unlabeled target domains come sequentially, and seeks to incrementally adapt the model to each new incoming domain without forgetting knowledge on previous domains. To this end, Dlow [11] bridges source and multiple target domains by generating a continuous flow of intermediate states, while VDFR [22] proposes to replay variational domain-agnostic features to tackle the domain shift and task shift. Recently, GRCL [43] regularizes the gradient of losses to learn discriminative features and preserve the previous knowledge, respectively.

Overall, the above methods are inapplicable in CI-UDA due to two aspects. On the one hand, closed-set UDA and PDA methods rely on the assumption that all target data are available in advance. In other words, these methods take no consideration of retaining previous knowledge. On the other hand, CDA assumes that the label set of each target domain is the same as the source label set, ignoring domain-shared classes detection. As a result, they tend to fail in handling the challenging CI-UDA.

2.2 Class-incremental Domain Adaptation

Class-incremental Domain Adaptation is related to class-incremental learning (CIL) that learns a model continuously from a data stream, where the classes increase gradually and only new classes are available at each time. CIL requires the model to classify the samples of all classes observed so far. To overcome the issue of catastrophic forgetting, existing CIL methods retain the knowledge of previous classes either by storing or generating data from previous classes [3,36,47], or by preserving the relevant model weights of the previous classes [20,28,54].

Recently, researchers extend CIL to domain adaptation and study a new task, called class-incremental domain adaptation [21,50]. Specifically, this task seeks to alleviate the domain shift between domains and incrementally learn the private classes in the target domain. To this end, with *partial labeled target private samples*, CIDA [21] generates class-specific prototypes and learns a target-specific latent space to obtain centroids under the source-free domain adaptation scenario, and CBSC [50] utilizes supervised contrastive learning for novel class adaptation and domain-invariant feature extraction.

The above class-incremental domain adaptation is different from CI-UDA in two aspects. 1) Goal: class-incremental domain adaptation seeks to handle the issue of learning new target private classes incrementally, while CI-UDA seeks to handle the issue of domain adaptation with a class-incremental target domain that has no target private class. 2) Target Labels: class-incremental domain adaptation requires one-shot or few-shot labeled target samples as a prerequisite, while CI-UDA assumes a totally unlabeled target domain. Thus, directly applying existing methods to solve CI-UDA is unfeasible. In contrast, ProCA conducts unsupervised domain alignment and knowledge replay by identifying target label prototypes, thus providing the first feasible solution to CI-UDA.

3 Problem Definition

Notation. Let $\mathcal{D}_s=\{(\mathbf{x}_j^s, y_j^s)|\ y_j^s \in \mathcal{C}_s\}_{j=1}^{n_s}$ denotes the source domain, where n_s is the number of source data pairs $(\mathbf{x}^s, y^s)$ and $\mathcal{C}_s$ denotes the source label set with the class number $|\mathcal{C}_s| = K$. Moreover, we denote the unlabeled target domain as $\mathcal{D}_t=\{\mathbf{x}_i\}_{i=1}^{n_t}$ with n_t target samples. $\mathcal{C}_t$ denotes the target label set.

Class-incremental unsupervised domain adaptation. Unsupervised domain adaptation (UDA) aims to transfer knowledge from a label-rich source domain $\mathcal{D}_s$ to an unlabeled target domain $\mathcal{D}_t$. The key to resolving UDA is to

conduct domain alignment for mitigating domain shift. Existing UDA methods generally assume that all target samples are accessible in advance and have a fixed label space that is the same to the source domain (*i.e.,* $\mathcal{C}_t = \mathcal{C}_s$). However, in real-world applications, target samples often come in a streaming manner, and meanwhile, the number of target categories may increase sequentially. To address this, we seek to explore a more practical task, namely Class-Incremental Unsupervised Domain Adaptation (CI-UDA), where labeled source samples are available all the time, but unlabeled target samples come incrementally and only partial target classes are available at a time. Here, we reuse $\mathcal{D}_t$ to denote the unlabeled target domain at the current time. Note that the label set of the target data in each time step is a subset of that of the source domain, *i.e.,* $\mathcal{C}_t \subset \mathcal{C}_s$.

Besides the domain shift that all UDA methods resolve, CI-UDA poses two new challenges: 1) how to identify the shared classes between two domains in each time step; 2) how to alleviate knowledge forgetting of old classes when learning new target classes. Due to the integration of these challenges, existing UDA methods [10,14,23,34,44,52] are incapable of handling CI-UDA. Therefore, how to handle this practical yet difficult task remains an open question.

4 Prototype-guided Continual Adaptation

Previous studies have shown that label prototypes are effective in independently handling either UDA [15,32,35] or class-incremental learning [3,36,47] tasks. Although these methods cannot be directly used to handle CI-UDA, they inspire us to explore a unified prototype-based method to handle all challenges in CI-UDA, simultaneously. This idea, however, is non-trivial to explore in practice. Since the source and target domains have different label spaces at different time steps, it is difficult to identify target label prototypes. To address these challenges, we propose a novel Prototype-guided Continual Adaptation (ProCA) method.

Method overview. We summarize the overall training scheme of ProCA in Fig. 2. ProCA consists of two solution strategies, that are, 1) label prototype identification and 2) prototype-based alignment and replay. We first briefly introduce the two strategies below.

First, we develop a label prototype identification strategy (c.f. Section 4.1) to identify target label prototypes at each time step under inconsistent label spaces between source and target domains. To this end, we firstly propose a shared class detection method to distinguish the domain-shared classes from the source private classes. Based on the detected shared label set and the target pseudo labels generated by clustering, we identify target label prototypes for each shared class and construct an adaptive memory bank $\mathcal{P}$ to record them.

Second, based on the identified label prototypes, we propose a prototype-based alignment and replay strategy (c.f. Section 4.2) to align each image prototype to the corresponding source center and enforce the model to retain knowledge learned on previous classes. Specifically, we conduct prototype-based alignment by training the feature extractor G to learn domain-invariant features through a supervised contrastive loss $\mathcal{L}_{con}$. Meanwhile, we impose a knowledge

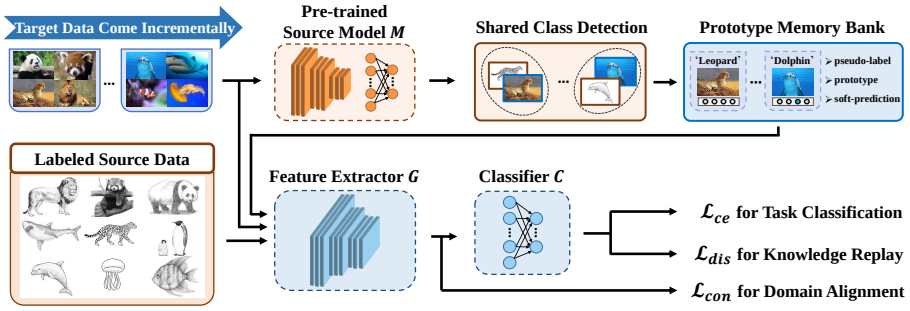

Fig. 2. An overview of Prototype-guided Continual Adaptation, ProCA consists of two strategies: 1) Label prototype identification: by detecting the shared classes between source and target domains at each time step, we identify target label prototypes for each class and record them in a memory bank. 2) Prototype-based alignment and replay: we align each target label prototype to the corresponding source center for training a domain-invariant feature extractor via $\mathcal{L}_{con}$; meanwhile, we use the saved label prototypes to enforce the model to retain knowledge learned from previous classes via $\mathcal{L}_{dis}$. Moreover, we use the cross-entropy $\mathcal{L}_{ce}$ for task classification based on the labeled source samples and pseudo-labeled target samples.

distillation loss $\mathcal{L}_{dis}$ for prototype-based knowledge replay. Moreover, based on the pseudo-labeled target data and labeled source data, we train the whole model $\{G, C\}$ via the standard cross-entropy loss $\mathcal{L}_{ce}$.

Overall, the training objective of ProCA is as follows:

$$\min_{\{\theta_g,\theta_c\}} \mathcal{L}_{ce}(\theta_g,\theta_c) + \lambda\mathcal{L}_{con}(\theta_g) + \eta\mathcal{L}_{dis}(\theta_g,\theta_c), \tag{1}$$

where θ_g and θ_c denote the parameters of the feature extractor G and the classifier C, respectively. Moreover, λ and η are trade-off parameters.

4.1 Label Prototype Identification

The key step in our proposed ProCA is to identify target label prototypes, which is non-trivial in the setting of CI-UDA. To this end, we propose a label prototype identification strategy that consists of four components: 1) shared class detection; 2) pseudo label generation for target data; 3) prototype memory bank construction and 4) prototype memory bank updating.

Shared class detection. When new unlabeled target samples arrive, it is difficult to detect the shared classes between the source and target domains since the target samples are unlabeled. To resolve this, we dig into the difference of the pre-trained source model in predicting the shared classes and the source private classes. As shown in Fig. 3, we find that the cumulative prediction probabilities of the target samples regarding the shared classes are higher than those regarding source private classes. Following this, we propose to detect the shared classes based on the cumulative probabilities of target samples. Specifically, as shown

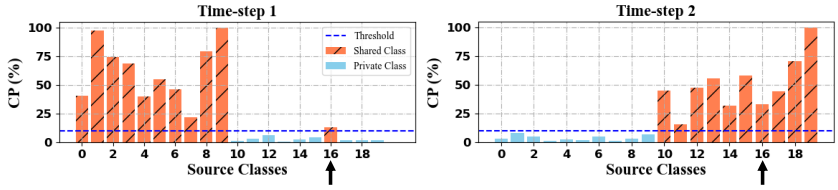

Fig. 3. The cumulative probability (CP) of target samples regarding source classes on Art→Real World, Office-Home-CI. For each time step, 10 target classes are available, *i.e.,* Class 0 to 9 in time step 1 and Class 10 to 19 in time step 2. The results show that the CP values of the shared classes are often higher than those of source private classes. Moreover, during training, since the CP of the 16-th class in time step 2 is higher than that in time step 1, we update the corresponding target class prototypes.

in Fig. 4, we exploit the source pre-trained model M to infer all target samples in each time step and obtain the cumulative probability of each class k by:

$$u_k = \sum_{i=1}^{n_t} C_k(G(\mathbf{x}_i)), \tag{2}$$

where $C_k(\cdot)$ denotes the k-th element in the softmax output prediction and n_t denotes the number of target samples at the current time. To enhance the generalization, we normalize the cumulative probability u_k to $[0, 1]$ by $u_k{=}\frac{u_k - min(\mathbf{u})}{max(\mathbf{u}) - min(\mathbf{u})}$, where $\mathbf{u}{=}[u_0, u_1, ..., u_K]$ is the probability vector in terms of all K classes.

Based on the cumulative probability u_k and a pre-defined threshold α, the judgement of the class k is made by: if $u_k \geq \alpha$, class k is a shared class; otherwise, class k is a source private class.

Pseudo label generation for target data. Based on the identified shared classes, we next generate pseudo labels for unlabeled target samples with a self-supervised pseudo-labeling strategy [26]. To be specific, let $\mathbf{q}_i{=}G(\mathbf{x}_i)$ be the extracted feature w.r.t. $\mathbf{x}_i$ and let $\hat{y}_i^k{=}C_k(\mathbf{q}_i)$ be the predicted probability of the classifier regarding class k, we first attain the initial centroid for each class k in the shared label set by:

$$\mathbf{c}_k = \frac{\sum_{i=1}^{n_t} \hat{y}_i^k \mathbf{q}_i}{\sum_{i=1}^{n_t} \hat{y}_i^k}. \tag{3}$$

Such an initialization is able to characterize well the distribution of different categories [26]. Based on these centroids, the pseudo label of the i-th target data is obtained via a nearest centroid approach:

$$\bar{y}_i = \arg\max_k \phi(\mathbf{q}_i, \mathbf{c}_k), \tag{4}$$

where $\phi(\cdot, \cdot)$ denotes the cosine similarity, and the pseudo label $\bar{y}_i \in \mathbb{R}^1$ is a scalar. During pseudo label generation, we update the centroid of each class by $\mathbf{c}_k = \frac{\sum_{i=1}^{n_t} \mathbb{I}(\bar{y}_i{=}k)\mathbf{q}_i}{\sum_{i=1}^{n_t} \mathbb{I}(\bar{y}_i{=}k)}$ and then update pseudo labels based on Eqn. (4) one more

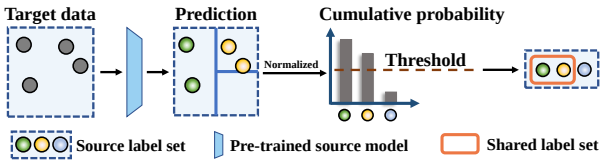

Fig. 4. The process of shared class identification. We first compute the cumulative probabilities by summing the output predictions of the pre-trained source model regarding all target samples. Then, we rescale the cumulative probabilities to $[0, 1]$ by min-max normalization. Based on the normalized probabilities, we judge the shared classes through thresholding.

time, where $\mathbb{I}(\cdot)$ is the indicator function. Note that we only compute the class centroids for the shared classes.

Prototype memory bank construction. Based on the detected shared label set and the generated target pseudo labels, we then identify target label prototypes for each shared class. Specifically, we maintain a memory bank $\mathcal{P} = \{(\mathbf{p}_i, \mathbf{h}_i, \bar{y}_i)\}_{i=1}^{N}$ to record prototypes for all detected shared classes, where $\mathbf{p}_i$, $\mathbf{h}_i$, $\bar{y}_i$ and N denote the image prototype, the predicted soft label, the predicted hard pseudo label and the number of prototypes, respectively. Moreover, we denote all seen target label set as $\mathcal{C}_{at}$ and save M image prototypes for each class in the memory bank, *i.e.,* $N = |\mathcal{C}_{at}|M$. During the training process, when a new pseudo-labeled target class comes, we expand the memory bank by adding the corresponding target prototypes. Formally, for the k-th class, we denote the pseudo-labeled target domain as $\mathcal{D}_t^k = \{\mathbf{x}_i^k\}_{i=1}^{n_k}$ and attain its feature center by $\mathbf{f}_t^k = \frac{1}{n_k}\sum_{i=1}^{n_k} G(\mathbf{x}_i^k)$. Inspired by iCaRL [36], we select the image prototype for the k-th class via a nearest neighbor approach based on the target feature center:

$$\mathbf{p}_m^k = \underset{\mathbf{x}^k \in \mathcal{D}_t^k}{\arg\min} \left\| \mathbf{f}_t^k - \frac{1}{m}[G(\mathbf{x}^k) + \sum_{i=1}^{m-1} G(\mathbf{p}_i^k)] \right\|_2, \tag{5}$$

where m is the iterative index range from 1 to M. Note that we iterate Eqn. (5) for M times to obtain M prototypes.

Prototype memory bank updating. During the shared class detection process, false shared classes may exist, disturbing pseudo label generation for target data. In this sense, the image prototypes of these classes need to be updated. To this end, we devise an updating strategy based on the cumulative probability u_k. Specifically, for a target class k existing in the memory bank, we update its prototypes when a higher u_k occurs. For example, in Fig. 3, when the cumulative probability of the 16-th class in the time step 2 is higher than that in the time step 1, the image prototypes of the 16-th class would be updated via Eqn. (5).

Algorithm 1 Training paradigm of ProCA

Require: Unlabeled target data $\mathcal{D}_t$=$\{\mathbf{x}_i\}_{i=1}^{n_t}$ at the current time; Pre-trained source model $\{G, C\}$; Prototype memory bank $\mathcal{P}$; Training epoch E; Parameters η, λ, α, T.

```
Detect shared classes based on Eqn. (2);
Obtain target pseudo labels based on Eqn. (4);
for e = 1 → E do
    Update target label prototypes based on Eqn. (5);
    Extract target data features G(x) based on G;
    Obtain target predictions C(G(x)) based on C;
    Compute L_con, L_dis based on Eqns. (7) and (8);
    Update G and C by optimizing Eqn. (1)
end for
return G and C.
```

4.2 Prototype-based Alignment and Replay

Based on target label prototypes, we develop a new prototype-based alignment and replay strategy to handle the issues of domain shifts and catastrophic forgetting below.

Prototype-based domain alignment. Based on the target label prototypes, we are able to conduct class-wise alignment to explicitly mitigate domain shifts. To this end, we propose to align each pseudo-labeled prototype to the source center of the corresponding class. To be specific, for the k-th class, we first attain source feature center $\mathbf{f}_s^k$ by:

$$\mathbf{f}_s^k = \frac{\sum_{j=1}^{n_s} \mathbb{I}(y_j^s = k) G(\mathbf{x}_j^s)}{\sum_{j=1}^{n_s} \mathbb{I}(y_j^s = k)}. \tag{6}$$

Then for any image prototype $\mathbf{p}_i$ as an anchor, we conduct prototype alignment via the contrastive loss [57,19]:

$$\mathcal{L}_{con} = -\log \frac{\exp(\mathbf{v}_i^\top \mathbf{f}_s^{\bar{y}_i}/\tau)}{\exp(\mathbf{v}_i^\top \mathbf{f}_s^{\bar{y}_i}/\tau) + \sum_{j=1, j\neq\bar{y}_i}^{K-1} \exp(\mathbf{v}_i^\top \mathbf{f}_s^{j}/\tau)}, \tag{7}$$

where $\mathbf{v}_i$=$G(\mathbf{p}_i)$ denotes the extracted feature of the image prototype $\mathbf{p}_i$ with $\bar{y}_i$ as its pseudo label and τ denotes a temperature factor. This loss enables the feature extractor G to learn domain-invariant features, which helps to alleviate domain discrepancies.

Prototype-based knowledge replay. Since target samples of previous classes are unavailable, the model suffers from catastrophic forgetting [47] during CI-UDA. To overcome this, based on the identified prototypes with soft labels, we adopt knowledge distillation [25] to enforce the model to retain the knowledge acquired from previous classes:

$$\mathcal{L}_{dis} = -\frac{1}{N} \sum_{i=1}^{N} \mathbf{h}_i^\top \log C(G(\mathbf{p}_i)), \tag{8}$$

where N denotes the number of prototypes.

At last, we summarize the pseudo-code of ProCA in Algorithm 1, while the pseudo-code of the prototype identification scheme is put in Appendix B.

5 Experiments

5.1 Experimental Setup

To verify the effectiveness of the proposed method, we conduct empirical studies based on the following experimental settings.

Datasets. We construct three dataset variants to simulate class-incremental scenarios, based on benchmark UDA datasets, *i.e.,* Office-31 [38], Office-Home [46], and ImageNet-Caltech [12,37]. 1) ***Office-31-CI*** consists of three distinct domains, *i.e.,* Amazon (A), Webcam (W) and DSLR (D). Three domains share 31 categories. We divide each domain into three disjoint subsets with each containing 10 categories in alphabetic order. 2) ***Office-Home-CI*** contains four distinct domains, *i.e.,* Artistic images (Ar), Clip Art (Cl), Product images (Pr) and Real-world images (Rw), each with 65 categories. For each domain, we build six disjoint subsets with 10 categories in random order. 3) ***ImageNet-Caltech-CI*** includes ImageNet-1K [37] and Caltech-256 [12]. Based on the shared 84 classes, we form two tasks: ImageNet (1000) $\rightarrow$ Caltech (84) and Caltech (256) $\rightarrow$ ImageNet (84). For target domains, we build eight disjoint subsets with each containing 10 categories. More details of data construction are in Appendix C.

Implementation details. We implement our method in PyTorch and report the mean $\pm$ stdev result over 3 different runs. ResNet-50 [13], pre-trained on ImageNet, is used as the network backbone. In ProCA, we train the model using the SGD optimizer with a learning rate of 0.001. In addition, the training epochs are set to 10 for Office-31-CI, 30 for Office-Home-CI, and 15 for ImageNet-Caltech-CI, respectively. For hyper-parameter, we set λ, η, α and M to 0.1, 1, 0.15 and 10. More training details of ProCA are in Appendix D.

Compared methods. We compare ProCA with four categories of baselines: (1) source-only: ResNet-50 [13]; (2) unsupervised domain adaptation: DANN [10]; (3) partial domain adaptation: PADA [1], ETN [2], BA^{3}US [27]; (4) class-increme ntal domain adaptation: CIDA [21].

Evaluation protocols. To fully evaluate the proposed method, we report three kinds of accuracy measures. 1) ***Final Accuracy***: the classification accuracy in the final time step of CI-UDA. 2) ***Step-level Accuracy***: the accuracy in each time step to evaluate the ability of sequentially learning. 3) ***Final S-1 Accuracy***: the average accuracy of step-1 classes in the final time step to evaluate the ability to handle catastrophic forgetting.

5.2 Comparisons with Previous Methods

We first compare our ProCA with previous methods in terms of Final Accuracy. The results are reported in Tables 1 and 2, which give the following observations. 1) ProCA outperforms all compared methods by a large margin in terms of the averaged Final Accuracy. To be specific, ProCA achieves the best or comparable performance on all transfer tasks (*e.g.,* Ar$\rightarrow$Cl on Office-Home-CI), which demonstrates the effectiveness of our method. 2) Compared with PDA methods, *i.e.,* PADA [1], ETN [2] and BA^{3}US [27], the superior performance of our

Table 1. Final Accuracy (%) on **Office-Home-CI**. DA and CI indicate domain adaptation and class-incremental learning.

Method	DA	CI	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-50	✗	✗	47.6	65.2	72.7	54.7	62.8	66.1	52.4	44.7	74.0	66.2	47.4	77.4	60.9
DANN [10]	✓	✗	33.1	40.0	45.8	36.8	36.6	44.1	32.0	29.8	49.8	42.4	40.2	55.2	40.5
PADA [1]	✓	✗	24.8	41.4	55.1	18.3	35.0	36.3	25.9	26.2	53.7	46.8	31.4	50.0	37.1
ETN [2]	✓	✗	42.4	2.8	7.4	4.3	60.3	6.3	50.7	33.8	70.8	3.7	43.5	75.1	33.4
BA^{3}US [27]	✓	✗	33.7	39.7	63.2	36.6	39.1	53.7	36.5	24.9	53.4	52.2	35.9	65.9	44.6
CIDA [21]	✓	✓	32.2	45.9	49.1	36.5	48.6	46.6	51.6	33.5	59.0	64.0	38.0	65.1	47.5
ProCA (ours)	✓	✓	**51.9**$_{\pm 0.4}$	**75.2**$_{\pm 0.2}$	**86.1**$_{\pm 0.3}$	**60.8**$_{\pm 0.1}$	**69.7**$_{\pm 0.1}$	**74.7**$_{\pm 0.7}$	**60.1**$_{\pm 0.2}$	**51.0**$_{\pm 0.2}$	**84.2**$_{\pm 0.4}$	**75.8**$_{\pm 0.2}$	**51.2**$_{\pm 0.5}$	**86.4**$_{\pm 0.1}$	**68.9**$_{\pm 0.1}$

Table 2. Final Accuracy (%) on **Office-31-CI** and **ImageNet-Caltech-CI**. DA and CI indicate adaptation and class-incremental learning.

Method	DA	CI	Office-31-CI							ImageNet-Caltech-CI		
			A→D	A→W	D→A	D→W	W→A	W→D	Avg.	C→I	I→C	Avg.
ResNet-50	✗	✗	74.1	74.4	58.5	96.9	61.2	99.6	77.5	72.3	70.7	71.5
DANN [10]	✓	✗	74.9	72.5	55.7	96.6	51.4	97.7	74.8	58.8	31.4	45.1
PADA [1]	✓	✗	56.9	61.5	12.5	82.4	46.7	84.3	57.4	37.3	45.9	41.6
ETN [2]	✓	✗	21.3	82.2	61.7	94.3	**64.1**	**100.0**	70.6	1.4	3.1	2.3
BA^{3}US [27]	✓	✗	74.1	73.3	63.3	94.8	64.0	**100.0**	78.3	60.8	45.0	52.9
CIDA [21]	✓	✓	70.4	64.5	48.1	95.1	52.7	98.8	71.6	69.3	49.2	59.2
ProCA (ours)	✓	✓	**81.8**$_{\pm 0.6}$	**82.5**$_{\pm 0.4}$	**65.2**$_{\pm 0.3}$	**99.1**$_{\pm 0.1}$	**64.1**$_{\pm 0.2}$	99.6$_{\pm 0.2}$	**82.1**$_{\pm 0.3}$	**82.9**$_{\pm 0.2}$	**83.1**$_{\pm 0.3}$	**83.0**$_{\pm 0.1}$

Table 3. Step-level Accuracy (%) on **Office-31-CI** and **Office-Home-CI**. DA and CI indicate adaptation and class-incremental learning.

Method	DA	CI	Office-31-CI				Office-Home-CI						
			Step 1	Step 2	Step 3	Avg.	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Avg.
ResNet-50 [13]	✗	✗	85.7	81.8	77.5	81.6	61.2	61.7	61.2	62.0	62.3	62.4	61.8
DANN [10]	✓	✗	82.4	79.6	74.8	78.9	42.7	40.5	41.1	41.1	39.8	40.5	40.9
PADA [1]	✓	✗	87.5	69.9	57.4	71.6	63.0	49.3	40.4	37.7	37.4	37.1	44.2
ETN [2]	✓	✗	**92.0**	82.7	70.6	81.8	62.7	62.0	59.2	58.7	49.0	33.4	54.2
BA^{3}US [27]	✓	✗	90.7	**85.9**	78.3	85.0	66.6	60.4	53.6	49.1	46.0	44.6	53.4
CIDA [21]	✓	✓	85.5	79.1	71.6	78.7	57.9	53.6	51.8	50.1	49.6	47.5	51.8
ProCA (ours)	✓	✓	91.3	**85.9**	**82.1**	**86.3**	**70.2**	**70.1**	**68.2**	**68.5**	**68.7**	**68.9**	**86.0**

method shows that retaining knowledge learned from previous categories is important for handling CI-UDA. 3) Since CIDA [21] also designs a regularization term to prevent catastrophic forgetting, it performs better than PDA methods in CI-UDA. However, CIDA ignores the source private classes in CI-UDA, which may result in negative transfer, and thus cannot handle CI-UDA very well. 4) Domain adaptation methods even perform worse than ResNet-50, which implies that only conducting alignment may make the model biased towards the target categories at the current step and forget the knowledge of previous categories.

We also report the Step-level Accuracy of all methods in Tables 3. If only considering the time step 1, CI-UDA degenerates to a standard PDA problem. In this case, previous PDA methods (*i.e.,* PADA [1], ETN [2] and BA^{3}US [27]) perform well. However, when learning new target samples at a new time step, these methods suffer from severe performance degradation while our ProCA maintains a relatively stable yet promising performance. To investigate the reason, we show the accuracy drop in percentage of these step-1 classes between the time step 1 and each time step. As shown in Fig. 5, when learning new target categories,

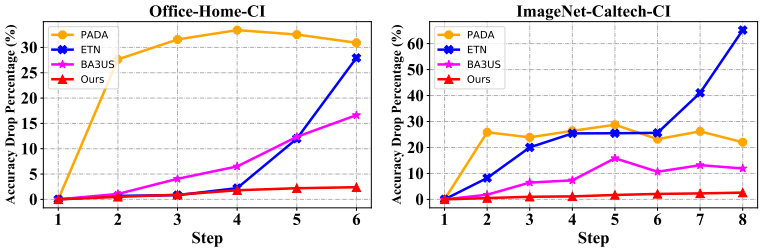

Fig. 5. Accuracy drop in percentage of different methods on **Office-Home-CI** and **ImageNet-Caltech-CI**. The accuracy drop means the accuracy difference of the step-1 classes between the time step 1 and the following each time step. The results show that our method has a significantly smaller accuracy drop in percentage than the compared methods, which shows that our method is skilled at alleviating catastrophic forgetting.

Table 4. Comparisons of the existing partial domain adaptation methods with and without our label prototype identification strategy on **Office-31-CI**. We show the final accuracy (%) and final S-1 accuracy (%) of two partial domain adaptation methods.

Method	Prototypes	Metric	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
PADA	✗	Final Acc. (%)	56.9	61.5	12.5	82.4	46.7	84.3	57.4
	✓		70.8	76.2	40.9	94.6	55.6	99.8	73.0
	✗	Final S-1 Acc. (%)	35.2	49.9	17.2	74.8	39.9	72.8	48.3
	✓		79.5	80.9	62.6	96.6	62.5	100.0	80.4
BA^{3}US	✗	Final Acc. (%)	74.1	73.3	63.3	94.8	64.0	100.0	78.3
	✓		75.4	77.4	64.2	100.0	65.5	100.0	80.4
	✗	Final S-1 Acc. (%)	89.7	89.0	76.7	100.0	77.3	99.8	88.7
	✓		92.4	90.9	78.6	100.0	77.3	100.0	89.8
ProCA (ours)	✓	Final Acc. (%)	81.6	82.6	65.5	99.1	63.9	99.8	82.1
	✓	Final S-1 Acc. (%)	96.7	94.2	74.1	100.0	80.0	100.0	90.8

the absence of target samples from previous categories causes state-of-the-art PDA methods to forget previous knowledge, leading to a severe accuracy drop of step-1 classes. In contrast, ProCA handles catastrophic forgetting effectively and shows a promising result in terms of Step-level Accuracy. Due to the page limitation, we put more detailed results of *each subtask* in the three datasets in terms of the Step-level Accuracy and the Final S-1 Accuracy in Appendix H.

5.3 Application to Enhancing Partial Domain Adaptation

In this section, we seek to determine whether ProCA can be used to enhance existing PDA methods, which cannot overcome catastrophic forgetting of previously seen categories in CI-UDA. To this end, we apply ProCA to improve classic PDA methods (*i.e.,* PADA [1] and BA^{3}US [27]) by integrating them with our label prototype identification strategy. As shown in Table 4, combining with ProCA significantly increases the performance of PDA methods, which demonstrates the applicability of our method to boost existing PDA methods for handling CI-UDA. Such an observation can also be supported by the results of applying ProCA to improve ETN [2], as shown in Appendix E.

Table 5. Ablation studies of the losses (*i.e.,* $\mathcal{L}_{ce}$, $\mathcal{L}_{dis}$ and $\mathcal{L}_{con}$) in ProCA. We show the Final Accuracy (%) and the Final S-1 Accuracy (%) on the 6 tasks of Office-31-CI.

Backbone	$\mathcal{L}_{ce}$	$\mathcal{L}_{dis}$	$\mathcal{L}_{con}$	Final Acc. (%)							Final S-1 Acc. (%)						
				A→D	A→W	D→A	D→W	W→A	W→D	Avg.	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
✓				74.1	74.4	58.5	96.9	61.2	99.6	77.5	87.8	85.3	68.5	100.0	71.4	100.0	85.5
✓	✓			76.8	78.8	59.6	99.0	62.4	99.6	79.4	90.0	90.8	68.6	100.0	71.5	100.0	86.8
✓	✓	✓		79.9	82.0	64.0	99.0	62.9	99.6	81.2	93.7	93.8	71.5	100.0	75.0	100.0	89.0
✓	✓		✓	78.7	81.5	63.2	99.0	63.5	99.0	80.8	91.3	92.4	69.7	100.0	76.7	98.0	88.0
✓	✓	✓	✓	81.6	82.6	65.5	99.1	63.9	99.8	**82.1**	96.7	94.2	74.1	100.0	80.0	100.0	**90.8**

5.4 Ablation Studies

To examine the effectiveness of the losses in ProCA, we show the quantitative results of the models optimized by different losses. As shown in Table 5, introducing $\mathcal{L}_{dis}$ or $\mathcal{L}_{con}$ enhances the model performance compared to optimizing the model with $\mathcal{L}_{ce}$ only. On the one hand, such a result verifies that prototype-based knowledge replay is able to alleviate catastrophic forgetting, resulting in promoting Final S-1 Accuracy. On the other hand, it also verifies that prototype-based domain alignment is able to mitigate domain shifts, resulting in promoting Final Accuracy. When combining the losses (*i.e.,* $\mathcal{L}_{ce}$, $\mathcal{L}_{dis}$, $\mathcal{L}_{con}$) together, we obtain the best performance.

In addition, we investigate the influences of hyper-parameters. The results in Appendix F show that ProCA is non-sensitive to λ and η, and the best performance of ProCA can be usually achieved by setting $\lambda = 0.1$ and $\eta = 1$. Moreover, we recommend setting $\alpha = 0.15$ since a high threshold helps to filter domain-shared classes out. Furthermore, we also investigate the influence of the number of prototypes and incremental classes in Appendix F, and evaluate the effectiveness of our shared class detection strategy in Appendix G.

6 Conclusions

In this paper, we have explored a practical transfer learning task, namely class-incremental unsupervised domain adaptation. To solve this challenging task, we have proposed a novel Prototype-guided Continual Adaptation (ProCA) method, which presents two solution strategies. 1) Label prototype identification: we identify target label prototypes with the help of a new shared class detection strategy. 2) Prototype-based alignment and replay: based on the identified label prototypes, we resolve the domain discrepancies and catastrophic forgetting via prototype-guided contrastive alignment and knowledge replay, respectively. Extensive experiments on three benchmark datasets , *i.e.,* Office-31-CI, Office-Home-CI and ImageNet-Caltech-CI, have demonstrated the effectiveness of ProCA in handling class-incremental unsupervised domain adaptation.

Acknowledgements. This work was partially supported by National Key R&D Program of China (No.2020AAA0106900), National Natural Science Foundation of China (NSFC) 62072190, Program for Guangdong Introducing Innovative and Enterpreneurial Teams 2017ZT07X183.

Supplementary Materials for "Prototype-Guided Continual Adaptation for Class-Incremental Unsupervised Domain Adaptation"

Hongbin Lin[1,3*], Yifan Zhang[2*], Zhen Qiu[1*],
Shuaicheng Niu[1], Chuang Gan[4], Yanxia Liu[1†], and Mingkui Tan[1,5†]

[1]South China University of Technology [2] National University of Singapore
[3] Information Technology R&D Innovation Center of Peking University
[4] MIT-IBM Watson AI Lab
[5] Key Laboratory of Big Data and Intelligent Robot, Ministry of Education
{sehongbinlin,seqiuzhen,sensc}@mail.scut.edu.cn,
yifan.zhang@u.nus.edu, ganchuang1990@gmail.com,
{cslyx,mingkuitan}@scut.edu.cn

In this supplementary, we provide more related work and discussions to clarify the differences of ProCA with existing methods. In addition, we also provide more implementation details and more experimental results. We organize the supplementary materials as follows.

1) In Appendix A, we review universal domain adaptation [53] and give more discussions on partial domain adaptation and prototype-based methods.
2) In Appendix B, we present the pseudo-code of our prototype identification scheme.
3) In Appendix C, we provide more construction details of class-incremental domain adaptation data.
4) In Appendix D, we provide more training details of our proposed ProCA.
5) In Appendix E, we provide more results of applying ProCA to improve the PDA method.
6) In Appendix F, we provide more ablation studies, including the influence of hyper-parameters, the number of target prototypes and the number of incremental classes.
7) In Appendix G, we examine the effectiveness of our shared class detection strategy.
8) In Appendix H, we provide more detailed results of *each subtask* in the three datasets in terms of the Step-level Accuracy and the Final S-1 Accuracy.

A More related work and discussions

In this appendix, we first review the literature of universal domain adaptation. After that, to better illustrate our novelty, we discuss the differences between

* Authors contributed equally.
† Corresponding authors.

CI-UDA and two types of relevant methods, *i.e.,* PDA methods and prototype-based methods.

Universal domain adaptation (Uni-DA). Uni-DA [53] assumes that the target label space is not limited to the source label space and may contain target private (unknown) classes. It seeks to classify unlabeled target samples into known classes from the source label space or an additional "unknown" category. With various transferability measures, most existing methods conduct domain alignment by quantifying sample-level transferability [9,53]. In addition, to exploit the structure information, DANCE [39] proposes to learn the structure of the target domain in a self-supervised way, while DCC [24] seeks to better exploit the intrinsic structure of the target domain and discover discriminative clusters. However, existing Uni-DA methods assume all target data are available in advance, making them incapable in CI-UDA.

Table I. The shared class indexes of different detection strategies at each time step on **Office-31-CI**. Note that correct shared classes are in blue while false shared classes are in magenta. Note that the higher SCD Acc. means the strategy detects more shared classes, and the higher TCD Acc. means the strategy detects less false shared classes.

Task	Method	Time Step	Shared Class Index	SCD Acc.	TCD Acc.	Avg.
A→D	HBW [16]	Step 1	[0, 1, 2, 9]	40.0	100.0	70.0
		Step 2	[0, 3, 4, 5, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 30]	100.0	40.0	70.0
		Step 3	[0, 5, 7, 9, 10, 12, 13, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]	100.0	47.6	73.8
	Ours	Step 1	[0, 1, 2, 3, 4, 5, 6, 7, 9, 12]	90.0	90.0	**90.0**
		Step 2	[8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]	100.0	90.9	**95.5**
		Step 3	[12, 13, 20, 21, 22, 23, 25, 26, 27, 28, 29]	90.0	81.8	**85.9**
A→W	HBW [16]	Step 1	[0, 1, 2, 6, 9]	50.0	100.0	75.0
		Step 2	[0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]	100.0	34.5	67.3
		Step 3	[0, 7, 9, 12, 13, 16, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]	100.0	55.6	77.8
	Ours	Step 1	[0, 1, 2, 3, 4, 5, 6, 7, 9, 12]	90.0	90.0	**90.0**
		Step 2	[10, 11, 12, 13, 15, 16, 17, 18, 19, 27]	90.0	90.0	**90.0**
		Step 3	[12, 13, 20, 21, 22, 23, 24, 26, 27, 28, 29]	90.0	81.8	**85.9**
D→A	HBW [16]	Step 1	[0, 1, 6]	30.0	100.0	65.0
		Step 2	[0, 11, 12, 13, 14, 15, 16, 17, 19, 27,29]	80.0	66.7	73.4
		Step 3	[0, 14, 20, 21, 22, 23, 26, 27, 29]	70.0	77.8	**73.9**
	Ours	Step 1	[0, 1, 2, 3, 5, 6, 7, 9, 14]	80.0	88.9	**84.5**
		Step 2	[11, 12, 13, 14, 15, 16, 17, 19, 27]	80.0	88.9	**84.5**
		Step 3	[0, 14, 20, 21, 22, 23, 26, 27, 29]	70.0	77.8	**73.9**
D→W	HBW [16]	Step 1	[0, 2, 6]	30.0	100.0	65.0
		Step 2	[0, 6, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 22, 24, 25]	100.0	66.7	83.4
		Step 3	[0, 12, 15, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]	100.0	71.4	85.7
	Ours	Step 1	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	100.0	100.0	**100.0**
		Step 2	[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]	100.0	100.0	**100.0**
		Step 3	[20, 21, 22, 23, 24, 26, 27, 28, 29, 30]	90.0	90.0	**90.0**
W→A	HBW [16]	Step 1	[0, 1, 2, 5, 6, 9]	60.0	100.0	**80.0**
		Step 2	[0, 2, 4, 5, 7, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 23, 24, 26, 27, 29]	100.0	47.6	73.8
		Step 3	[0, 4, 5, 11, 13, 14, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29]	90.0	50.0	70.0
	Ours	Step 1	[0, 1, 2, 3, 5, 6, 7, 9, 11, 27, 29]	80.0	72.7	76.4
		Step 2	[10, 11, 12, 14, 15, 16, 17, 18, 19, 27]	90.0	90.0	**90.0**
		Step 3	[11, 14, 16, 18, 20, 21, 22, 23, 24, 26, 27, 29]	80.0	66.7	**73.4**
W→D	HBW [16]	Step 1	[0, 1, 2, 4, 6, 7, 8, 9]	80.0	100.0	90.0
		Step 2	[7, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]	100.0	90.9	95.5
		Step 3	[20, 21, 22, 23, 26, 27, 29]	70.0	100.0	85.0
	Ours	Step 1	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	100.0	100.0	**100.0**
		Step 2	[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]	100.0	100.0	**100.0**
		Step 3	[20, 21, 22, 23, 24, 25, 26, 27, 28, 29]	100.0	100.0	**100.0**

Relations to partial domain adaptation methods. PDA [1] assumes that the target label set is a subset of the source label set, and seeks to transfer a model trained from a big labeled source domain to a small unlabeled target domain. To alleviate the negative transfer caused by source private classes, existing PDA methods [1,2,16] decrease the transferability weights of source pri-

vate classes when aligning the source and target domains. To be specific, ETN [2] quantifies the *instance-level* transferability weights to all source samples. In contrast, PADA [1] and DPDAN [16] assign *class-level* transferability weights to all source classes. However, PADA directly exploits the cumulative probabilities as the transferability weights of all source classes, leading to the presence of the negative impact of source private classes in transfer. To address this, DPDAN [16] proposes the Hard Binary Weights (HBW) strategy which decomposes the source domain into two distributions (*i.e.,* source-positive and source-negative distributions). More specifically, to detect the shared classes, HBW sets the cumulative probabilities threshold by maximizing the variance of these distributions. Similar to HBW, our shared class detection strategy also alleviates the negative transfer at the class level and aims to eliminate the negative impact of source private classes.

However, HBW tends to fail in CI-UDA since the target label space is inconsistent between steps, *i.e.,* the shared classes between the source and the target are inconsistent between different steps. Unfortunately, the target label space inconsistency may bring noise into the optimization of the variance in HBW, resulting in false source-positive and source-negative distributions. In contrast, our strategy sets the pre-defined cumulative probabilities threshold α to detect shared classes, which is more robust to the variation of the target label space. To verify this, we visualize the detected shared classes by HBW and our strategy in different learning steps of CI-UDA. As shown in Table I, compared with our method, the HBW strategy detects more false shared classes (*e.g.,* Step 2 of A→D) and filters more shared classes out (*e.g.,* Step 1 of D→W) in CI-UDA. To quantify the results of shared class detection, we use two accuracy measures: 1) ***Shared Class Detection Accuracy (SCD Acc.)***: the truly shared classes divided by the number of ground truths; and 2) ***Total Class Detection Accuracy (TCD Acc.)***: the truly detected shared classes divided by the number of all detected classes. The Experiment shows that our shared class detection strategy outperforms the HBW strategy in CI-UDA with the higher average accuracy in almost all tasks on the Office-31-CI.

Relations to prototype-based methods. Existing prototype-based methods have separately explored prototypes to conduct domain alignment [32] or prevent catastrophic forgetting [36]. However, these methods are different from our ProCA. To be specific, existing prototype-based domain adaptation methods [32,35] conduct domain alignment by aligning source feature prototypes to all target data, while ProCA aligns class-wise source centers and the feature prototypes extracted from the target label prototypes. In addition, even though we select prototypes in the same manner with iCaRL [36] for replaying knowledge, iCaRL constructs the memory bank via images and ignores updating, while ProCA constructs the prototype memory bank based on our target label prototypes and designs a novel way to update this memory bank based on the cumulative probabilities. Note that obtaining image prototypes for knowledge retaining in iCaRL [36] requires data labels but the target domain in CI-UDA is totally unlabeled. Meanwhile, feature prototypes [32,35] for domain adapta-

tion cannot update the feature extractor, so simply detecting them is unable to overcome the knowledge forgetting issue of the feature extractor in CI-UDA. Therefore, a simple combination of existing prototype-based methods is not feasible for CI-UDA while our ProCA provides the first feasible prototype-based solution to CI-UDA (cf. Section 5 in the main paper).

B Pseudo-code of Label Prototype Identification

In this section, we present the pseudo-code of the prototype identification scheme. Specifically, we first detect shared classes in each time step and generate pseudo labels for target data. As shown in Algorithm 2, for each class k in the detected shared class set, we obtain T target label prototypes via a nearest neighbor approach.

Algorithm 2 Label Prototype identification of ProCA

Require: Pseudo-labeled target data $\mathcal{D}_t^k = \{\mathbf{x}_i^k\}_{i=1}^{n_k}$ of class k at the current time; Model G ; Hyper-parameter T.

1: Attain the k-th class feature center: $\mathbf{f}_t^k = \frac{1}{n_k}\sum_{i=1}^{n_k} G(\mathbf{x}_i^k)$;
2: **for** $m = 1 \to T$ **do**
3: $\quad \mathbf{p}_m^k = \arg\min_{\mathbf{x}^k \in \mathcal{D}_t^k} \left\| \mathbf{f}_t^k - \frac{1}{m}[G(\mathbf{x}^k) + \sum_{i=1}^{m-1} G(\mathbf{p}_i^k)] \right\|_2$;
4: **end for**
5: **return** Label prototypes of class k $\{\mathbf{p}_1^k, ..., \mathbf{p}_T^k\}$.

C Details of Data Construction

In this section, we show the containing classes in each disjoint subset of all the three benchmark datasets (*i.e.,* Office-31-CI, Office-Home-CI and ImageNet-Caltech-CI) in Tables III and II. Specifically, we choose 10 for the number of incremental classes on the three benchmark datasets. For Office-31-CI, we sort the class name in alphabetic order and group every 10 categories into a step. For Office-Home-CI, we randomly group every 10 categories into a step. As a result, each domain of Office-31-CI has 3 disjoint subsets for 3 time steps, while each domain of Office-Home-CI has 6 disjoint subsets for 6 time steps. For ImageNet-Caltech-CI, we adopt the class indexes following [37,12]. As shown in Table III, we also group every 10 categories into a time step based on the sorted class indexes. Thus, each domain of ImageNet-Caltech-CI has 8 disjoint subsets for 8 time steps. We have put the splits of three datasets into the code.

Table II. Class names in each time step on Office-31-CI and Office-Home-CI.

Dataset	Time Step	Class Index	Class Name
Office-31-CI	Step 1	[0,1,2,3,4,5,6,7,8,9]	back pack, bike, bike helmet, bookcase, bottle, calculator, desk chair, desk lamp, desktop computer, file cabinet
	Step 2	[10,11,12,13,14,15,16,17,18,19]	headphones, keyboard, laptop computer, letter tray, mobile phone, monitor, mouse, mug, paper notebook, pen
	Step 3	[20,21,22,23,24,25,26,27,28,29]	phone, printer, projector, punchers, ring binder, ruler, scissors, speaker, stapler, tape dispenser
Office-Home-CI	Step 1	[0,1,2,3,4,5,6,7,8,9]	Drill, Exit Sign, Bottle, Glasses, Computer, File Cabinet, Shelf, Toys, Sink, Laptop
	Step 2	[10,11,12,13,14,15,16,17,18,19]	Kettle, Folder, Keyboard, Flipflops, Pencil, Bed, Hammer, ToothBrush, Couch, Bike
	Step 3	[20,21,22,23,24,25,26,27,28,29]	Postit Notes, Mug, Webcam, Desk Lamp, Telephone, Helmet, Mouse, Pen, Monitor, Mop
	Step 4	[30,31,32,33,34,35,36,37,38,39]	Sneakers, Notebook, Backpack, Alarm Clock, Push Pin, Paper Clip, Batteries, Radio, Fan, Ruler
	Step 5	[40,41,42,43,44,45,46,47,48,49]	Pan, Screwdriver, Trash Can, Printer, Speaker, Eraser, Bucket, Chair, Calendar, Calculator
	Step 6	[50,51,52,53,54,55,56,57,58,59]	Flowers, Lamp Shade, Spoon, Candles, Clipboards Scissors, TV, Curtains, Fork, Soda

Table III. Class indexes in each time step on ImageNet-Caltech-CI.

Task	Time Step	Class Index
I→C	Step 1	[1, 9, 24, 39, 51, 69, 71, 79, 94, 99]
	Step 2	[112, 113, 145, 148, 171, 288, 308, 311, 314, 315]
	Step 3	[327, 334, 340, 354, 355, 361, 366, 367, 413, 414]
	Step 4	[417, 435, 441, 447, 471, 472, 479, 504, 508, 515]
	Step 5	[543, 546, 555, 560, 566, 571, 574, 579, 593, 594]
	Step 6	[604, 605, 620, 621, 637, 651, 664, 671, 713, 745]
	Step 7	[760, 764, 779, 784, 805, 806, 814, 839, 845, 849]
	Step 8	[852, 859, 870, 872, 876, 879, 895, 907, 910, 920]
C→I	Step 1	[0, 2, 7, 9, 11, 27, 28, 29, 30, 33]
	Step 2	[37, 39, 40, 44, 45, 47, 50, 60, 62, 68]
	Step 3	[71, 75, 76, 82, 85, 86, 87, 88, 89, 90]
	Step 4	[92, 94, 96, 97, 106, 107, 108, 109, 110, 112]
	Step 5	[114, 115, 116, 123, 126, 128, 133, 134, 141, 145]
	Step 6	[146, 150, 151, 157, 160, 163, 165, 170, 172, 177]
	Step 7	[178, 179, 181, 185, 188, 192, 193, 196, 198, 200]
	Step 8	[209, 211, 215, 219, 225, 227, 228, 229, 230, 234]

D More Implementation Details

We train ProCA using SGD optimizer with the learning rate, weight decay and momentum set to 1×10^{-3}, 1×10^{-6} and 0.9, respectively. When training in each time step, we update pseudo-labels, label prototypes and source centers every 4, 7 and 5 epochs. Due to the lack of train-validation splits in the three datasets, we report the results at the last epoch for all methods. Note that we do not exploit any additional target augmentation, *e.g.,* [7], for training or evaluation.

E More Results of Enhancing Partial Domain Adaptation

In this section, we apply ProCA to improve ETN [2] to fully investigate the ability of our method to boost existing PDA methods for handling CI-UDA. As shown in Table IV, ProCA could enhance existing partial domain adaptation methods to alleviate catastrophic forgetting and thus overcome CI-UDA.

Table IV. Comparisons of the existing partial domain adaptation methods with and without our label prototype identification strategy on **Office-31-CI**. We show the final accuracy (%) and final S-1 accuracy (%).

Method	Prototypes	Metric	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
ETN	✗	Final Acc. (%)	21.3	82.2	61.7	94.3	64.1	100.0	70.6
	✓		60.4	83.1	65.2	97.9	65.1	100.0	78.6
	✗	Final S-1 Acc. (%)	38.8	95.5	72.2	100.0	67.9	100.0	79.1
	✓		68.3	95.5	75.5	100.0	68.1	100.0	84.6
ProCA (ours)	✓	Final Acc. (%)	81.6	82.6	65.5	99.1	63.9	99.8	82.1
	✓	Final S-1 Acc. (%)	96.7	94.2	74.1	100.0	80.0	100.0	90.8

F More Ablation Studies

In this section, we first study the effect of three hyper-parameters (*i.e.,* λ, η and α) on three datasets. We fix the other hyper-parameters when studying ones. As shown in Table V, ProCA usually achieves the best performance in terms of Final Accuracy when setting $\lambda = 0.1$ and $\eta = 1.0$. Moreover, the results demonstrate that our method is non-sensitive for λ and η. Although ProCA may obtain the best performance in terms of Final Accuracy with a high α, we recommend setting a lower α, *e.g.,* 0.15, since a high threshold possibly filters shared classes out. One may concern false shared classes, but it can be handled by our method in fact since they would be updated by our label prototype identification strategy when a higher cumulative probability comes (c.f. Fig.3 in the main paper).

In addition, we train ProCA with a varying number of prototypes and incremental classes to investigate the effect of the number of target prototypes and that of incremental classes. As shown in Table VI, our method can achieve competitive performance (*i.e.,* 81.4% final Acc.) even with one prototype. With the increase of prototypes, the model retains the previous knowledge better and 20 prototypes in each class are sufficient for our ProCA. As for incremental classes, our method is non-sensitive to the number of incremental classes and performs well on all these settings. More specifically, when adding 10 classes each time step, ProCA achieves the best performance.

Table V. Effect of hyper-parameters λ, η and α on Office-31-CI (A→W, W→A), Office-Home-CI (Pr→Rw, Rw→Pr) and ImageNet-Caltech-CI (C→I). The value of λ is chosen from $[0, 0.05, 0.1, 0.2, 0.5, 1.0]$ and η is chosen from $[0, 0.1, 0.5, 1.0, 1.5, 2.0]$. Moreover, the value of α is chosen from $[0.1, 0.15, 0.2, 0.25, 0.30]$. In each experiment, the rest of hyper-parameters are fixed to the value reported in the main paper.

Dataset	Metric	λ						η						α				
		0	0.05	0.1	0.2	0.5	1.0	0	0.1	0.5	1.0	1.5	2.0	0.1	0.15	0.2	0.25	0.30
Office-31-CI	Final Acc.	84.7	84.9	**85.6**	84.5	84.3	84.1	84.1	84.7	84.6	**85.6**	84.4	84.9	81.6	85.6	85.3	**85.4**	83.5
	Final S-1 Acc.	85.1	85.4	**87.1**	85.8	85.7	84.3	84.8	85.4	87.0	87.1	**87.3**	86.8	86.6	**87.1**	85.9	86.1	85.2
Office-Home-CI	Final Acc.	72.5	73.2	**73.3**	73.0	73.0	73.2	72.5	72.6	72.9	**73.3**	73.2	73.3	73.7	73.3	73.1	**73.4**	73.0
	Final S-1 Acc.	77.5	78.3	**80.6**	78.3	77.1	76.1	76.7	79.0	78.2	**80.6**	77.4	79.0	73.4	**80.6**	79.5	78.9	74.2
ImageNet-Caltech-CI	Final Acc.	82.4	**83.5**	83.1	83.1	83.1	82.3	79.8	82.4	83.0	83.1	**83.5**	83.4	77.2	83.1	84.8	**87.8**	87.0
	Final S-1 Acc.	70.4	73.6	72.0	72.8	**74.2**	69.0	67.8	71.2	71.8	72.0	**73.4**	73.0	69.2	**72.0**	71.6	67.4	68.0

Table VI. Effect of the number of prototypes and incremental classes each time step on **ImageNet-Caltech-CI**. The number of prototypes is chosen from $[1, 5, 10, 20, 40]$ and the number of incremental classes is chosen from $[10, 15, 20, 30, 40]$. Note that we fix the other hyper-parameters when studying ones.

Setting	# Target Label Prototypes					# Incremental classes				
	1	5	10	20	40	10	15	20	30	40
Final Acc.	81.4	82.5	83.1	**83.9**	83.5	**83.1**	80.6	80.0	79.3	80.6
Final S-1 Acc.	65.8	70.6	72.0	73.8	**74.0**	**72.0**	69.0	71.0	68.2	67.8

G Effectiveness of Shared Class Detection

To further investigate the effectiveness of our shared class detection strategy, we compare our method with two variants. The first variant (*i.e.,* Pseudo-labeling) removes the shared class detection strategy and directly clusters target samples for all classes [26]. The second variant (*i.e.,* Pseudo-labeling with HBW) applies the HBW strategy [16] to the clustering method [26] and generates pseudo-labels for target samples. As shown in Table VII, the Final Accuracy of pseudo-labels of [26] yields inferior accuracy (74.1% Avg. Acc.) and even performs worse than source-only (77.5% Avg. Acc., ResNet-50) on Office-31-CI. This is because the pseudo labels generated by clustering may be noisy when facing the label space inconsistency between domains. The second variant also suffers performance degradation (66.6% Avg. Acc.). The reason lies in that the HBW strategy may fail to get the best variance of the source-positive and source-negative distributions in CI-UDA (cf. Appendix A), so it is unable to distinguish the source positive classes and the shared classes well (cf. Table I). In contrast, when using our shared class detection strategy, ProCA detects the shared classes well in various learning steps (cf. Table I) and thus achieves much better performance (cf. Table VII). Such a result demonstrates the superiority of our shared class detection strategy in CI-UDA over existing baselines.

Table VII. Final Accuracy (%) of the pseudo-labels with and without shared class detection (SCD) strategy on **Office-31-CI**.

Method	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
ResNet-50 [13]	74.1	74.4	58.5	96.9	61.2	99.6	77.5
Pseudo-labeling [26]	71.2	73.8	60.3	84.8	63.0	91.5	74.1
Pseudo-labeling [26] with HBW [16]	67.5	71.5	42.7	82.9	44.9	90.2	66.6
Pseudo-labeling with our SCD	**79.7**	**78.3**	**63.5**	**99.0**	**64.9**	**100.0**	**80.9**

H More Experimental Results

To evaluate the ability of our method in sequential learning, we report Step-level Accuracy and the average accuracy of step-1 classes in each time step (S-1 Accuracy) on ImageNet-Caltech-CI (Table VIII), Office-31-CI (Table IX) and Office-Home-CI (Tables X, XI, XII and XIII). The experiments show that: 1) ProCA achieves the best (or at least comparable) performance w.r.t. Step-level Accuracy on all steps of all transfer tasks, which demonstrates the effectiveness of our method. 2) Compared with the other baselines, ProCA shows the least S-1 Accuracy drop on most transfer tasks, which shows that the proposed ProCA is good at alleviating catastrophic forgetting.

Table VIII. Classification accuracies (%) on ImageNet-Caltech-CI. Note that the results outside the brackets are Step-level Accuracy, while the results in brackets represent the average accuracy of step-1 classes in each time step (S-1 Accuracy).

Task	Method	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Avg.
C→I	ResNet-50 [13]	50.6 (50.6)	53.8 (50.6)	64.1 (50.6)	66.4 (50.6)	66.6 (50.6)	69.7 (50.6)	71.3 (50.6)	71.2 (50.6)	61.9 (50.6)
	DANN [10]	47.6 (47.6)	52.4 (55.6)	53.9 (52.2)	53.0 (52.2)	53.0 (52.6)	51.9 (49.4)	58.0 (53.6)	58.8 (54.8)	53.6 (52.3)
	PADA [1]	52.1 (52.1)	38.8 (19.7)	41.1 (21.5)	28.7 (19.8)	34.5 (22.2)	30.4 (34.3)	31.6 (29.5)	37.3 (29.1)	36.8 (28.5)
	ETN [2]	54.6 (54.6)	55.3 (35.6)	25.3 (11.0)	9.4 (0.8)	4.4 (0.0)	3.4 (0.0)	1.8 (0.0)	1.4 (0.0)	19.4 (12.8)
	BA^{3}US [27]	65.6 (65.6)	52.2 (68.4)	54.1 (65.8)	59.4 (65.6)	59.4 (60.0)	58.5 (58.2)	61.9 (56.2)	60.8 (53.0)	59.0 (61.6)
	CIDA [21]	58.6 (58.6)	61.3 (56.8)	65.4 (58.4)	67.1 (56.4)	65.9 (55.0)	69.4 (55.8)	68.8 (53.8)	69.3 (58.0)	65.7 (56.6)
	ProCA (ours)	**74.4** (74.4)	**74.8** (73.6)	**75.2** (73.4)	**77.5** (73.6)	**78.5** (72.4)	**81.6** (72.0)	**82.1** (71.2)	**83.1** (72.0)	**78.4** (72.8)
I→C	ResNet-50 [13]	81.2 (81.2)	71.8 (81.2)	76.5 (81.2)	73.8 (81.2)	75.2 (81.2)	73.1 (81.2)	71.3 (81.2)	70.7 (81.2)	74.2 (81.2)
	DANN [10]	62.8 (63.4)	53.0 (72.8)	43.3 (66.8)	43.2 (56.5)	36.0 (49.0)	33.9 (43.8)	33.1 (43.9)	31.4 (37.2)	42.1 (54.2)
	PADA [1]	72.4 (72.0)	53.5 (52.7)	50.0 (54.8)	46.1 (51.5)	53.6 (44.4)	40.3 (43.4)	44.5 (42.2)	45.9 (51.0)	50.8 (51.5)
	ETN [2]	75.9 (75.8)	70.5 (78.4)	73.5 (79.4)	69.1 (78.8)	72.2 (79.5)	71.0 (79.2)	48.5 (48.3)	3.1 (0.0)	60.5 (64.9)
	BA^{3}US [27]	94.0 (94.1)	71.2 (95.4)	82.2 (95.3)	84.5 (93.5)	81.2 (92.3)	77.8 (90.7)	64.0 (79.7)	45.0 (64.7)	75.0 (88.2)
	CIDA [21]	78.2 (78.7)	58.8 (80.5)	61.5 (81.1)	55.9 (78.5)	59.5 (77.8)	58.2 (76.9)	59.1 (77.9)	49.2 (64.6)	60.1 (77.0)
	ProCA (ours)	**97.8** (97.7)	**85.5** (97.5)	**87.6** (96.7)	**85.4** (96.2)	**86.9** (96.3)	**85.3** (95.9)	**84.2** (96.4)	**82.8** (95.0)	**86.9** (96.5)

Table IX. Classification accuracies (%) on Office-31-CI. Note that the results outside the brackets are Step-level Accuracy, while the results in brackets represent the average accuracy of step-1 classes in each time step (S-1 Accuracy).

Task	Method	Step 1	Step 2	Step 3	Avg.
A→D	ResNet-50 [13]	89.0 (87.8)	75.8 (87.8)	74.1 (87.8)	79.6 (87.8)
	DANN [10]	87.0 (85.6)	77.1 (87.1)	74.9 (85.4)	79.7 (86.0)
	PADA [1]	88.3 (88.7)	63.5 (35.7)	56.9 (35.2)	69.6 (53.2)
	ETN [2]	96.8 (96.2)	63.5 (89.2)	21.3 (38.8)	60.5 (74.7)
	BA^{3}US [27]	89.0 (89.7)	76.8 (89.7)	74.1 (89.7)	80.0 (89.7)
	CIDA [21]	90.3 (89.4)	77.1 (88.5)	70.4 (86.5)	79.3 (88.1)
	ProCA (ours)	**97.4** (97.3)	**84.5** (96.7)	**81.6** (96.7)	**87.8** (96.9)
A→W	ResNet-50 [13]	85.5 (85.3)	75.9 (85.3)	74.4 (85.3)	78.6 (85.3)
	DANN [10]	85.1 (86.1)	75.2 (86.4)	72.5 (85.2)	77.6 (85.9)
	PADA [1]	84.7 (82.9)	72.0 (53.5)	61.5 (49.9)	72.7 (62.1)
	ETN [2]	**97.9** (96.5)	**85.6** (96.5)	82.2 (95.5)	88.6 (96.2)
	BA^{3}US [27]	92.3 (89.1)	84.5 (88.6)	73.3 (89.0)	83.4 (88.9)
	CIDA [21]	82.1 (82.5)	70.6 (84.1)	64.5 (79.8)	72.4 (82.1)
	ProCA (ours)	92.3 (93.7)	83.3 (94.2)	**82.6** (94.2)	**86.1** (94.0)
D→A	ResNet-50 [13]	68.8 (68.6)	68.6 (68.6)	58.5 (68.5)	65.3 (68.5)
	DANN [10]	68.7 (68.2)	64.9 (68.0)	55.7 (67.7)	63.1 (68.0)
	PADA [1]	78.0 (78.2)	57.8 (63.7)	12.5 (17.2)	49.4 (53.0)
	ETN [2]	80.3 (79.5)	73.5 (74.7)	61.7 (72.2)	71.8 (75.5)
	BA^{3}US [27]	**81.3**(81.0)	**78.0** (78.7)	63.3 (76.7)	**74.2** (78.8)
	CIDA [21]	71.0 (71.3)	63.9 (71.5)	48.1 (64.9)	61.0 (69.2)
	ProCA (ours)	78.0 (74.6)	75.0 (73.0)	**65.5** (74.1)	72.8 (73.9)
D→W	ResNet-50 [13]	100.0 (100.0)	99.1 (100.0)	96.9 (100.0)	98.7 (100.0)
	DANN [10]	85.1 (86.1)	75.2 (86.4)	72.5 (85.2)	77.6 (85.9)
	PADA [1]	84.7 (82.9)	72.0 (53.5)	61.5 (49.9)	72.7 (62.1)
	ETN [2]	97.9 (96.5)	85.6 (96.5)	82.2 (95.5)	88.6 (96.2)
	BA^{3}US [27]	100.0 (100.0)	98.1 (100.0)	94.8 (100.0)	97.6 (100.0)
	CIDA [21]	97.4 (97.8)	99.8 (100.0)	95.1 (99.0)	97.4 (99.0)
	ProCA (ours)	**100.0** (100.0)	**100.0** (100.0)	**99.1** (100.0)	**99.7** (100.0)
W→A	ResNet-50 [13]	70.9 (71.4)	71.5 (71.4)	61.2 (71.4)	67.9 (71.4)
	DANN [10]	54.0 (55.1)	62.7 (65.9)	51.4 (65.8)	56.0 (62.3)
	PADA [1]	74.2 (73.9)	60.5 (50.4)	46.7 (39.9)	60.5 (54.7)
	ETN [2]	76.9 (73.5)	74.8 (70.4)	**64.1** (67.9)	71.9 (70.6)
	BA^{3}US [27]	**81.7** (81.3)	**78.3** (79.2)	64.0 (77.3)	**74.7** (79.3)
	CIDA [21]	72.1 (71.8)	65.7 (71.8)	52.7 (70.6)	63.5 (71.4)
	ProCA (ours)	80.0 (82.0)	72.8 (81.3)	63.9 (80.0)	72.2 (81.1)
W→D	ResNet-50 [13]	100.0 (100.0)	99.7 (100.0)	99.6(100.0)	99.8 (100.0)
	DANN [10]	100.0 (100.0)	99.0 (99.2)	97.7 (99.2)	98.9 (99.5)
	PADA [1]	100.0 (100.0)	83.9 (66.2)	84.3 (72.8)	89.4 (79.7)
	ETN [2]	100.0 (100.0)	99.7 (100.0)	**100.0** (100.0)	99.9 (100.0)
	BA^{3}US [27]	100.0 (100.0)	**100.0** (100.0)	100.0 (99.8)	**100.0** (99.9)
	CIDA [21]	100.0 (100.0)	97.7 (100.0)	98.8 (100.0)	98.8 (100.0)
	ProCA (ours)	**100.0** (100.0)	99.7 (100.0)	99.8 (100.0)	99.8 (100.0)

Table X. Classification accuracies (%) on Office-Home-CI with Ar as source domain. Note that the results outside the brackets are Step-level Accuracy, while the results in brackets represent the average accuracy of step-1 classes in each time step (S-1 Accuracy).

Task	Method	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Avg.
Ar→Cl	ResNet-50 [13]	48.9 (51.2)	48.3 (51.2)	45.1 (51.2)	46.1 (51.2)	47.5 (51.2)	47.6 (51.2)	47.2 (51.2)
	DANN [10]	39.6 (43.6)	29.6 (43.9)	29.6 (43.8)	35.2 (48.0)	28.7 (36.6)	33.1 (39.3)	32.6 (42.5)
	PADA [1]	49.9 (52.0)	43.6 (37.0)	29.3 (30.1)	23.9 (23.7)	27.2 (35.0)	24.8 (30.7)	33.1 (34.8)
	ETN [2]	49.2 (51.8)	49.6 (50.4)	42.0 (51.2)	42.2 (50.2)	43.4 (51.3)	42.4 (51.4)	44.8 (51.1)
	BA^{3}US [27]	53.0 (53.0)	47.0 (54.7)	35.4 (54.9)	33.0 (54.9)	30.7 (52.5)	33.7 (54.6)	38.8 (54.1)
	CIDA [21]	50.5 (54.6)	45.1 (53.0)	40.4 (51.2)	37.3 (52.3)	34.8 (48.7)	32.2 (45.4)	40.1 (50.9)
	ProCA (ours)	**53.3** (57.6)	**54.2** (57.6)	**47.8** (57.3)	**51.4** (58.8)	**52.2** (58.2)	**51.5** (57.1)	**51.7** (57.8)
Ar→Pr	ResNet-50 [13]	66.5 (66.2)	62.9 (66.2)	62.7 (66.2)	64.5 (66.2)	65.6 (66.2)	65.2 (66.2)	64.6 (66.2)
	DANN [10]	51.0 (51.1)	45.3 (53.6)	39.8 (50.4)	40.6 (52.2)	45.9 (52.7)	40.0 (53.9)	43.8 (52.3)
	PADA [1]	61.1 (59.4)	42.7 (27.0)	44.2 (34.5)	41.2 (39.7)	39.9 (35.5)	41.4 (36.4)	45.1 (38.8)
	ETN [2]	65.2 (65.5)	71.3 (64.6)	64.8 (64.9)	64.4 (65.0)	59.4 (60.4)	2.8 (1.1)	54.7 (53.6)
	BA^{3}US [27]	62.0 (62.7)	50.0 (62.5)	42.3 (61.4)	39.7 (60.4)	41.0 (56.9)	39.7 (54.3)	45.8 (59.7)
	CIDA [21]	66.2 (65.8)	59.6 (64.2)	57.0 (63.8)	52.0 (61.8)	50.8 (61.5)	45.9 (55.9)	55.2 (62.2)
	ProCA (ours)	**86.7** (84.6)	**75.3** (84.1)	**74.0** (83.5)	**73.9** (79.5)	**75.3** (78.2)	**75.1** (77.1)	**76.7** (81.2)
Ar→Rw	ResNet-50 [13]	73.1 (72.3)	70.7 (72.3)	69.8 (72.3)	72.0 (72.3)	72.0 (72.3)	72.7 (72.3)	71.7 (72.3)
	DANN [10]	57.0 (55.8)	51.3 (56.6)	51.3 (60.6)	45.4 (55.3)	42.0 (49.3)	45.8 (54.7)	48.8 (55.4)
	PADA [1]	77.1 (74.9)	60.1 (43.3)	56.9 (42.6)	49.0 (36.9)	56.3 (42.2)	55.1 (43.8)	59.1 (47.3)
	ETN [2]	75.0 (74.6)	73.5 (73.6)	71.9 (72.2)	63.3 (59.2)	29.0 (25.1)	7.4 (0.2)	53.4 (50.8)
	BA^{3}US [27]	79.4 (78.2)	71.5 (77.6)	64.8 (77.8)	62.9 (77.4)	58.1 (74.3)	63.2 (74.7)	66.7 (76.7)
	CIDA [21]	67.6 (66.7)	60.5 (67.9)	60.2 (69.8)	58.7 (68.2)	58.3 (67.0)	49.1 (54.0)	59.1 (65.6)
	ProCA (ours)	**86.2** (86.5)	**86.2** (84.4)	**83.7** (83.5)	**85.3** (81.5)	**85.4** (82.6)	**85.9** (80.8)	**85.5** (83.2)

Table XI. Classification accuracies (%) on Office-Home-CI with Cl as source domain. Note that the results outside the brackets are Step-level Accuracy, while the results in brackets represent the average accuracy of step-1 classes in each time step (S-1 Accuracy).

Task	Method	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Avg.
Cl→Ar	ResNet-50 [13]	57.7 (58.1)	55.4 (58.1)	52.7 (58.1)	52.7 (58.1)	51.9 (58.1)	54.7 (58.1)	54.2 (58.1)
	DANN [10]	30.9 (30.5)	32.1 (39.0)	34.0 (47.0)	35.4 (45.3)	35.7 (45.7)	36.8 (44.1)	34.1 (41.9)
	PADA [1]	58.7 (52.9)	41.1 (30.3)	31.5 (27.5)	25.4 (19.9)	19.8 (20.3)	18.3 (18.5)	32.5 (28.2)
	ETN [2]	63.3 (63.2)	57.3 (63.0)	54.2 (63.8)	55.5 (59.6)	25.9 (25.5)	4.3 (0.2)	43.4 (45.9)
	BA^{3}US [27]	**71.4** (68.7)	57.8 (69.3)	48.5 (68.8)	37.5 (64.8)	40.2 (56.4)	36.6 (59.6)	48.7 (64.6)
	CIDA [21]	41.8 (46.0)	41.6 (61.3)	37.1 (55.0)	35.2 (59.3)	35.4 (59.8)	36.5 (57.8)	37.9 (56.5)
	ProCA (ours)	66.1 (63.6)	**64.9** (64.4)	**60.9** (64.7)	**59.7** (62.2)	**58.8** (63.6)	**60.9** (63.5)	**61.9** (63.7)
Cl→Pr	ResNet-50 [13]	67.3 (68.9)	63.8 (68.9)	63.5 (68.9)	60.8 (68.9)	62.6 (68.9)	62.8 (68.9)	63.5 (68.9)
	DANN [10]	40.1 (39.9)	41.3 (40.7)	36.7 (40.1)	39.4 (48.9)	39.4 (45.6)	36.6 (47.3)	38.9 (43.8)
	PADA [1]	75.2 (74.9)	48.0 (31.8)	39.3 (29.8)	36.7 (24.2)	36.6 (23.1)	35.0 (23.6)	45.1 (34.6)
	ETN [2]	67.1 (67.0)	63.6 (69.0)	62.0 (66.1)	63.7 (68.6)	63.7 (68.8)	60.3 (68.3)	63.4 (68.0)
	BA^{3}US [27]	70.0 (70.6)	63.3 (70.7)	58.4 (72.3)	52.9 (73.0)	46.2 (71.5)	39.1 (64.5)	55.0 (70.4)
	CIDA [21]	60.1 (60.2)	54.2 (65.5)	53.9 (66.3)	50.0 (66.4)	50.0 (66.7)	48.6 (61.6)	52.8 (64.5)
	ProCA (ours)	**71.3** (75.3)	**71.1** (75.4)	**69.8** (75.4)	**65.8** (74.7)	**69.7** (73.9)	**69.7** (74.0)	**69.6** (74.8)
Cl→Rw	ResNet-50 [13]	66.7 (66.0)	65.9 (66.0)	65.6 (66.0)	64.4 (66.0)	65.5 (66.0)	66.1 (66.0)	65.7 (66.0)
	DANN [10]	46.9 (46.9)	46.7 (51.7)	49.1 (54.0)	44.1 (49.1)	44.0 (51.2)	44.1 (53.4)	45.8 (51.1)
	PADA [1]	62.5 (63.3)	49.2 (31.6)	43.5 (36.0)	41.2 (30.2)	34.3 (25.6)	36.3 (30.1)	44.5 (36.1)
	ETN [2]	63.5 (64.0)	65.3 (65.3)	65.2 (63.5)	65.1 (62.9)	39.8 (37.7)	6.3 (0.5)	50.9 (49.0)
	BA^{3}US [27]	**75.6** (74.7)	68.6 (75.6)	63.1 (74.6)	54.3 (72.5)	46.0 (68.8)	53.7 (72.9)	60.2 (73.2)
	CIDA [21]	64.9 (64.2)	50.4 (63.9)	52.7 (62.8)	52.1 (62.8)	50.8 (62.9)	46.6 (56.0)	52.9 (62.1)
	ProCA (ours)	71.9 (71.6)	**74.6** (69.9)	**74.1** (70.0)	**73.2** (67.4)	**76.0** (67.2)	**75.3** (69.4)	**74.2** (69.3)

Table XII. Classification accuracies (%) on Office-Home-CI with Pr as the source domain. Note that the results outside the brackets are Step-level Accuracy, while the results in brackets represent the average accuracy of step-1 classes in each time step (S-1 Accuracy).

Task	Method	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Avg.
Pr→Ar	ResNet-50 [13]	49.1 (49.5)	50.7 (49.5)	47.7 (49.5)	50.8 (49.5)	50.5 (49.5)	52.4 (49.5)	50.2 (49.5)
	DANN [10]	34.2 (31.2)	25.6 (27.9)	32.3 (35.4)	32.5 (38.4)	30.4 (36.8)	32.0 (35.9)	31.2 (34.3)
	PADA [1]	52.2 (49.0)	36.4 (26.4)	26.2 (17.4)	28.1 (15.9)	25.5 (16.4)	25.9 (12.6)	32.4 (23.0)
	ETN [2]	**63.8** (63.9)	55.8 (60.5)	53.7 (60.7)	53.0 (61.0)	51.5 (58.7)	50.7 (61.6)	54.8 (61.1)
	BA^{3}US [27]	62.5 (59.7)	56.3 (61.0)	47.2 (60.7)	41.7 (63.8)	39.0 (62.5)	36.5 (64.6)	47.2 (62.1)
	CIDA [21]	49.1 (51.9)	48.9 (52.8)	48.5 (52.6)	50.0 (52.0)	50.2 (51.7)	51.6 (52.5)	49.7 (52.2)
	ProCA (ours)	62.0 (60.8)	**66.0** (61.2)	**60.1** (60.0)	**61.3** (57.8)	**60.0** (58.6)	**59.9** (57.7)	**61.6** (59.4)
Pr→Cl	ResNet-50 [13]	49.8 (50.3)	47.3 (50.3)	47.2 (50.3)	46.7 (50.3)	46.3 (50.3)	44.7 (50.3)	47.0 (50.3)
	DANN [10]	37.5 (39.6)	33.4 (42.3)	33.1 (38.6)	30.7 (34.4)	23.3 (30.0)	29.8 (37.4)	31.3 (37.1)
	PADA [1]	49.3 (46.6)	35.0 (35.0)	28.5 (33.2)	24.4 (23.2)	29.0 (37.7)	26.2 (27.5)	32.1 (33.9)
	ETN [2]	47.7 (48.9)	42.4 (48.4)	38.7 (49.3)	36.3 (48.4)	38.2 (47.4)	33.8 (49.0)	39.5 (48.6)
	BA^{3}US [27]	55.4 (57.4)	44.5 (57.6)	38.0 (56.4)	32.3 (54.6)	29.5 (52.1)	24.9 (50.0)	37.4 (54.7)
	CIDA [21]	50.7 (51.6)	42.8 (51.7)	39.4 (51.3)	36.2 (50.2)	36.6 (50.5)	33.5 (49.0)	39.9 (50.7)
	ProCA (ours)	**60.4** (61.6)	**54.6** (59.6)	**54.2** (58.9)	**53.7** (59.0)	**53.2** (57.0)	**50.9** (58.2)	**54.5** (59.1)
Pr→Rw	ResNet-50 [13]	70.1 (69.5)	72.0 (69.5)	72.3 (69.5)	73.1 (69.5)	74.1 (69.5)	74.0 (69.5)	72.6 (69.5)
	DANN [10]	46.3 (46.0)	42.9 (43.6)	48.3 (48.8)	49.7 (50.8)	47.4 (46.4)	49.8 (49.3)	47.4 (47.5)
	PADA [1]	71.6 (71.5)	62.8 (43.4)	46.1 (33.6)	53.8 (43.3)	47.4 (30.8)	53.7 (40.3)	55.9 (43.8)
	ETN [2]	72.8 (73.1)	70.5 (69.6)	71.2 (71.5)	72.4 (71.0)	71.5 (69.5)	70.8 (70.2)	71.5 (70.8)
	BA^{3}US [27]	79.8 (79.3)	73.6 (78.8)	69.3 (77.8)	63.4 (76.8)	59.1 (71.9)	53.4 (65.9)	66.4 (75.1)
	CIDA [21]	65.0 (64.9)	62.3 (68.0)	61.4 (67.1)	59.4 (65.9)	59.7 (65.9)	59.0 (62.3)	61.1 (65.7)
	ProCA (ours)	**81.3** (80.9)	**83.4** (79.6)	**83.2** (78.8)	**85.4** (79.1)	**83.3** (78.7)	**84.7** (79.3)	**83.6** (79.4)

Table XIII. Classification accuracies (%) on Office-Home-CI with Rw as source domain. Note that the results outside the brackets are Step-level Accuracy, while the results in brackets represent the average accuracy of step-1 classes in each time step (S-1 Accuracy).

Task	Method	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Avg.
Rw→Ar	ResNet-50 [13]	63.0 (65.0)	66.0 (65.0)	64.5 (65.0)	67.0 (65.0)	64.4 (65.0)	66.2 (65.0)	65.2 (65.0)
	DANN [10]	37.2 (39.9)	43.9 (47.5)	39.9 (44.6)	41.7 (44.3)	44.3 (47.8)	42.4 (49.0)	41.6 (45.5)
	PADA [1]	70.9 (70.1)	59.3 (37.2)	50.6 (34.0)	40.9 (24.8)	39.5 (17.9)	46.8 (32.8)	51.3 (36.1)
	ETN [2]	68.6 (68.7)	68.1 (67.2)	67.3 (68.2)	66.2 (68.4)	44.4 (51.2)	3.7 (0.9)	53.1 (54.1)
	BA^{3}US [27]	65.1 (68.8)	64.3 (69.0)	58.2 (69.3)	57.4 (68.4)	55.2 (68.8)	52.2 (69.2)	58.7 (68.9)
	CIDA [21]	62.3 (65.1)	63.5 (66.9)	61.9 (67.0)	62.9 (64.7)	62.4 (65.4)	64.0 (69.0)	62.8 (66.3)
	ProCA (ours)	**77.5** (74.0)	**79.3** (74.4)	**74.9** (74.0)	**76.0** (73.9)	**72.7** (72.0)	**75.8** (73.9)	**76.0** (73.7)
Rw→Cl	ResNet-50 [13]	40.1 (43.8)	48.2 (43.8)	48.0 (43.8)	48.6 (43.8)	48.7 (43.8)	47.4 (43.8)	46.8 (43.8)
	DANN [10]	37.8 (42.5)	38.8 (41.5)	39.9 (45.1)	41.1 (43.2)	40.8 (43.7)	40.2 (47.1)	39.7 (43.9)
	PADA [1]	50.2 (52.7)	**52.2** (37.3)	38.8 (27.6)	34.9 (23.8)	38.1 (30.9)	31.4 (33.2)	40.9 (34.3)
	ETN [2]	46.4 (50.6)	50.5 (50.8)	45.0 (50.9)	45.6 (49.9)	45.2 (51.4)	43.5 (51.8)	46.0 (50.9)
	BA^{3}US [27]	**51.6** (55.5)	52.3 (56.6)	45.6 (54.6)	42.5 (56.2)	38.2 (54.7)	35.9 (55.0)	44.3 (55.4)
	CIDA [21]	44.6 (47.3)	45.4 (46.0)	41.3 (46.5)	41.2 (49.6)	40.0 (45.0)	38.0 (46.2)	41.7 (46.8)
	ProCA (ours)	45.4 (47.3)	46.7 (48.4)	**47.9** (48.8)	**50.8** (50.4)	**51.5** (49.4)	**51.0** (47.3)	**48.9** (48.6)
Rw→Pr	ResNet-50 [13]	67.3 (69.2)	74.0 (69.2)	78.1 (69.2)	77.9 (69.2)	78.1 (69.2)	77.4 (69.2)	75.5 (69.2)
	DANN [10]	53.5 (54.7)	55.4 (56.3)	58.7 (60.4)	57.1 (60.6)	55.5 (55.1)	55.2 (57.2)	55.9 (57.4)
	PADA [1]	77.6 (80.6)	61.0 (35.9)	50.5 (22.8)	52.8 (41.3)	55.8 (42.0)	50.0 (47.4)	57.9 (45.0)
	ETN [2]	70.0 (72.3)	75.9 (72.9)	74.4 (71.3)	76.5 (72.8)	76.2 (72.9)	75.1 (73.3)	74.7 (72.6)
	BA^{3}US [27]	73.4 (76.5)	75.4 (76.6)	72.8 (75.7)	71.6 (74.9)	69.0 (76.4)	65.9 (73.2)	71.3 (75.6)
	CIDA [21]	71.6 (73.5)	69.6 (70.4)	68.3 (70.7)	66.4 (68.7)	66.2 (68.0)	65.1 (68.5)	67.9 (69.9)
	ProCA (ours)	**80.6** (85.1)	**85.0** (83.9)	**88.3** (82.9)	**85.4** (82.7)	**86.0** (82.5)	**86.4** (81.8)	**85.3** (83.2)

References

1. Cao, Z., Ma, L., Long, M., et al.: Partial adversarial domain adaptation. In: ECCV (2018)
2. Cao, Z., You, K., Long, M., et al.: Learning to transfer examples for partial domain adaptation. In: CVPR. pp. 2985–2994 (2019)
3. Castro, F.M., Marín-Jiménez, M.J., Guil, N., et al.: End-to-end incremental learning. In: ECCV. pp. 233–248 (2018)
4. Chen, C., Fu, Z., Chen, Z., et al.: Homm: Higher-order moment matching for unsupervised domain adaptation. In: AAAI. pp. 3422–3429 (2020)
5. Chen, S., Harandi, M., Jin, X., Yang, X.: Domain adaptation by joint distribution invariant projections. IEEE Transactions on Image Processing pp. 8264–8277 (2020)
6. Chen, Y., Li, W., Sakaridis, C., et al.: Domain adaptive faster r-cnn for object detection in the wild. In: CVPR. pp. 3339–3348 (2018)
7. Cubuk, E.D., Zoph, B., et al.: Autoaugment: Learning augmentation policies from data. ArXiv (2018)
8. Du, Z., Li, J., Su, H., Zhu, L., Lu, K.: Cross-domain gradient discrepancy minimization for unsupervised domain adaptation. In: CVPR. pp. 3937–3946 (2021)
9. Fu, B., Cao, Z., Long, M., Wang, J.: Learning to detect open classes for universal domain adaptation. In: ECCV. pp. 567–583 (2020)
10. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: ICML (2015)
11. Gong, R., Li, W., Chen, Y., et al.: Dlow: Domain flow for adaptation and generalization. In: CVPR. pp. 2477–2486 (2019)
12. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset (2007)
13. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: CVPR (2016)
14. Hoffman, J., Tzeng, E., Park, T., et al.: Cycada: Cycle-consistent adversarial domain adaptation. In: ICML (2018)
15. Hu, D., Liang, J., Hou, Q., Yan, H., Chen, Y.: Adversarial domain adaptation with prototype-based normalized output conditioner. IEEE Transactions on Image Processing (2021)
16. Hu, J., Tuo, H., Wang, C., et al.: Discriminative partial domain adversarial network. In: ECCV. pp. 632–648 (2020)
17. Inoue, N., Furuta, R., Yamasaki, T., et al.: Cross-domain weakly-supervised object detection through progressive domain adaptation. In: CVPR. pp. 5001–5009 (2018)
18. Kang, G., Jiang, L., Yang, Y., et al.: Contrastive adaptation network for unsupervised domain adaptation. In: CVPR. pp. 4893–4902 (2019)
19. Khosla, P., Teterwak, P., Wang, C., et al.: Supervised contrastive learning. In: NeurIPS (2020)
20. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (2017)
21. Kundu, J.N., Venkatesh, R.M., Venkat, N., et al.: Class-incremental domain adaptation. In: ECCV. pp. 53–69 (2020)
22. Lao, Q., Jiang, X., Havaei, M., et al.: Continuous domain adaptation with variational domain-agnostic feature replay. ArXiv (2020)
23. Li, C., Lee, G.H.: From synthetic to real: Unsupervised domain adaptation for animal pose estimation. In: CVPR. pp. 1482–1491 (2021)

24. Li, G., Kang, G., Zhu, Y., et al.: Domain consensus clustering for universal domain adaptation. In: CVPR. pp. 9757–9766 (2021)
25. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**, 2935–2947 (2018)
26. Liang, J., Hu, D., Feng, J.: Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In: ICML (2020)
27. Liang, J., Wang, Y., Hu, D., et al.: A balanced and uncertainty-aware approach for partial domain adaptation. In: ECCV. pp. 123–140 (2020)
28. Liu, X., Masana, M., Herranz, L., et al.: Rotate your networks: Better weight consolidation and less catastrophic forgetting. International Conference on Pattern Recognition pp. 2262–2268 (2018)
29. Melas-Kyriazi, L., Manrai, A.K.: Pixmatch: Unsupervised domain adaptation via pixelwise consistency training. In: CVPR. pp. 12435–12445 (2021)
30. Na, J., Jung, H., Chang, H.J., Hwang, W.: Fixbi: Bridging domain spaces for unsupervised domain adaptation. In: CVPR. pp. 1094–1103 (2021)
31. Niu, S., Wu, J., Zhang, Y., et al.: Efficient test-time model adaptation without forgetting. In: ICML (2022)
32. Pan, Y., Yao, T., Li, Y., et al.: Transferrable prototypical networks for unsupervised domain adaptation. In: CVPR (2019)
33. Panareda Busto, P., Gall, J.: Open set domain adaptation. In: ICCV. pp. 754–763 (2017)
34. Pei, Z., Cao, Z., Long, M., et al.: Multi-adversarial domain adaptation. In: AAAI (2018)
35. Qiu, Z., Zhang, Y., Lin, H., et al.: Source-free domain adaptation via avatar prototype generation and adaptation. In: IJCAI (2021)
36. Rebuffi, S.A., Kolesnikov, A., Sperl, G., et al.: icarl: Incremental classifier and representation learning. In: CVPR. pp. 5533–5542 (2017)
37. Russakovsky, O., Deng, J., Su, H., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
38. Saenko, K., Kulis, B., Fritz, M., et al.: Adapting visual category models to new domains. In: ECCV (2010)
39. Saito, K., Kim, D., Sclaroff, S., et al.: Universal domain adaptation through self supervision. NeurIPS pp. 16282–16292 (2020)
40. Saito, K., Watanabe, K., Ushiku, Y., et al.: Maximum classifier discrepancy for unsupervised domain adaptation. In: CVPR. pp. 3723–3732 (2018)
41. Sankaranarayanan, S., Balaji, Y., Castillo, C.D., et al.: Generate to adapt: Aligning domains using generative adversarial networks. In: CVPR (2018)
42. Tang, H., Chen, K., Jia, K.: Unsupervised domain adaptation via structurally regularized deep clustering. In: CVPR (2020)
43. Tang, S., Su, P., Chen, D., et al.: Gradient regularized contrastive learning for continual domain adaptation. In: AAAI. pp. 2–13 (2021)
44. Tzeng, E., Hoffman, J., Saenko, K., et al.: Adversarial discriminative domain adaptation. In: CVPR. pp. 2962–2971 (2017)
45. Tzeng, E., Hoffman, J., Zhang, N., et al.: Deep domain confusion: Maximizing for domain invariance. ArXiv (2014)
46. Venkateswara, H., Eusebio, J., Chakraborty, S., et al.: Deep hashing network for unsupervised domain adaptation. In: CVPR (2017)
47. Wu, Y., Chen, Y., Wang, L., et al.: Large scale incremental learning. In: CVPR. pp. 374–382 (2019)
48. Xia, H., Ding, Z.: Hgnet: Hybrid generative network for zero-shot domain adaptation. In: ECCV. pp. 55–70 (2020)

49. Xie, X., Chen, J., Li, Y., et al.: Self-supervised cyclegan for object-preserving image-to-image domain adaptation. In: ECCV. pp. 498–513 (2020)
50. Xu, M., Islam, M., Lim, C.M., et al.: Class-incremental domain adaptation with smoothing and calibration for surgical report generation. In: MICCAI. pp. 269–278 (2021)
51. Yang, J., Shi, S., Wang, Z., et al.: St3d: Self-training for unsupervised domain adaptation on 3d object detection. In: CVPR. pp. 10363–10373 (2021)
52. Yang, J., Shi, S., Wang, Z., et al.: St3d: Self-training for unsupervised domain adaptation on 3d object detection. In: CVPR. pp. 10368–10378 (2021)
53. You, K., Long, M., Cao, Z., et al.: Universal domain adaptation. In: CVPR. pp. 2720–2729 (2019)
54. Zenke, F., Poole, B., Ganguli, S.: Continual learning through synaptic intelligence. In: ICML. pp. 3987–3995 (2017)
55. Zhang, Y., David, P., Gong, B.: Curriculum domain adaptation for semantic segmentation of urban scenes. In: ICCV. pp. 2039–2049 (2017)
56. Zhang, Y., Chen, H., Wei, Y., et al.: From whole slide imaging to microscopy: Deep microscopy adaptation network for histopathology cancer image classification. In: MICCAI (2019)
57. Zhang, Y., Hooi, B., Hong, L., et al.: Unleashing the power of contrastive self-supervised visual models via contrast-regularized fine-tuning. In: NeurIPS (2021)
58. Zhang, Y., Kang, B., Hooi, B., Yan, S., Feng, J.: Deep long-tailed learning: A survey. Arxiv (2021)
59. Zhang, Y., Wei, Y., Wu, Q., Zhao, P., Niu, S., Huang, J., Tan, M.: Collaborative unsupervised domain adaptation for medical image diagnosis. IEEE Transactions on Image Processing **29**, 7834–7844 (2020)
60. Zou, Y., Yu, Z., Kumar, B.V.K.V., et al.: Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: ECCV (2018)