

# Supplementary: Universal Source-Free Domain Adaptation

This Supplementary is organized as follows,

- Sec. 1: Notations
- Sec. 2: Implementation details
  - *Procurement* Stage. (Sec. 2.1, Algo. 1)
  - *Deployment* Stage. (Sec. 2.2)
- Sec. 3: Additional Results
  - Pretraining the backbone network on Places instead of ImageNet. (Sec. 3.1, Table 2)
  - Space and Time complexity analysis. (Sec. 3.2)
  - Varying label-set relationship. (Sec. 3.3, Fig. 1)
  - Sensitivity analysis. (Sec. 3.4, Fig. 2)
  - Closed-set adaptation. (Sec. 3.5, Table 3)
  - Accuracy on source dataset after *Procurement*. (Sec. 3.6)
  - Incremental one-shot classification. (Sec. 3.7)
  - Feature space visualization. (Sec. 3.8, Fig. 3)
- Sec. 4: Miscellaneous
  - Specifications of computing resources. (Sec. 4.1)
  - References to code. (Sec. 4.2)

## 1. Notations

We summarize the notations used in the paper in Table. 1.

## 2. Implementation Details

Here, we describe the network architectures and the training process used for the *Procurement* and the *Deployment* stages of our approach.

### 2.1. Procurement Stage

**a) Design of the classifier  $D$ .** Keeping in mind the possibility of the model encountering an additional domain shift after having adapted from the source domain to a target domain (e.g. encountering domain  $\mathbf{W}$  after performing the adaptation  $\mathbf{A} \rightarrow \mathbf{D}$  in **Office-31** dataset), we design the classifier’s architecture in a manner which allows for modification in the number of negative classes post *Procurement*.

Table 1: Notation Table

|                       | Symbol                       | Description   |
|-----------------------|------------------------------|---|
| Distribution          | $p$                          | Marginal source input distribution                      |
|                       | $p_n$                        | Marginal negative feature distribution                  |
|                       | $q$                          | Marginal target input distribution                      |
|                       | $p_{\bar{s}}$                | Marginal source-private distribution                    |
|                       | $q_{\bar{t}}$                | Marginal target-private distribution                    |
|                       | $P(u_s c_i)$                 | Gaussian prior for the source samples                   |
| Network               | $M$                          | Backbone model  |
|                       | $F_s$                        | Source feature extractor                                |
|                       | $F_t$                        | Target feature extractor                                |
|                       | $G$                          | Decoder   |
|                       | $D$                          | Classifier  |
| Sets                  | $\mathcal{D}_s$              | Labeled source dataset                                  |
|                       | $\mathcal{D}_n$              | Labeled negative dataset                                |
|                       | $\mathcal{D}_t$              | Unlabelled target dataset                               |
|                       | $\mathcal{C}_s$              | Label-set of the source domain                          |
|                       | $\mathcal{C}_n$              | Label-set of the negative samples                       |
|                       | $\mathcal{C}_t$              | Label-set of the target domain                          |
|                       | $\mathcal{C}$                | Shared label-set  |
|                       | $\bar{\mathcal{C}}_s$        | Source-private label-set                                |
|                       | $\bar{\mathcal{C}}_t$        | Target-private label-set                                |
| Samples / Misc.       | $(x_s, y_s)$                 | Paired source samples                                   |
|                       | $(x_n, y_n)$                 | Paired negative samples                                 |
|                       | $x_t$                        | Unpaired target samples                                 |
|                       | $v_s, v_t$                   | Output of $M$ for source / target resp.                 |
|                       | $\hat{v}_s$                  | Output of $G$   |
|                       | $u_s, u_t$                   | Output of $F_s / F_t$ resp.                             |
|                       | $u_r$                        | Samples drawn from class priors                         |
|                       | $\mu_{c_i}$                  | Mean feature $u_s$ for $c_i \in \mathcal{C}_s$          |
|                       | $\Sigma_{c_i}$               | Covariance of $u_s$ for $c_i \in \mathcal{C}_s$         |
|                       | $d$                          | Output of $D$ (logit vector)                            |
|                       | $\sigma^{(k)}(\cdot)$        | $k^{\text{th}}$ element of the softmax vector           |
|                       | $\hat{z}$                    | Softmax over $ \mathcal{C}_s  +  \mathcal{C}_n $ logits |
|                       | $\tilde{z}_s$                | Softmax over $ \mathcal{C}_s $ logits                   |
|                       | $\tilde{z}_n$                | Softmax over $ \mathcal{C}_n $ logits                   |
| $w(\cdot), w'(\cdot)$ | SSM and its complement resp. |   |

---

**Algorithm 1** Negative dataset generation using Image-composition

---

- 1: **input:** Image pair  $(I_1, I_2) \in \mathcal{D}_s$ . (image shape  $H \times W \times 3 = 224 \times 224 \times 3$ )
- 2:  $k \leftarrow 30$
- 3:  $x_1, x_2, y_1, y_2 \leftarrow \text{rand}(0, W), \text{rand}(0, W), \text{rand}(0, H), \text{rand}(0, H)$
- 4:  $c_x, c_y \leftarrow \text{rand}(W/2 - k, W/2 + k), \text{rand}(H/2 - k/3, H/2 + k/3)$
- 5:  $d_x, d_y \leftarrow \text{rand}(W/2 - k/3, W/2 + k/3), \text{rand}(H/2 - k, H/2 + k)$

**Horizontal Splicing**

- 6:  $s_1 \leftarrow \text{quadratic\_interpolation}([(0, y_1), (c_x, c_y), (223, y_2)])$

**Vertical Splicing**

- 7:  $s_2 \leftarrow \text{quadratic\_interpolation}([(x_1, 0), (d_x, d_y), (x_2, 223)])$

**Defining Masks**

- 8:  $m_1 \leftarrow$  mask region below  $s_1$
- 9:  $m_2 \leftarrow$  mask region to the left of  $s_2$

**Merging alternate regions to create composite images**

- 10:  $I_a \leftarrow m_1 * I_1 + (1 - m_1) * I_2$
- 11:  $I_b \leftarrow m_2 * I_1 + (1 - m_2) * I_2$
- 12:  $I_c \leftarrow m_1 * I_2 + (1 - m_1) * I_1$
- 13:  $I_d \leftarrow m_2 * I_2 + (1 - m_2) * I_1$

- 14: **return**  $I_a, I_b, I_c, I_d$
- 

We achieve this by maintaining two separate classifiers during the *Procurement* stage -  $D_{src}$  that operates on the positive source classes, and,  $D_{neg}$  that operates on the negative source classes (see architecture in Table 5). The final classification score is obtained by computing softmax over the concatenation of logit vectors produced by  $D_{src}$  and  $D_{neg}$ . Therefore, the model can be retrained on a different number of negative classes post *Deployment* (using another negative class classifier  $D_{neg}^2$ ), thus preparing it for a subsequent adaptation step to another domain.

**b) Negative dataset generation.** We generate the negative dataset  $\mathcal{D}_n$  by compositing images taken from different positive source classes, as described in Algo. 1. We generate random masks using quadratic splines passing through a central image region (lines 3-9). Using these masks, we merge alternate regions of the images, both horizontally and vertically, resulting in 4 negative images for each pair of images (lines 10-13). To effectively cover the inter-class negative region, we randomly sample image pairs from  $\mathcal{D}_s$  belonging to different classes, however we do not impose any constraint on how the classes are selected (for e.g. one can composite images from an animal and a non-animal class). We choose 5000 pairs for tasks on **Office-31**, **Office-Home** and **VisDA** datasets, and 12000 for **ImageNet-Caltech**. Since the input source distribution ( $p$ ) is fixed we first synthesize a negative dataset offline (instead of creating them on the fly) to ensure finiteness of the training set. The training algorithm for *USFDA* is given in Algo. 1 of the paper.

**c) Justification of  $\mathcal{L}_p$ .** The cross-entropy loss enforced on the likelihoods (referred to as  $\mathcal{L}_p$  in the paper) not only enforces intra-class compactness but also ensures inter-class separability in the latent  $u$ -space. Since the negative samples are only an approximation of the future target private classes that are expected to be encountered, we choose not to employ this loss for negative samples. Such a training procedure, eventually results in a natural development of bias towards the confident positive source classes. This subsequently leads to the placement of source clusters in a manner which enables source-free adaptation (See Fig. 4C of the paper).

**d) Minibatch negative sampling strategy.** We create an unbiased batch of training samples for a training iteration by sampling equal number of positive and negative samples from the dataset. Particularly, we sample 32 positive source class images ( $b_{+ve} = 32$ ) and 32 negative images ( $b_{-ve} = 32$ ) for each training iteration. This gives an effective batch size of  $b_{+ve} + b_{-ve} = 64$ .

**e) Use of multiple optimizers for training.** In the presence of multiple losses, we subvert a time-consuming loss-weighting hyperparameter search by making use of multiple Adam optimizers during training. Essentially, we define a separate optimizer for each loss term, and optimize only one of the losses (chosen in a round robin fashion) in each iteration of training. We use a learning rate of 0.0001 for each Adam optimizer. Intuitively, the moment parameters in each Adam optimizer adaptively scales the corresponding gradients, thereby avoiding loss-scaling hyperparameters.

**f) Label-Set Relationships.** For **Office-31** dataset in the UDA setting, we use the 10 classes shared by Office-31 [5] and Caltech-256 [1] as the shared label-set  $\mathcal{C}$ . These classes are: *back\_pack, calculator, keyboard, monitor, mouse, mug, bike, laptop\_computer, headphones, projector*. From the remaining classes, in alphabetical order, we choose the first 10 classes as source-private ( $\bar{\mathcal{C}}_s$ ) classes, and the rest 11 as target-private ( $\bar{\mathcal{C}}_t$ ) classes. For **VisDA**, alphabetically, the first 6 classes are considered as  $\mathcal{C}$ , the next 3 as  $\bar{\mathcal{C}}_s$  and the last 3 comprise  $\bar{\mathcal{C}}_t$ . The **Office-Home** dataset has 65 categories, of which we use the first 10 classes as  $\mathcal{C}$ , the next 5 for  $\bar{\mathcal{C}}_s$ , and the rest 50 classes as  $\bar{\mathcal{C}}_t$ .

## 2.2. Deployment Stage

**a) Architecture.** The network architecture used during the *Deployment* stage is given in Table 6. Note that the decoder  $G$  used during the *Procurement* stage, is not available during *Deployment*, restricting complete access to the source data.

**b) Training.** The only trainable component is the Feature Extractor  $F_t$ , which is initialized from  $F_s$ . Here, the  $SSM$  is calculated by passing the target images through the network trained on source data (source model), i.e for each image  $x_t$ , we calculate  $\hat{y} = \sigma(D \circ F_s \circ M(x_t))$ . Note that the softmax is calculated over all  $|\mathcal{C}_s| + |\mathcal{C}_n|$  classes. This is done by concatenating the outputs of  $D_{src}$  and  $D_{neg}$ , and then calculating softmax. Then, the  $SSM$  is determined by the exponential confidence of a target sample, where confidence is the highest softmax value in the categories in  $\mathcal{C}_s$ .

## 3. Additional Results

### 3.1. Pretraining the backbone network on Places instead of ImageNet.

We find that widely adopted standard domain adaptation datasets such as **Office-31** [5] and **VisDA** [4] often share a part or all of their label-set with **ImageNet**. Therefore, to validate our method’s applicability when initialized from a network pretrained on an unrelated dataset, we attempt to solve the adaptation task **A**→**D** in **Office-31** dataset by pretraining the ResNet-50 backbone on **Places** dataset [8]. In Table 2 it can be observed that our method outperforms even source-dependent methods (e.g. UAN [7], which is also initialized a ResNet-50 backbone pretrained on **Places**). In contrast to our method, the algorithm in UAN [7] involves ResNet-50 finetuning. Therefore, we also compare against a variant of UAN with a frozen backbone network, by inserting an additional feature extractor that operates on the features extracted from ResNet-50 (similar to  $F_s$  in the proposed method). The architecture of the feature extractor used for this variant of UAN is outlined in Table 4. We observe that our method significantly outperforms this variant of UAN with lesser number of trainable parameters (see Table 2).

Table 2: Evaluation of the proposed method on **A**→**D** task of **Office-31** [5] dataset, pretraining the ResNet-50 backbone ( $M$ ) on Places instead of Imagenet. Note that, we set  $|\mathcal{C}|/|\mathcal{C}_s \cup \mathcal{C}_t| = 0.32$ , similar to the setting used in Table 2 of the paper. Additionally, the last two columns of the table show a comparison between our method and UAN [7] with regard to the number of trainable parameters and total training time for adaptation.

| Method       | ResNet-50 finetuning | Avg. per-class accuracy, $\mathcal{T}_{avg}$ | Number of Trainable Params. | Training time for Adaptation |
|--------------|----------------------|--|-----------------------------|------------------------------|
| UAN*         | ✓                    | 60.98  | 26.7 Million                | 280s                         |
| UAN*         | ✗                    | 52.48  | 5.6 Million                 | 125s                         |
| <b>USFDA</b> | ✗                    | <b>62.74</b>                                 | <b>3.5 Million</b>          | <b>44s</b>                   |

### 3.2. Space and Time complexity analysis.

On account of keeping the weights of the backbone network ( $M$ ) frozen throughout the training process, and devoid of networks such as adversarial discriminator our method makes use of significantly lesser trainable parameters when compared to previous methods such as UAN [7] (See Table 2). Bereft of adversarial training, the proposed method also has a significantly lesser total training time for adaptation: 44 sec versus 280 sec in UAN (for the **A**→**D** task of Office-31 dataset and batch size of 32). Thus, the proposed framework offers a much simpler adaptation pipeline, with a superior computational complexity while achieving state-of-the-art domain adaptation performance across different datasets, even without accessing labeled source data at the time of adaptation (See Table 2). This corroborates the superiority of our method in real-time deployment scenarios.

### 3.3. Varying label-set relationship

In addition to the  $\mathcal{T}_{avg}$  reported in Fig. 6 in the paper, we also compare the target-unknown accuracy  $\mathcal{T}_{unk}$  for UAN\* and USFDA. The results are presented in Fig. 1. Clearly, our method achieves a significant improvement over UAN on most settings. This demonstrates the capability of USFDA to detect outlier classes more efficiently, which can be attributed to the ingeniously developed *Procurement* stage.

### 3.4. Sensitivity Analysis

In all our experiments (across datasets as in Tables 1-2 of the paper and across varied label-set relationships as in Fig. 6 of the paper), we fix the hyperparameters as,  $\alpha = 0.2$ ,  $\beta = 0.1$ ,  $|\mathcal{C}_n| = |\mathcal{C}_s|C_2$  and  $b_{+ve}/b_{-ve} = 1$ . As mentioned in Sec. 4.3 of the paper, one can treat these hyperparameters as global constants. Nevertheless, in Fig. 2 we demonstrate the sensitivity of the model to these hyperparameters. Specifically, in Fig. 2A we show the sensitivity of the adaptation performance, to the choice of  $|\mathcal{C}_n|$  during the *Procurement* stage, across a spectrum of label-set relationships. In Fig. 2B we show the sensitivity of the model to  $\alpha$  and the batch-size

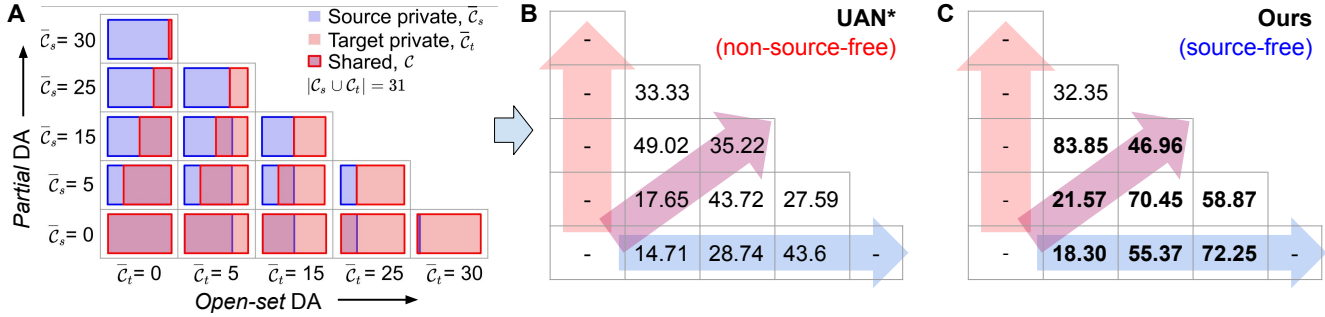


Figure 1: Comparison of  $\mathcal{T}_{unk}$  across varied label-set relationships for the task A→D in Office-31 dataset. **A)** Visual representation of label-set relationships and  $\mathcal{T}_{unk}$  at the corresponding instances for **B)** UAN\* [7] and **C)** ours *source-free* model. Effectively, the direction along x-axis (blue horizontal arrow) characterizes increasing *Open-set* complexity. The direction along y-axis (red vertical arrow) shows increasing complexity of *Partial* DA scenario. And the pink diagonal arrow denotes the effect of decreasing shared label space.

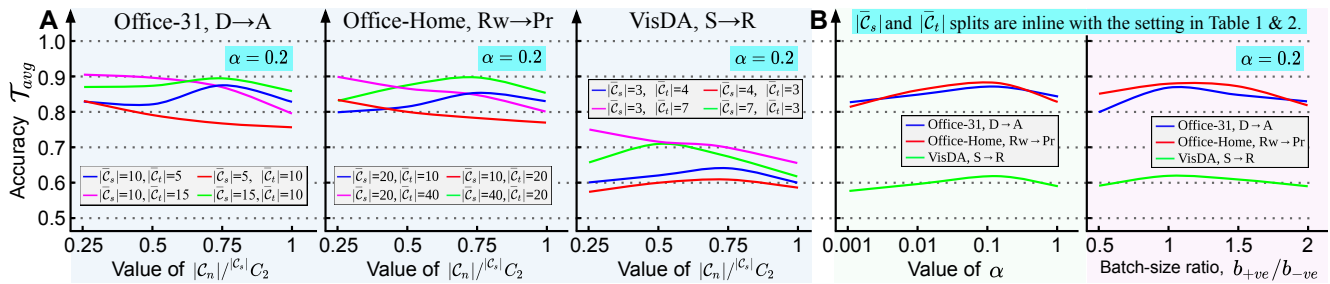


Figure 2: **A.** Sensitivity against  $|C_n|$ , represented by  $|C_n|/|C_s| C_2$  for varying  $|\bar{C}_s|$  or  $|\bar{C}_t|$  (see fig. legend) by fixing the others (top cyan box), across varied datasets. **B.** Sensitivity against  $\alpha$  and batch-size ratio (fixed  $b_{+ve} + b_{-ve} = 64$ ). Note the scale of X and Y-axis.

ratio  $b_{+ve}/b_{-ve}$  (ratio of positive vs. negative samples during *Procurement*). Sensitivity to  $\beta$  is shown in Fig. 5B of the paper. The model exhibits a reasonably low sensitivity to the hyperparameters, even in the challenging source-free scenario that allows for a reliable adaptation pipeline.

### 3.5. Closed-set adaptation

We additionally evaluate our method in the unsupervised source-free closed set adaptation scenario. In Table 3 we compare with the closed-set DA methods DAN [2], ADDA [6], CDAN [3] and the universal domain adaptation method UAN [7]. Note that, DAN, ADDA and CDAN rely on the assumption of a shared label space between the source and the target, and hence are not suited for a universal setting. Furthermore, all other methods require an explicit retraining on the source data during adaptation to perform well, even in the closed-set scenario. This clearly establishes the superiority of our method in the source-free setting.

### 3.6. Accuracy on source dataset after Procurement

We observe in our experiments that the accuracy on the source samples does not drop as a result of the partially generative framework. For the experiments conducted in Fig. 5C of the paper, we observe similar classification accuracy

on the source validation set, on increasing the number of negative classes from 0 to 190. This effect can be attributed to a carefully chosen  $\alpha = 0.2$ , which is deliberately biased towards positive source samples to help maintain the discriminative power of the model even in the presence of class imbalance (*i.e.*  $|C_n| \gg |C_s|$ ). This enhances the model's generative ability without compromising on the discriminative capacity on the positive source samples.

### 3.7. Incremental one-shot classification

In universal adaptation, we seek to transfer the knowledge of "class separability" obtained from the source domain to the deployed target environment. More concretely, it is attributed to the segregation of data samples based on an expected characteristics, such as classification of objects according to their pose, color, or shape etc. To quantify this, we consider an extreme case where  $C_s \cap C_t = \emptyset$  (**A→D** in **Office-31** with  $|C_s| = 15, |C_t| = 16$ ). Considering access to a single labeled target sample from each target category in  $\bar{C}_t = C_t$ , which are denoted as  $x_t^{c_j}$ , where  $j = 1, 2, \dots, |C_t|$ , we perform one-shot Nearest-Neighbour based classification by obtaining the predicted class label as  $\hat{c}_t = \operatorname{argmin}_{c_j} \|F_t \circ M(x_t) - F_t \circ M(x_t^{c_j})\|_2$ . Then, the classification accuracy for the entire target set is computed by comparing  $\hat{c}_t$  with

Table 3: Accuracy(%) on unsupervised closed-set DA (all use *ResNet50*). Ours is w/o hyperparameter tuning. Refer Sec. 3.5.

| Closed-set DA methods | source-free | Universal-DA | Office-31 |      |      |      |      |      |      | Avg. | VisDA<br>S → R |
|-----------------------|-------------|--------------|-----------|------|------|------|------|------|------|------|----------------|
|                       |             |              | D→A       | A→D  | A→W  | W→D  | W→A  | D→W  |      |      |                |
| DAN (ICML'15)         | ✗           | ✗            | 63.6      | 78.6 | 80.5 | 99.6 | 62.8 | 97.1 | 80.4 | 61.1 |                |
| ADDA (CVPR'17)        | ✗           | ✗            | 69.5      | 77.8 | 86.2 | 98.4 | 68.9 | 96.2 | 82.8 | -    |                |
| CDAN (NeurIPS'18)     | ✗           | ✗            | 70.1      | 89.8 | 93.1 | 100  | 68.0 | 98.2 | 86.5 | 66.8 |                |
| UAN (CVPR'19)         | ✗           | ✓            | 68.4      | 85.3 | 81.2 | 99.1 | 69.7 | 98.1 | 83.6 | -    |                |
| Ours <i>USFDA</i>     | ✓           | ✓            | 70.4      | 85.4 | 81.6 | 98.0 | 69.4 | 98.4 | 83.9 | 59.8 |                |

the corresponding ground-truth category. We obtain 64.72% accuracy for the proposed framework as compared to 13.43% for UAN\* [7]. A higher accuracy indicates that, the samples are inherently clustered in the intermediate feature level  $M \circ F_t(x_t)$  validating an efficient transfer of “class separability” in a fully unsupervised manner.

### 3.8. Feature space visualization

We obtain a t-SNE plot at the intermediate feature level  $u$  for both target and source samples (see Fig. 3), where the embedding for the target samples is obtained as  $u_t = F_t \circ M(x_t)$  and the same for the source samples is obtained as  $u_s = F_s \circ M(x_s)$ . This is because we aim to learn domain-specific features in contrast to domain-agnostic features as a result of the restriction imposed by the source-free scenario (“cannot disturb placement of source clusters”). Firstly we obtain compact clusters for the source-categories as a result of the partially generative *Procurement* stage. Secondly, the target-private clusters are placed away from the source-shared and source-private as expected as a result of the carefully formalized *SSM* weighting scheme in the *Deployment* stage. This plot clearly validates our hypothesis.

## 4. Miscellaneous

### 4.1. Specifications of computing resources

For both *Procurement* and *Deployment* stages, we make use of the machine with the specifications as follows. CPU: Intel core i7-7700K, RAM: 32 GB, GPU: NVIDIA GeForce GTX 1080Ti (11 GB). The model is trained in Python 2.7 with PyTorch 1.0.0, with CUDA v8.0.61.

### 4.2. References to code

Our complete documented code (including data loaders, training pipeline etc.) used for the experiments is available at <https://github.com/val-iisc/usfda>. For evaluating UAN [7], we execute the official implementation provided by the authors on github<sup>1</sup>.

<sup>1</sup>UAN [7]: <https://github.com/thuml/Universal-Domain-Adaptation>

Table 4: Architecture of the feature extractor used for UAN [7] under the “no ResNet-50 finetuning” case (see Table 2 and Sec. 3.1)

| Operation       | Features | Non-Linearity |
|-----------------|----------|---------------|
| Input           | 2048     |               |
| Fully connected | 512      | ReLU          |
| Fully connected | 256      | ReLU          |
| Fully connected | 512      | ReLU          |
| Fully connected | 2048     | ReLU          |

## References

- [1] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.
- [2] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.
- [3] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018.
- [4] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. In *CVPR workshops*, 2018.
- [5] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.
- [6] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.
- [7] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Universal domain adaptation. In *CVPR*, June 2019.
- [8] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

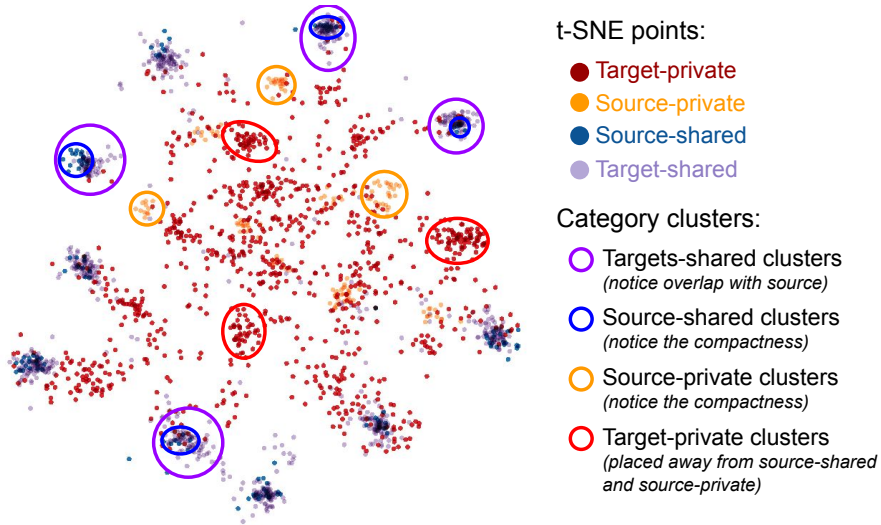


Figure 3: t-SNE plot showing the placement of all the four clusters computed after adaptation for the task A→D in Office-31. It validates our hypothesis in both the *Procurement* and the *Deployment* stages as shown by the highlighted clusters and the corresponding inferences in the legend under "Category clusters".

Table 5: Network architecture for the *Procurement* stage. Hyperparameter  $\alpha = 0.2$

| Component                                | Trainable? | Operation       | Notation  | Features                            | Batch Norm? | Non-Linearity |
|--|------------|-----------------|-----------|-------------------------------------|-------------|---------------|
| <b>Resnet-50</b><br>(Upto AvgPool layer) | ✗          |                 | $M$       | 2048                                |             |               |
| <b>Feature Extractor</b>                 | ✓          |                 | $F_s$     | 256                                 |             |               |
|  |            | Input           |           | 2048                                | ✗           |               |
|  |            | Fully connected |           | 1024                                | ✗           | ELU           |
|  |            | Fully connected |           | 1024                                | ✓           | ELU           |
|  |            | Fully connected |           | 256                                 | ✗           | ELU           |
|  |            | Fully connected |           | 256                                 | ✓           | ELU           |
| <b>Decoder</b>                           | ✓          |                 | $G$       | 2048                                |             |               |
|  |            | Input           |           | 256                                 | ✗           |               |
|  |            | Fully connected |           | 1024                                | ✗           | ELU           |
|  |            | Fully connected |           | 1024                                | ✓           | ELU           |
|  |            | Fully connected |           | 2048                                | ✗           | ELU           |
|  |            | Fully connected |           | 2048                                | ✗           | -             |
| <b>Classifier</b>                        | ✓          |                 | $D$       | $ \mathcal{C}_s  +  \mathcal{C}_n $ |             |               |
|  |            | Input           |           | 256                                 | ✗           |               |
|  |            | Fully connected | $D_{src}$ | $ \mathcal{C}_s $                   | ✗           |               |
|  |            | Input           |           | 256                                 | ✗           |               |
|  |            | Fully connected | $D_{neg}$ | $ \mathcal{C}_n $                   | ✗           |               |

Table 6: Network architecture for the *Deployment* stage. Hyperparameter  $\beta = 0.1$

| Component                                | Trainable?   | Operation       | Notation  | Features                            | Batch Norm?  | Non-Linearity |
|--|--------------|-----------------|-----------|-------------------------------------|--------------|---------------|
| <b>Resnet-50</b><br>(Upto AvgPool layer) | $\times$     |                 | $M$       | 2048                                |              |               |
| <b>Feature Extractor</b>                 | $\checkmark$ |                 | $F_t$     | 256                                 |              |               |
|  |              | Input           |           | 2048                                | $\times$     |               |
|  |              | Fully connected |           | 1024                                | $\times$     | ELU           |
|  |              | Fully connected |           | 1024                                | $\checkmark$ | ELU           |
|  |              | Fully connected |           | 256                                 | $\times$     | ELU           |
|  |              | Fully connected |           | 256                                 | $\checkmark$ | ELU           |
| <b>Classifier</b>                        | $\times$     |                 | $D$       | $ \mathcal{C}_s  +  \mathcal{C}_n $ |              |               |
|  |              | Input           |           | 256                                 | $\times$     |               |
|  |              | Fully connected | $D_{src}$ | $ \mathcal{C}_s $                   | $\times$     |               |
|  |              | Input           |           | 256                                 | $\times$     |               |
|  |              | Fully connected | $D_{neg}$ | $ \mathcal{C}_n $                   | $\times$     |               |