

Supplementary Material

Class-Incremental Domain Adaptation

Reference in paper	Contents	Section in Suppl.
§2	Experimental details of STA	1
§3	Notation Table	2
§3	Algorithms	3
§3.1b	Generation of <i>negative</i> samples ($\mathbf{u}_n$)	3.1
§3.1c	Foresighted source-model training	3.2
§3.2c	Selection of confident target samples ($\mathcal{B}_t^c$)	3.3
§3.2e	Class-Incremental Domain Adaptation	3.4
§3.2d	Pseudo-labeling precision	4
§4.1	Dataset (label-set) details	5
§4	Architecture details	6
§4.1b	Office and Visda datasets	6.1
§4.1b	Digits dataset	6.2
§4.2a	Illustrations for baselines	6.3
§4.2b	Theoretical Insights	7
§4.2f	Discussion on multi-step model upgrade	8

1 Experimental details of STA

The label-sets used for the experiment are as mentioned in Sec. 5 and the architecture is given in [7]. After the adaptation, we predict labels for all target samples and collect the samples predicted as *unknown*. We then label 5% of these samples (ensuring equal number of samples in each class) to obtain few-shot target-private samples. Note that the few-shot target-private samples do not contain the mis-classified target-shared samples. Finally, we learn a target-private classifier (TPC) that is similar to g_t (Table 2).

We evaluate the method over 5 separate runs (with different sets of few-shot samples). First, we classify all target samples into $\mathcal{C}_s \cup \{\text{unknown}\}$. Then, samples predicted as *unknown* are further classified into $\mathcal{C}'_t$ using the TPC, thereby assigning a unique target-private label to each sample predicted as *unknown*. With target samples classified into $|\mathcal{C}_t|$ classes, we obtain the accuracy on target-shared samples ($95.9 \pm 0.3\%$) and target-private samples ($17.7 \pm 3.5\%$).

This evaluation scheme is followed for all UDA methods. Particularly, in DANN [3] and UAN [13], we obtain the *unknown* label prediction using confidence thresholding. In STA [7] and OSBP [12], the classifier is trained with an additional *unknown* class that is used to detect *unknown* samples.

2 Notation Table

In Table 1, we provide an exhaustive list of symbols used in the paper and the corresponding description.

Table 1. Notations used in the paper

	Symbol	Description
Space	$\mathcal{X}$	Input space
	$\mathcal{Y}$	Label space
	$\mathcal{U}$	Output space of f_s
	$\mathcal{V}$	Output space of f_t
Distr.	p	Source distribution on $\mathcal{X} \times \mathcal{Y}$
	q	Target distribution on $\mathcal{X} \times \mathcal{Y}$
	$q_{\mathcal{X}}$	Target marginal input distribution
	r	<i>Negative</i> data distribution in $\mathcal{U}$ -space
Network	f_s	Feature extractor of the source-model
	g_s	Classifier of the source-model
	f_t	Target-specific feature extractor
	g_t	Classifier accounting for $\mathcal{C}'_t$
	f_e, f_d	Domain projection networks
Sets	$\mathcal{D}_s$	Set of labeled source samples
	$\mathcal{D}_t$	Set of unlabeled target samples
	$\mathcal{D}_n$	Set of labeled <i>Negative</i> samples
	$\mathcal{C}_s, \mathcal{C}_t, \mathcal{C}'_t$	Set of shared classes, target classes and target-private classes
	$\mathcal{B}^c_t$	Set of confident target samples for each class $c \in \mathcal{C}_t$
Samples	$\mathbf{x}_s, \mathbf{x}_t$	Source and Target domain input samples (image)
	y_s, y_t	Source and Target domain labels
	$\{(\tilde{\mathbf{x}}^c_t, \tilde{y}^c_t)\}$	One-shot target-private samples $\forall c \in \mathcal{C}'_t$
	$\hat{y}(\cdot)$	Concatenated output pertaining to $\mathcal{C}_s$ ($g_s(\cdot) _{c \in \mathcal{C}_s}$) and $\mathcal{C}_t$ (g_t)
	$\mathbf{u}_s$	Source / proxy-source sample feature in the $\mathcal{U}$ -space
	$\mathbf{u}_n$	<i>Negative</i> sample feature in the $\mathcal{U}$ -space
	$\mathbf{v}_t$	Target sample feature in the $\mathcal{V}$ -space
	$\mathbf{v}^c_g$	<i>Guide</i> for class $c \in \mathcal{C}_t$ in the $\mathcal{V}$ -space
Prototypes	$\mathcal{P}^c_s, \mathcal{P}^c_t$	Source and Target domain class-specific Gaussian Prototypes
	$\mathcal{P}_s, \mathcal{P}_t$	Source and Target domain Global Gaussian Prototypes
	$\boldsymbol{\mu}^c_s, \boldsymbol{\Sigma}^c_s$	Class-specific mean and covariance of source features $f_s(\mathbf{x}_s)$
	$\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s$	Mean and covariance of source features $f_s(\mathbf{x}_s)$ for all $\mathbf{x}_s$ in $\mathcal{D}_s$
	$\boldsymbol{\mu}^c_t, \boldsymbol{\Sigma}^c_t$	Class-specific mean and covariance of target features $f_t(\mathbf{x}_t)$
	$\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$	Mean and covariance of source features $f_t(\mathbf{x}_t)$ for all $\mathbf{x}_t$ in $\mathcal{D}_t$
Misc.	$\mathbb{E}$	Expectation
	$l_{ce}(\cdot, \cdot)$	Standard cross-entropy function
	$l_2(\cdot, \cdot)$	Euclidean distance function
	d, k	Distance to the nearest <i>guide</i> and pseudo-label of a target sample
	n	Percentage of confident target samples during CIDA
	N_{src}, N_{neg}	Number of source and <i>negative</i> samples in a batch

3 Algorithms

Here we provide the algorithms corresponding to each stage of our approach, along with a description of the major steps involved.

3.1 Generation of *negative* samples ($\mathbf{u}_n$)

During the source-model training, we generate *negative* samples at each iteration as outlined in Algo. 1. Specifically, *negative* samples are those samples obtained from the global Gaussian Prototype $\mathcal{P}_s = \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s)$ that fall outside the $3\text{-}\sigma$ confidence threshold of each class-specific Gaussian Prototype $\mathcal{P}_s^c = \mathcal{N}(\boldsymbol{\mu}_s^c, \boldsymbol{\Sigma}_s^c)$.

This $3\text{-}\sigma$ threshold is calculated in lines 3-6 of Algo. 1, where we obtain the eigen decomposition of the covariance matrix $\boldsymbol{\Sigma}_s^c$ and select the largest eigen value λ_s^c and the corresponding eigen vector $\boldsymbol{\nu}_s^c$. Then, we define the $3\text{-}\sigma$ threshold for class c as the likelihood of the sample $\boldsymbol{\mu}_s^c + 3 \cdot \sqrt{\lambda_s^c} \cdot \boldsymbol{\nu}_s^c$.

To generate *negative* samples ($\mathbf{u}_n$), we obtain samples from the global Gaussian Prototype $\mathcal{P}_s$ (line 8) and reject those samples that are within the $3\text{-}\sigma$ confidence interval of any class. In other words, we keep those samples which do not fall within the $3\text{-}\sigma$ interval of any class (lines 10-12).

3.2 Foresighted source-model training

The source-model training is outlined in Algo. 2. We pre-train the network for 1 epoch (lines 2-5) using the vanilla cross-entropy loss (to provide a fair initialization for the Gaussian Prototypes). Here, line 4 denotes the parameter update step performed by gradient descent using the Adam optimizer. The term inside the *Adam* operator denotes the (negative) gradient of the mean loss for the mini-batch of the source samples.

After this initialization, we calculate the Gaussian Prototypes, both class-specific (lines 6-11) and global (lines 12-14). Note that, in the paper we mention that an alternating minimization scheme is used for training the network. This is implemented by maintaining a list of corresponding losses, optimizers and model parameters. Particularly, use a separate *Adam* optimizer (line 16) for each loss (line 15) that trains only the parameters specific to the loss (line 17). Further, we maintain a counter *iter* (line 18) that is used to determine the loss that is to be optimized at each iteration ($cur = iter \bmod 2$).

During training, we obtain a mini-batch with an equal amount of source (N_{src}) and *negative* samples (N_{neg}), i.e. $N_{src} = N_{neg}$ (lines 22-29). Line 30 denotes the optimization process, where *cur* is used to index the list of parameters $\boldsymbol{\Theta}$, optimizers **Opt** and losses **Loss**. Note, here the update step denotes the update of the parameters $\boldsymbol{\Theta}[cur]$ specific to the loss **Loss**[*cur*] being optimized by the optimizer **Opt**[*cur*] (similar to that in line 4). Finally, at the end of each epoch, we recalculate the Gaussian Prototypes which are then used for the next epoch.

3.3 Selection of confident target samples $\mathcal{B}_t^c$

The process of confident target sample ($\mathcal{B}_t^c$) selection is shown in Algo. 3. Specifically, for each *guide* $\mathbf{v}_g^c$, we obtain the nearest n -percent pseudo-labeled target samples. Lines 3-7 denote the pseudo-labeling process, while the selection of nearest n -percent target samples is shown in lines 8-12.

3.4 Class-Incremental Domain Adaptation

We outline the algorithm for performing CIDA on the target domain in Algo. 4. The target feature extractor f_t is initialized from f_s and the domain projection networks f_e, f_d are initialized with near-identity mappings (to effectively “initialize $\mathcal{V}$ -space with the $\mathcal{U}$ -space”). This is depicted in lines 2-7. Particularly, we use separate Adam optimizers ($Adam_{f_e}$ for training f_e , and $Adam_{f_d}$ for training f_d) and train the networks with the reconstruction losses (similar to a vanilla auto-encoder). Note that, lines 5-6 denote the parameter update step performed by gradient descent using the Adam optimizer specific to each loss. Here, the term inside the $Adam$ operator is the (negative) gradient of the mean loss for the mini-batch of the proxy-source samples.

After the initialization, we begin the model upgrade. Note that, as mentioned in the paper, we use separate Adam optimizers for each loss and follow a round-robin alternating minimization scheme by optimizing a single loss using its specific optimizer. This allows each optimizer to adaptively scale the gradients of the corresponding loss, thus, avoiding the need to have separate hyperparameters for loss weighting. To this end, we define lists of corresponding losses (**Loss**), optimizers (**Opt**) and model parameters (Θ) (lines 8-10 in Algo. 4). Further, $iter$ (line 11) is a counter that is used to obtain the index of the loss to be optimized at each iteration ($cur = iter \bmod 5$).

At each iteration, we obtain the *guides* (line 15) for each class. For the losses $\mathcal{L}_{r1}, \mathcal{L}_{r2}$, we use the proxy-source samples (lines 16-18). The adaptation loss $\mathcal{L}_{a1}$ is applied on the pseudo-labeled target samples (lines 19-22), while the losses $\mathcal{L}_{a2}$ and $\mathcal{L}_c$ are applied on the confident target samples (23-25). For convenience, we depict the round-robin alternating loss optimization in line 26 where for the iteration $iter$, the index of the loss to be optimized is $cur = iter \bmod 5$. Thus, the corresponding loss is **Loss**[cur], which is optimized using the optimizer **Opt**[cur] thereby training the parameters $\Theta[cur]$. Note that the implementation of the parameter update step (line 26) is similar to that in lines 5-6, where the actual loss optimized is the mean loss calculated over the samples in the mini-batch under consideration. Finally, at the end of each epoch, we recalculate the Gaussian Prototypes $\mathcal{P}_t^c$ (line 29) and obtain confident target samples $\mathcal{B}_t^c$.

4 Pseudo-labeling Precision

We analyze the precision of the pseudo-labels obtained for the top- n percent confident target samples $\mathcal{B}_t^c$ for each class $c \in \mathcal{C}_t$. The precision of the target pseudo-labels for the task $\mathbf{A} \rightarrow \mathbf{D}$ on **Office** dataset is shown in Fig. 1 (in line with the setting in Table 3 in the paper). The overall precision is computed as the class-averaged precision of the pseudo-labels. Specifically, Fig. 1A shows the precision for samples predicted with a pseudo-label $k \in \mathcal{C}_t$ (ALL) and Fig. 1 shows the precision for samples predicted with a pseudo-label $k \in \mathcal{C}'_t$ (PRIV).

We observe that the pseudo-labeling precision for $n = 20\%$ target samples is high. This validates the efficacy of the pseudo-labeling process. Further, we also show the precision-curve obtained as the training proceeds. We find that the area

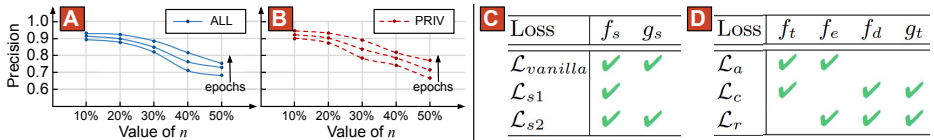


Fig. 1. **A)** Pseudo-labeling precision on $A \rightarrow D$ for ALL classes ($\mathcal{C}_t$) during adaptation. **B)** Pseudo-labeling precision on $A \rightarrow D$ for PRIV classes ($\mathcal{C}'_t$) during adaptation. **C)** Losses imposed during Foresighted source-model training. **D)** Losses imposed during CIDA on the target domain. ✓ indicates the modules trained by each loss.

under the curve increases as the training proceeds (increasing number of epochs is indicated by the arrow $\uparrow$). This demonstrates the reliability of pseudo-labels during adaptation. We find this trend in all our experiments, *i.e.* samples that are nearest to the *guides* have the highest pseudo-labeling precision. This validates our intuition for obtaining the set of confident samples $\mathcal{B}_t^c$.

5 Dataset details

The **Office** [11] dataset consists of 31 classes and has 3 domains: Amazon (A), DSLR (D) and Webcam (W). In our experiments, we use the first 10 classes in alphabetical order as the shared classes $\mathcal{C}_s$ and those ranked 21-31 alphabetically as target-private classes $\mathcal{C}'_t$. Since the DSLR and Webcam domains contain a relatively small number of images, we perform image augmentations as shown in Fig. 2 for each domain. The **Digits** dataset contains three domains MNIST (M) [6], SVHN (S) [8], USPS (U) [5]. We use the classes from 0 through 4 as the shared categories $\mathcal{C}_s$ whereas the classes from 5 through 9 are used as target-private categories $\mathcal{C}'_t$. The **VisDA** [9] dataset consists of 12 classes in total with Synthetic (Sy) and Real (Re) domains. The source domain consists of synthetic images that were created by rendering CAD models while the target domain consists of real-world images. Here, we set $\mathcal{C}_s = \{\text{bicycle, bus, car, motorcycle, train, truck}\}$, and set $\mathcal{C}'_t = \{\text{aeroplane, horse, knife, person, plant, skateboard}\}$.

6 Architecture details

In this section, we provide the details of the architectures used for **Office** [11], **VisDA** [9] and **Digits** datasets, along with the architectures of the baseline models presented in the paper (§4.2a). Additionally, in Fig. 1C-D, we show the modules that are trained with each loss imposed during the foresighted source-model training and during CIDA on the target domain.

6.1 Office and VisDA datasets

For training on the **Office** and **VisDA** datasets, we use ResNet-50 [4], pretrained on the **ImageNet** [10] dataset, as the CNN backbone for the feature extractor f_s . The rest of the network architecture is given in Table 2A.

6.2 Digits dataset

We use a modified version of LeNet [5] architecture (Table 2B) for training on **Digits** dataset, similar to that used in [12]. We use RGB images for training on each dataset (for MNIST and USPS, we repeat the pixel values across channels).

6.3 Illustrations for baselines

We illustrate the architectures (and gradient backpropagation pathways) for each baseline model in Fig. 3. In *Ours-a*, the clusters of the proxy-source samples in the $\mathcal{V}$ -space are frozen (blue clusters in Fig. 3A) and are the same as that obtained in the $\mathcal{U}$ -space at the end of the source-model training. In *Ours-b*, the loss $\mathcal{L}_r$ is not imposed as shown in Fig. 3B. In *Ours-c*, the loss $\mathcal{L}_{a2}$ is not enforced, as shown in Fig. 3C. In *Ours-d*, the source features $f_s(\mathbf{x}_s)$ are used in place of the Gaussian Prototypes as shown in Fig. 3D.

7 Theoretical Insights

In the paper, we argue that the adaptation process tightens the upper bound of target risk of the model, yielding a superior adaptation guarantee. In this section, we provide a formal discussion for this observation and demonstrate that our approach conforms with the theory proposed in [1]. In our approach, since we aim to learn a target-specific model to perform the model upgrade, we are interested in the dynamics of the target-specific latent space $\mathcal{V}$. Therefore, we begin by analyzing the properties of the distributions in the $\mathcal{V}$ -space.

7.1 Definitions

Let $q_{\mathcal{V}}$ denote the marginal distribution of target features in the $\mathcal{V}$ -space. Subsequently, we denote the marginal target-shared and target-private distributions by $q_{\mathcal{V}}^s$ and $q_{\mathcal{V}}^p$ respectively. Further, let the marginal distribution of the high-source-density region (effectively modeled by proxy-source samples) be $p_{\mathcal{V}}$, and the marginal distribution corresponding to the low-source-density region (effectively modeled by *negative* samples) be $r_{\mathcal{V}}$.

In our formulation, for a given sample, the label is predicted by taking the $\arg \max$ over the logits corresponding to $\mathcal{C}_s$ and $\mathcal{C}_t$. In other words, for a sample $\mathbf{v}$ in the $\mathcal{V}$ -space, we can define the label predictor (classifier) as follows,

$$g(\mathbf{v}) = \arg \max_{c \in \mathcal{C}_t} g_s \circ f_d(\mathbf{v})|_{c' \in \mathcal{C}_s} \parallel g_t(\mathbf{v}) \quad (1)$$

where $g \in \mathcal{H}$ is an instance of predictors pertaining to the hypothesis space $\mathcal{H}$ and $\parallel$ denotes concatenation. With this setup, we derive an upper bound on the target-domain risk of the predictor g and show that the bound is tightened by the adaptation process. To this end, we define two useful tools - the disagreement among predictors, and, the triangle inequality for classification error.

Definition 1 (Disagreement Measure). *The disagreement between two predictors g_1 and g_2 under a distribution d is given as,*

$$\epsilon_d(g_1, g_2) = \Pr_{\mathbf{v} \sim d} (g_1(\mathbf{v}) \neq g_2(\mathbf{v})) \quad (2)$$

Eq. 2 is a multi-class variation of the classifier disagreement proposed in [1]. We now extend Def. 1 to define the risk of a predictor.

Definition 2 (Risk Measure). *Let the ground-truth labeling function be h . The risk of a predictor g under a distribution d is defined as the probability with which the classifier g disagrees with the ground-truth labeling function h ,*

$$\epsilon_d(g, h) = \Pr_{\mathbf{v} \sim d} (g(\mathbf{v}) \neq h(\mathbf{v})) \quad (3)$$

For convenience, we shall use the short-hand notation to denote the risk of a predictor g as $\epsilon_d(g)$, i.e., $\epsilon_d(g) = \epsilon_d(g, h)$. Further, we use the disagreement measure to define the $\mathcal{H}\Delta\mathcal{H}$ -distance [1] between two distributions d_1 and d_2 .

Definition 3 ($\mathcal{H}\Delta\mathcal{H}$ -distance). *Given a hypothesis space $\mathcal{H}$ and two distributions d_1 and d_2 , the $\mathcal{H}\Delta\mathcal{H}$ -distance is given by*

$$l_{\mathcal{H}\Delta\mathcal{H}}(d_1, d_2) = 2 \sup_{g_1, g_2 \in \mathcal{H}} |\epsilon_{d_1}(g_1, g_2) - \epsilon_{d_2}(g_1, g_2)| \quad (4)$$

Here, $l_{\mathcal{H}\Delta\mathcal{H}}$ is a measure of the distribution shift between the distributions d_1 and d_2 . In other words, $l_{\mathcal{H}\Delta\mathcal{H}}$ grows as the discrepancy between the two distributions (domain-shift) grows. Note, in this section we use the notation $l_{\mathcal{H}\Delta\mathcal{H}}$ in place of $d_{\mathcal{H}\Delta\mathcal{H}}$ as done in [1], since here, d denotes a generic distribution. Now we describe the second tool, the triangle inequality for classification error [2].

Lemma 1 (Triangle Inequality). *Given three predictors $g_1, g_2, g_3 \in \mathcal{H}$, the triangle inequality for the disagreement measure is stated as*

$$\epsilon_d(g_1, g_3) \leq \epsilon_d(g_1, g_2) + \epsilon_d(g_2, g_3) \quad (5)$$

Finally, let us describe the performance of an ideal predictor. We define a predictor as ideal if it achieves the minimal combined risk over all distributions under consideration.

Definition 4 (Ideal Predictor). *Given a hypothesis space $\mathcal{H}$ and a set of n distributions $\{d_1, d_2, \dots, d_n\}$, an ideal predictor $g^* \in \mathcal{H}$ achieves the minimal combined risk on the given set of distributions. In other words,*

$$g^* = \arg \min_{g \in \mathcal{H}} \sum_{i=1}^n \epsilon_{d_i}(g) \quad (6)$$

Particularly, we consider the following distributions in the $\mathcal{V}$ -space: the source distribution $p_{\mathcal{V}}$, the *negative* distribution $r_{\mathcal{V}}$, the target-shared distribution $q_{\mathcal{V}}^s$ and the target-private distribution $q_{\mathcal{V}}^p$. Thus, the ideal predictor g^* is given as,

$$g^* = \arg \min_{g \in \mathcal{H}} \epsilon_{p_V}(g) + \epsilon_{r_V}(g) + \epsilon_{q_V^s}(g) + \epsilon_{q_V^p}(g) \quad (7)$$

Note that, given a hypothesis space $\mathcal{H}$, the combined risk of the ideal predictor g^* is a constant (*i.e.* $\lambda = \epsilon_{p_V}(g^*) + \epsilon_{r_V}(g^*) + \epsilon_{q_V^s}(g^*) + \epsilon_{q_V^p}(g^*)$).

7.2 A theoretical target risk bound

Using these definitions, we now derive a bound for the target-domain risk of the predictor g (Eq. 1). We do this by an extension of the theory in [1]. Recall that the target risk under consideration is $\epsilon_{q_V}(g)$. We state the theorem as follows.

Theorem 1 (Target risk bound). *Let $\mathcal{H}$ be a hypothesis space and let $g : \mathcal{V} \rightarrow \mathcal{Y}$ be a predictor such that $g \in \mathcal{H}$. Given the distributions p_V, r_V, q_V^s, q_V^p , the upper-bound for the target risk $\epsilon_{q_V}(g)$ is given as,*

$$\epsilon_{q_V}(g) \leq \epsilon_{p_V}(g) + \epsilon_{r_V}(g) + \frac{1}{2} \left\{ l_{\mathcal{H}\Delta\mathcal{H}}(q_V^s, p_V) + l_{\mathcal{H}\Delta\mathcal{H}}(q_V^p, r_V) \right\} + \lambda \quad (8)$$

where $\lambda = \epsilon_{p_V}(g^*) + \epsilon_{r_V}(g^*) + \epsilon_{q_V^s}(g^*) + \epsilon_{q_V^p}(g^*)$ is the combined risk of the ideal predictor $g^* \in \mathcal{H}$.

Proof. By definition, the target-shared and target-private samples correspond to two mutually exclusive sets, and hence, independently contribute to the target risk. Thus, we can decompose the risk as,

$$\underbrace{\epsilon_{q_V}(g)}_{\text{target-domain risk}} = \underbrace{\epsilon_{q_V^s}(g)}_{\text{target-shared risk}} + \underbrace{\epsilon_{q_V^p}(g)}_{\text{target-private risk}} \quad (9)$$

Consider the first term on the right hand side of Eq. 9, *i.e.* the target-shared risk $\epsilon_{q_V^s}$. We derive an upper bound for the target-shared risk as follows,

$$\begin{aligned} \epsilon_{q_V^s}(g) &= \epsilon_{q_V^s}(g, h) \\ &\leq \epsilon_{q_V^s}(g, g^*) + \epsilon_{q_V^s}(g^*, h) && \text{using Lemma 1} \\ &= \epsilon_{q_V^s}(g, g^*) - \epsilon_{p_V}(g, g^*) + \epsilon_{p_V}(g, g^*) + \epsilon_{q_V^s}(g^*, h) && \text{introducing } \epsilon_{p_V}(g, g^*) \\ &\leq |\epsilon_{q_V^s}(g, g^*) - \epsilon_{p_V}(g, g^*)| + \epsilon_{p_V}(g, g^*) + \epsilon_{q_V^s}(g^*, h) && \text{modulus operation} \\ &\leq \frac{1}{2} l_{\mathcal{H}\Delta\mathcal{H}}(q_V^s, p_V) + \epsilon_{p_V}(g, g^*) + \epsilon_{q_V^s}(g^*, h) && \text{using Definition 3} \\ &\leq \frac{1}{2} l_{\mathcal{H}\Delta\mathcal{H}}(q_V^s, p_V) + \epsilon_{p_V}(g, h) + \epsilon_{p_V}(g^*, h) + \epsilon_{q_V^s}(g^*, h) && \text{using Lemma 1} \end{aligned}$$

Using the short-hand notations for $\epsilon_{p_V}(g, h)$, $\epsilon_{p_V}(g^*, h)$ and $\epsilon_{q_V^s}(g^*, h)$ we get,

$$\epsilon_{q_V^s}(g) \leq \frac{1}{2} l_{\mathcal{H}\Delta\mathcal{H}}(q_V^s, p_V) + \epsilon_{p_V}(g) + \epsilon_{p_V}(g^*) + \epsilon_{q_V^s}(g^*) \quad (10)$$

In the derivation above, we have the following steps. First, we use the triangle inequality (Lemma 1) to obtain a relation among the disagreement between the

predictor g , the ideal predictor g^* and the ground-truth labeling function h . Then, we introduce the term $\epsilon_{p_V}(g, g^*)$ into the equation (zero-sum), and obtain an upper bound by the property of the modulus operation. At this stage, we analyze the modulus term on the RHS. Recall that the definition of $l_{\mathcal{H}\Delta\mathcal{H}}(d_1, d_2)$ between two distributions d_1, d_2 is the supremum of $2|\epsilon_{q_V^s}(g_1, g_2) - \epsilon_{p_V}(g_1, g_2)|$ over all $g_1, g_2 \in \mathcal{H}$. Since, the modulus term on RHS ($|\epsilon_{q_V^s}(g, g^*) - \epsilon_{p_V}(g, g^*)|$) is a special case (with g and g^*), it is upper bounded by $\frac{1}{2}l_{\mathcal{H}\Delta\mathcal{H}}(q_V^s, p_V)$. Finally, we use the triangle inequality to obtain an upper bound for the term $\epsilon_{p_V}(g, g^*)$. Thus, we obtain the inequality in Eq. 10.

In a similar manner, we can derive an upper bound for the second term on the RHS of Eq. 9, *i.e.* the target-private risk $\epsilon_{q_V^p}(g)$. Here, we shall consider $l_{\mathcal{H}\Delta\mathcal{H}}(q_V^p, r_V)$. Now, following the steps used to derive Eq. 10 we get,

$$\begin{aligned}
\epsilon_{q_V^p}(g) &= \epsilon_{q_V^p}(g, h) \\
&\leq \epsilon_{q_V^p}(g, g^*) + \epsilon_{q_V^p}(g^*, h) && \text{using Lemma 1} \\
&= \epsilon_{q_V^p}(g, g^*) - \epsilon_{r_V}(g, g^*) + \epsilon_{r_V}(g, g^*) + \epsilon_{q_V^p}(g^*, h) && \text{introducing } \epsilon_{r_V}(g, g^*) \\
&\leq |\epsilon_{q_V^p}(g, g^*) - \epsilon_{r_V}(g, g^*)| + \epsilon_{r_V}(g, g^*) + \epsilon_{q_V^p}(g^*, h) && \text{modulus operation} \\
&\leq \frac{1}{2}l_{\mathcal{H}\Delta\mathcal{H}}(q_V^p, r_V) + \epsilon_{r_V}(g, g^*) + \epsilon_{q_V^p}(g^*, h) && \text{using Definition 3} \\
&\leq \frac{1}{2}l_{\mathcal{H}\Delta\mathcal{H}}(q_V^p, r_V) + \epsilon_{r_V}(g, h) + \epsilon_{r_V}(g^*, h) + \epsilon_{q_V^p}(g^*, h) && \text{using Lemma 1}
\end{aligned}$$

Thus, the upper bound for target-private risk is,

$$\epsilon_{q_V^p}(g) \leq \frac{1}{2}l_{\mathcal{H}\Delta\mathcal{H}}(q_V^p, r_V) + \epsilon_{r_V}(g) + \epsilon_{r_V}(g^*) + \epsilon_{q_V^p}(g^*) \quad (11)$$

Finally, we combine the bounds for the target-shared risk (Eq. 10) and target-private risk (Eq. 11), and substitute them in Eq. 9,

$$\begin{aligned}
\epsilon_{q_V} &= \epsilon_{q_V^s} + \epsilon_{q_V^p} \\
&\leq \frac{1}{2}l_{\mathcal{H}\Delta\mathcal{H}}(q_V^s, p_V) + \epsilon_{p_V}(g) + \epsilon_{p_V}(g^*) + \epsilon_{q_V^s}(g^*) \\
&\quad + \frac{1}{2}l_{\mathcal{H}\Delta\mathcal{H}}(q_V^p, r_V) + \epsilon_{r_V}(g) + \epsilon_{r_V}(g^*) + \epsilon_{q_V^p}(g^*)
\end{aligned}$$

Rearranging the terms, we get the target risk bound stated in Theorem 1.

$$\begin{aligned}
\epsilon_{q_V} &\leq \underbrace{\epsilon_{p_V}(g) + \epsilon_{r_V}(g)}_{\text{augmented source domain risk}} + \frac{1}{2} \left\{ \underbrace{l_{\mathcal{H}\Delta\mathcal{H}}(q_V^s, p_V) + l_{\mathcal{H}\Delta\mathcal{H}}(q_V^p, r_V)}_{\text{distribution-shift between target and augmented source domains}} \right\} \\
&\quad + \underbrace{\epsilon_{p_V}(g^*) + \epsilon_{r_V}(g^*) + \epsilon_{q_V^s}(g^*) + \epsilon_{q_V^p}(g^*)}_{\lambda \text{ (minimal combined risk of the ideal predictor)}}
\end{aligned} \quad (12)$$

□

We have three major terms in Eq. 12 - the augmented source domain risk (source samples augmented with *negative* samples), distribution-shift between the target and the augmented source domains, and the minimal combined risk of the ideal predictor. We will now analyze the significance of this result.

7.3 Analysis

We now show that the target risk bound (Eq. 12) is tightened during adaptation.

Minimizing the distribution-shift. In the paper, we argue that the adaptation loss $\mathcal{L}_a$ minimizes the distribution-shift in the $\mathcal{V}$ -space between the target and the augmented source domain. Here, we describe the process. From the bound in Eq. 12, it is evident that the minimization of the distribution-shift entails the alignment of target-shared and the source distribution, and the alignment of target-private and *negative* sample distribution. Note that, the *negative* samples are essentially used to model the low-source-density region, while the proxy-source samples are used to model the high-source-density region.

In our approach, the loss $\mathcal{L}_{a2}$ pulls the target samples closer to the *guides*. As a result, the target-shared samples are aligned with high source-density region (proxy-source distribution). As we show in the paper in §4.2a, the proxy-source samples are effective in capturing the source-distribution. Therefore, $\mathcal{L}_{a2}$ contributes to the minimization of the term $l_{\mathcal{H}\Delta\mathcal{H}}(q_{\mathcal{V}}^s, p_{\mathcal{V}})$. Secondly, $\mathcal{L}_{a1}$ disperses the target clusters away, thereby separating target-private clusters into the low-source-density *negative* region in the $\mathcal{V}$ -space. Thus, the shift between the target-private and the low-source-density distribution $l_{\mathcal{H}\Delta\mathcal{H}}(q_{\mathcal{V}}^p, r_{\mathcal{V}})$ (modelled by the *negative* samples) is minimized. Overall, the adaptation process enforces the minimization of the distribution-shift between the target and the augmented source domains $l_{\mathcal{H}\Delta\mathcal{H}}(q_{\mathcal{V}}^s, p_{\mathcal{V}}) + l_{\mathcal{H}\Delta\mathcal{H}}(q_{\mathcal{V}}^p, r_{\mathcal{V}})$.

Mitigating Catastrophic Forgetting. We argue that $\mathcal{L}_r = \mathcal{L}_{r1} + \mathcal{L}_{r2}$ mitigates catastrophic forgetting, by preserving a low augmented source risk. Specifically, $\mathcal{L}_{r1}$ is the classification loss applied on the proxy-source samples. The supervision obtained from $\mathcal{L}_{r1}$ ensures that the discriminatory knowledge among the source samples is maintained. This preserves a low source risk $\epsilon_{p_{\mathcal{V}}}(g)$.

Further, to understand how $\epsilon_{r_{\mathcal{V}}}(g)$ is minimized, we must study the effect of compact decision boundaries. Intuitively, during the source training (§3.1b of the paper), the $(|\mathcal{C}_s| + 1)^{\text{th}}$ logit of g_s is trained to have the highest activation for the *negative* samples, while the source samples are subject to a lower *negative* class activation. This conditioning ensures that in the $\mathcal{U}$ -space, whenever a sample falls in the high-source-density region (Fig. 2A of the paper), the highest class logit corresponds to $\mathcal{C}_s$, and vice-versa (if a source class logit is the highest, the sample should pertain to the high-source-density region). Similarly, whenever the source class logits have a lower activation than the *negative* class logit, the sample pertains to the *negative* regime. This property is achieved due to the compact decision boundaries (Fig. 2A of the paper).

During adaptation, by applying the loss $\mathcal{L}_{r1}$ on the proxy-source samples we enforce the logits corresponding to $\mathcal{C}_s$ to have a higher activation than those

corresponding to $\mathcal{C}_t$. Indeed, this ensures that the proxy-source samples fall into the high-source-confidence regions when projected back to the $\mathcal{U}$ -space using f_d (see Fig. 2B in the paper). Additionally, $\mathcal{L}_{r2}$ avoids degenerate solutions (such as mode-collapse at the projected $\mathcal{U}$ -space), which maintains the property that *negative* samples would be projected to a low-source-density region by f_d . Thus, the *negative* samples receive a low source class confidence (alternatively, a higher target-private class confidence from g) which is the criterion to detect *negative* samples. Thus, the *negative* sample risk ($\epsilon_{r_V}(g)$) is minimized. To summarize, the augmented source domain risk $\epsilon_{p_V}(g) + \epsilon_{r_V}(g)$ is minimized by $\mathcal{L}_r$.

Performance of the ideal predictor. Finally, we also claim that the precision of pseudo-labels for the confident target samples is high. This is verified in Sec. 4. Thus, training the model with $\mathcal{L}_c$ along with $\mathcal{L}_{r1}$ achieves a performance close to the combined risk λ of the ideal predictor (referred to as the minimal risk of the optimal joint classifier). This is because, by virtue of $\mathcal{L}_c$ the model is conditioned to minimize the risk on q_V^s and q_V^p . Further, as argued above, the loss $\mathcal{L}_{r1}$ drives the model to minimize the risk on p_V and r_V . Note that, if we had enough labeled samples from each of the four distributions, we could learn the ideal predictor g^* . Nevertheless, using the pseudo-labeled target samples in addition to the proxy-source samples, we can learn a predictor whose performance is close to g^* .

The observations made above provide a basis for our intuition that the adaptation process tightens the target risk bound (Eq. 12). This enables a superior model upgrade under the CIDA paradigm.

8 Discussion on multi-step model upgrade

In this section, we provide details of the multi-step model upgrade experiment. Further, we highlight certain key features of our approach that makes our approach useful in a lifelong learning scenario.

Approach. In our approach, a foresighted source-model is trained with the aim of reducing domain and category bias incurred in the given source training dataset. This allows for a source-free model upgrade. We achieve this by modelling Gaussian Prototypes that allows us to enforce the *class separability* objective and to perform *negative* training using samples that fall outside the $3\text{-}\sigma$ confidence region of each class-specific Gaussian Prototype.

We adopt the same strategy during the model upgrade to prepare the target-specific model for a potential future upgrade (*i.e.* a second class-incremental upgrade). Specifically, we add an additional *negative* training loss (similar to $\mathcal{L}_{s2}$) during the first model-upgrade.

Implementation. In §4.2f of the paper, we demonstrate CIDA of the (A+D)-specific model to the **W** domain. During the first upgrade, the feature extractor f_t is adapted to the domain **D**. Simultaneously, we introduce a *negative*-class ($|\mathcal{C}'_t| + 1$)th to the classifier g_t which is trained to detect *negative* samples in the $\mathcal{V}$ -space. These *negative* samples are obtained using the target Gaussian Prototypes

by following a sampling strategy similar to that done in the foresighted source-model training on the domain $\mathbf{A}$ (i.e. we choose samples beyond the $3\text{-}\sigma$ confidence interval of each class-specific target Gaussian Prototype $\mathcal{P}_t^c$).

The model being used for the target domain $\mathbf{D}$ is $\{f_t, g_t, f_d, g_s\}$. For the second model upgrade, we consider a new domain $\mathbf{W}$. In order to adapt to $\mathbf{W}$, we initialize the parameters of the $\mathbf{W}$ -specific feature extractor f_{t_2} with those of f_t . For convenience, we denote the $\mathbf{W}$ -specific semantic space as $\mathcal{V}_2$. The domain projection networks $f_{e_2} : \mathcal{V} \rightarrow \mathcal{V}_2$ and $f_{d_2} : \mathcal{V}_2 \rightarrow \mathcal{V}$ are pre-trained to a near-identity function. Likewise, we introduce a $\mathbf{W}$ -specific target-private classifier g_{t_2} which is applied on the $\mathcal{V}_2$ -space. Further, the final predictor for $\mathbf{W}$ is the concatenation of the logits produced by the predictor for $\mathbf{D}$ and g_{t_2} , i.e., $\{\{g_s \circ f_d\} || f_t\}_{c \in \mathcal{C}_t}$ acting on the $\mathcal{V}$ -space and g_t acting on the $\mathcal{V}_2$ -space.

Suitability for lifelong learning. One of the key aspects of our proposed framework is the seamless deployment of the target model to multiple steps of CIDA. This is attributed to the following:

- Our approach to tackle CIDA is *source-free* in nature, and does not require the presence of the past source training datasets.
- The adaptation step ensures that the model is ready for a future upgrade on a new domain, without having to be retrained with the previously seen data.
- The domain-specific representation ($\mathcal{V}$ -space) allows for a better trade-off between domain-specificity and domain-generalization, where the target-specific model achieves a semantic granularity suitable for the target domain under consideration.
- The domain projection networks $\{f_e, f_d\}$ effectively establish a transit mechanism between two semantically diverse latent spaces, allowing the model to exploit the knowledge of all the previously seen classes.
- The efficacy of the pseudo-labeling process and the selection of confident target samples $\mathcal{B}_t^c$ enhances the reliability of the adaptation process.

We believe that our framework will provide new insights for future studies on lifelong learning in the presence of new domains and tasks.

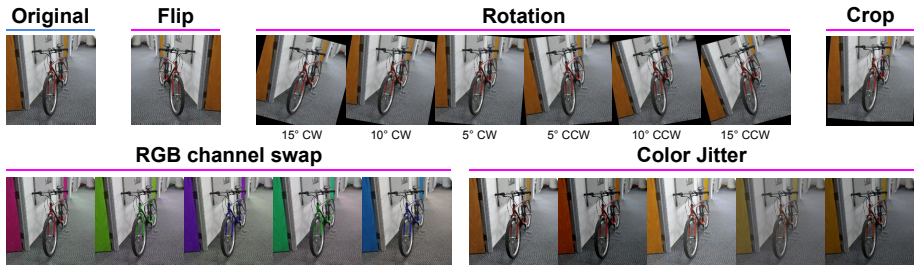


Fig. 2. Augmentations applied to the images in Office dataset

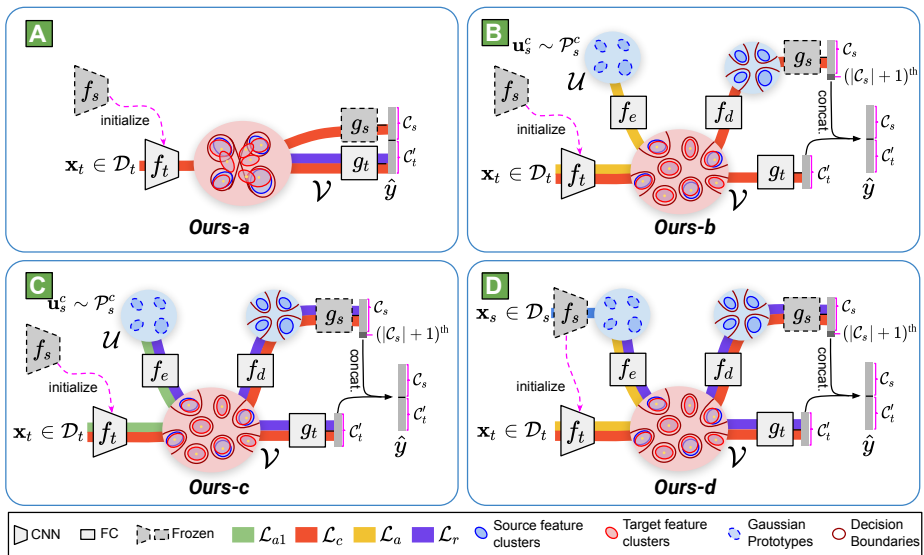


Fig. 3. Illustrations of the various baselines of our approach. See §4.2a in the paper for the discussion.

Algorithm 1 GetNegativeSamples

```

1: require: Global Gaussian Prototype  $\mathcal{P}_s$ , Class-specific Gaussian Prototypes  $\{\mathcal{P}_s^c : c \in \mathcal{C}_s\}$ , number of negative samples  $N_{neg}$  /* || denotes append */

2: initialize:  $\mathbf{U}_n \leftarrow \{\}, \mathbf{Y}_n \leftarrow \{\}$ 

3: for  $c$  in  $\mathcal{C}_s$  do // Obtain 3- $\sigma$  confidence interval thresholds for each class
4:    $\lambda_s^c, \boldsymbol{\nu}_s^c \leftarrow$  largest eigen value and the corresponding eigen vector of  $\boldsymbol{\Sigma}_s^c$ 
5:    $T_s^c \leftarrow \mathcal{P}_s^c(\boldsymbol{\mu}_s^c + 3 \cdot \sqrt{\lambda_s^c} \cdot \boldsymbol{\nu}_s^c)$ 
6: end for

7: while  $|\mathbf{U}_n| \leq N_{neg}$  do // Obtain negative samples
8:   Draw a sample  $\mathbf{u}_n \sim \mathcal{P}_s$ 
9:    $\hat{y}_n \leftarrow \arg \max_c \mathcal{P}_s^c(\mathbf{u}_n)$ 
10:  if  $\mathcal{P}_s^{\hat{y}_n}(\mathbf{u}_n) < T_s^{\hat{y}_n}$  then
11:     $\mathbf{U}_n \leftarrow \mathbf{U}_n \parallel \mathbf{u}_n$ 
12:     $\mathbf{Y}_n \leftarrow \mathbf{Y}_n \parallel \{\text{unknown}\}$ 
13:  end if
14: end while

15: return  $\{\mathbf{U}_n, \mathbf{Y}_n\}$ 

```

Algorithm 2 Foresighted Source-model training

```

1: require: Source samples  $\mathcal{D}_s$ , model parameters  $\theta_{f_s}, \theta_{g_s}$ , batch size of source samples
    $N_{src}$  and negative samples  $N_{neg}$  // || denotes append

2: repeat // Pre-train the network for 1 epoch with cross-entropy loss
3:   Obtain a mini-batch of source samples  $S_s = \{(\mathbf{x}_s, y_s) \sim p\}$ 
4:    $\theta_{f_s} \leftarrow \theta_{f_s} + Adam_{f_s} \left( -\nabla_{\theta_{f_s}} \frac{1}{|S_s|} \sum_{(\mathbf{x}_s, y_s) \in S_s} l_{ce}(g_s \circ f_s(\mathbf{x}_s), y_s) \right)$ 
5: until reached the end of 1 epoch

   // Calculate Gaussian Prototypes
6: for  $c \in \mathcal{C}_s$  do
7:    $\mathcal{D}_s^c \leftarrow \{(\mathbf{x}_s, y_s) : (\mathbf{x}_s, y_s) \in \mathcal{D}_s \text{ and } y_s = c\}$  // subset of samples in class  $c$ 
8:    $\mu_s^c \leftarrow \text{mean}_{(\mathbf{x}_s, y_s) \in \mathcal{D}_s^c} (f_s(\mathbf{x}_s))$  // mean vector
9:    $\Sigma_s^c \leftarrow \text{cov}_{(\mathbf{x}_s, y_s) \in \mathcal{D}_s^c} (f_s(\mathbf{x}_s))$  // covariance matrix
10:   $\mathcal{P}_s^c \leftarrow \mathcal{N}(\mu_s^c, \Sigma_s^c)$  // class-specific Gaussian Prototypes
11: end for
12:  $\mu_s \leftarrow \text{mean}_{(\mathbf{x}_s, y_s) \in \mathcal{D}} (f_s(\mathbf{x}_s))$  // mean vector
13:  $\Sigma_s \leftarrow \text{cov}_{(\mathbf{x}_s, y_s) \in \mathcal{D}} (f_s(\mathbf{x}_s))$  // covariance matrix
14:  $\mathcal{P}_s \leftarrow \mathcal{N}(\mu_s, \Sigma_s)$  // global Gaussian Prototype

   // Define list of losses and optimizers for alternating minimization
15: Loss  $\leftarrow [\mathcal{L}_{s1}, \mathcal{L}_{s2}]$ 
16: Opt  $\leftarrow [Adam_{\{f_s\}}, Adam_{\{f_s, g_s\}}]$ 
17:  $\Theta \leftarrow [\{\theta_{f_s}\}, \{\theta_{f_s}, \theta_{g_s}\}]$ 
18:  $iter \leftarrow 0$ 

19: repeat // Training loop for source-model
20:    $iter \leftarrow iter + 1$ 
21:    $cur \leftarrow iter \bmod 2$ 

   // Get a mini-batch of source and negative samples
22:    $S_s \leftarrow \{(\mathbf{x}_s, y_s) \sim p\}$  //  $|S_s| = N_{src}$ 
23:    $S_n \leftarrow \text{GetNegativeSamples}(\mathcal{P}_s, \{\mathcal{P}_s^c : c \in \mathcal{C}_s\}, N_{neg})$  //  $|S_n| = N_{neg}$ 
24:   for  $(\mathbf{x}_s, y_s) \in S_s$  do
25:      $\mathbf{y}_s \leftarrow g_s \circ f_s(\mathbf{x}_s)$ 
26:   end for
27:   for  $(\mathbf{u}_n, y_n) \in S_n$  do
28:      $\mathbf{y}_n \leftarrow g_s(\mathbf{u}_n)$ 
29:   end for

   // Optimize the loss Loss[ $cur$ ] using the optimizer Opt[ $cur$ ]
30:    $\Theta[cur] \leftarrow \Theta[cur] + \text{Opt}[cur] \left( -\nabla_{\Theta[cur]} \text{Loss}[cur] \right)$ 

31:   if reached the end of an epoch then
32:     Recalculate Gaussian Prototypes following lines 6-14
33:   end if
34: until convergence

```

Algorithm 3 GetConfidentSamples

```

1: require: Target samples  $\mathcal{D}_t$ , guides  $\{\mathbf{v}_g^c : c \in \mathcal{C}_t\}$ , percentage of confident samples
    $n$ , model parameters  $\theta_{f_t}$  //  $\parallel$  denotes append,  $[:]$  denotes indexing

2: initialize:  $\mathcal{B}_t^c \leftarrow \{\}$  //  $\forall c \in \mathcal{C}_t$ 

3: for  $\mathbf{x}_t \in \mathcal{D}_t$  do
4:    $\mathbf{v}_t \leftarrow f_t(\mathbf{x}_t)$ 
5:    $d \leftarrow \min_{c \in \mathcal{C}_t} l_2(\mathbf{v}_t, \mathbf{v}_g^c)$  // distance to the nearest guide
6:    $k \leftarrow \arg \min_{c \in \mathcal{C}_t} l_2(\mathbf{v}_t, \mathbf{v}_g^c)$  // pseudo-label for the target sample
7: end for

8: for  $c \in \mathcal{C}_t$  do
9:    $T \leftarrow$  Target samples with  $k = c$  sorted in asc. order of  $d$ 
10:   $b = \frac{n}{100} \cdot |T|$  // number of confident samples for class  $c$ 
11:   $\mathcal{B}_t^c \leftarrow T[0 : b]$  // the nearest  $b$  samples are the confident target samples for  $\mathbf{v}_g^c$ 
12: end for

13: return  $\{\mathcal{B}_t^c : c \in \mathcal{C}_t\}$ 

```

Algorithm 4 CIDA on the Target Domain

```

1: require: Target samples  $\mathcal{D}_t$ , Gaussian Prototypes  $\mathcal{P}_s^c$ , parameters
    $\theta_{f_s}, \theta_{g_s}, \theta_{f_e}, \theta_{f_d}, \theta_{f_t}, \theta_{g_t}$ , percentage of confident samples  $n$  //  $\|\$  denotes append

2:  $\theta_{f_t} \leftarrow \theta_{f_s}$  // Initialize  $f_t$  from  $f_s$ 
3: repeat // Initialize  $f_e, f_d$  with near-identity functions
4:   Obtain a mini-batch of proxy-source samples  $S = \{\mathbf{u}_s^c \sim \mathcal{P}_s^c : c \in \mathcal{C}_s\}$ 
5:    $\theta_{f_e} \leftarrow \theta_{f_e} + \text{Adam}_{f_e} \left( -\nabla_{\theta_{f_e}} \frac{1}{|S|} \sum_{\mathbf{u}_s^c \in S} l_2(\mathbf{u}_s, f_e(\mathbf{u}_s))^2 \right)$ 
6:    $\theta_{f_d} \leftarrow \theta_{f_d} + \text{Adam}_{f_d} \left( -\nabla_{\theta_{f_d}} \frac{1}{|S|} \sum_{\mathbf{u}_s^c \in S} l_2(\mathbf{u}_s, f_d(\mathbf{u}_s))^2 \right)$ 
7: until convergence

   // Define list of losses and optimizers for round-robin alternating minimization
8: Loss  $\leftarrow [\mathcal{L}_{r1}, \mathcal{L}_{r2}, \mathcal{L}_c, \mathcal{L}_{a1}, \mathcal{L}_{a2}]$ 
9: Opt  $\leftarrow [\text{Adam}_{\{f_e, f_d, g_t\}}, \text{Adam}_{\{f_e, f_d\}}, \text{Adam}_{\{f_t, g_t\}}, \text{Adam}_{\{f_t\}}, \text{Adam}_{\{f_e, f_t\}}]$ 
10:  $\Theta$   $\leftarrow [\{\theta_{f_e}, \theta_{f_d}, \theta_{g_t}\}, \{\theta_{f_e}, \theta_{f_d}\}, \{\theta_{f_t}, \theta_{g_t}\}, \{\theta_{f_t}\}, \{\theta_{f_e}, \theta_{f_t}\}]$ 
11: iter  $\leftarrow 0$ 

12: repeat // Training loop for CIDA
13:   iter  $\leftarrow \text{iter} + 1$ 
14:   cur  $\leftarrow \text{iter} \bmod 5$ 

15:    $\mathbf{v}_g^c \leftarrow f_e(\mu_s^c) \forall c \in \mathcal{C}_s, \mathbf{v}_g^c \leftarrow f_t(\tilde{\mathbf{x}}_t^c) \forall c \in \mathcal{C}_t'$  // Calculate guides

16:   for  $\mathbf{u}_s^c \in \{\mathbf{u}_s^c \sim \mathcal{P}_s^c : c \in \mathcal{C}_s\}$  do // Mini-batch of proxy-source samples
17:      $\mathbf{v}_s^c \leftarrow f_e(\mathbf{u}_s^c); \hat{\mathbf{u}}_s^c \leftarrow f_d(\mathbf{v}_s^c); \hat{\mathbf{y}}_s \leftarrow g_s(\hat{\mathbf{u}}_s^c)|_{c \in \mathcal{C}_s} \parallel g_t(\mathbf{v}_s^c)$ 
18:   end for

19:   for  $\mathbf{x}_t \in \{\mathbf{x}_t \sim q_{\mathcal{X}}\}$  do // Mini-batch of target samples
20:      $\mathbf{v}_t \leftarrow f_t(\mathbf{x}_t); \mathbf{u}_t \leftarrow f_d(\mathbf{v}_t); \hat{\mathbf{y}}_t \leftarrow g_s(\mathbf{u}_t)|_{c \in \mathcal{C}_s} \parallel g_t(\mathbf{v}_t)$ 
21:      $d \leftarrow \min_{c \in \mathcal{C}_t} l_2(\mathbf{v}_t, \mathbf{v}_g^c); k \leftarrow \arg \min_{c \in \mathcal{C}_t} l_2(\mathbf{v}_t, \mathbf{v}_g^c)$  // Pseudo-label
22:   end for

23:   for  $\mathbf{x}_t^c \in \{\mathbf{x}_t^c \sim \mathcal{B}_t^c : c \in \mathcal{C}_t\}$  do // Mini-batch of confident target samples
24:      $\mathbf{v}_t^c \leftarrow f_t(\mathbf{x}_t^c), \mathbf{u}_t^c \leftarrow f_d(\mathbf{v}_t^c), \mathbf{y}_t^c \leftarrow g_s(\mathbf{u}_t^c)|_{c \in \mathcal{C}_t} \parallel g_t(\mathbf{v}_t^c)$ 
25:   end for

   // Optimize the loss Loss[cur] using the optimizer Opt[cur]
26:    $\Theta[\text{cur}] \leftarrow \Theta[\text{cur}] + \text{Opt}[\text{cur}] \left( -\nabla_{\Theta[\text{cur}]} \text{Loss}[\text{cur}] \right)$ 

27:   if reached the end of an epoch then
28:     Pseudo-label all samples in  $\mathcal{D}_t$  using guides  $\{\mathbf{v}_g^c : c \in \mathcal{C}_t\}$ 
29:      $\mathcal{P}_t^c \leftarrow$  Gaussian Prototypes obtained using pseudo-labeled target samples
30:      $\mathcal{B}_t^c \leftarrow \text{GetConfidentSamples}(\mathcal{D}_t, \{\mathbf{v}_g^c : c \in \mathcal{C}_t\}, n)$ 
31:   end if
32: until convergence

```

Table 2. The architecture used for training on **Office**, **VisDA** and **Digits** datasets. FC denotes a Fully Connected layer. BN denotes a BatchNorm layer. Conv is Convolutional layer. **Act.** denotes the activation applied at the output of the module.

	A. Office and VisDA			B. Digits		
	Module	Features	Act.	Module	Features	Act.
f_s	ResNet-50 (till avg pool)	2048	-	Conv	$28 \times 28 \times 64$	LeakyReLU
	FC	1024	ELU	Conv	$24 \times 24 \times 64$	LeakyReLU
	BN	-	-	Conv	$11 \times 11 \times 128$	LeakyReLU
	FC	256	ELU	Conv	$5 \times 5 \times 128$	LeakyReLU
	FC	256	ELU	Flatten	3200	-
	BN	-	-	FC	100	LeakyReLU
g_s	Input	256	-	Input	100	-
	FC	64	ELU	FC	32	LeakyReLU
	FC	$ \mathcal{C}_s + 1$	-	FC	$ \mathcal{C}_s + 1$	-
f_t	same as f_s			same as f_s		
f_e	Input	256	-	Input	100	-
	FC	256	ELU	FC	100	LeakyReLU
	FC	256	ELU	FC	100	LeakyReLU
	FC	256	ELU	-	-	-
f_d	same as f_e			same as f_e		
g_t	Input	256	-	Input	100	-
	FC	64	ELU	FC	32	LeakyReLU
	FC	$ \mathcal{C}'_t $	-	FC	$ \mathcal{C}'_t $	-

References

1. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. *Machine learning* **79**(1-2), 151–175 (2010) [6](#), [7](#), [8](#)
2. Crammer, K., Kearns, M., Wortman, J.: Learning from multiple sources. *Journal of Machine Learning Research* **9**(Aug), 1757–1774 (2008) [7](#)
3. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. *JMLR* **17**(1), 2096–2030 (2016) [1](#)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *CVPR* (2016) [5](#)
5. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11), 2278–2324 (1998) [5](#), [6](#)
6. LeCun, Y., Cortes, C., Burges, C.J.C.: The mnist database of handwritten digits <http://yann.lecun.com/exdb/mnist/> [5](#)
7. Liu, H., Cao, Z., Long, M., Wang, J., Yang, Q.: Separate to adapt: Open set domain adaptation via progressive separation. In: *CVPR* (2019) [1](#)
8. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: *Deep Learning and Unsupervised Feature Learning Workshop at NeurIPS* (2011) [5](#)
9. Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., Saenko, K.: Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924* (2017) [5](#)
10. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. *IJCV* (2015) [5](#)
11. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: *ECCV* (2010) [5](#)
12. Saito, K., Yamamoto, S., Ushiku, Y., Harada, T.: Open set domain adaptation by backpropagation. In: *ECCV* (2018) [1](#), [6](#)
13. You, K., Long, M., Cao, Z., Wang, J., Jordan, M.I.: Universal domain adaptation. In: *CVPR* (2019) [1](#)