Supplementary Material : Domain-Specificity Inducing Transformers for Source-Free Domain Adaptation

Sunandini Sanyal * Ashish Ramayee Asokan * Suvaansh Bhambri * Akshay Kulkarni
Jogendra Nath Kundu R Venkatesh Babu
Vision and AI Lab, Indian Institute of Science, Bengaluru

In this supplementary material, we provide more details on the training algorithm, experimental settings, additional comparisons, and analysis experiments. We have released our code at our project page: https://val.cds.iisc.ac.in/DSiT-SFDA/. The remainder of the supplementary material is structured as shown below:

- Sec. 1: Approach (Algorithm 1)
- Sec. 2: Implementation Details
  - Sec. 2.1: Domain Augmentations (Fig. 1A)
  - Sec. 2.2: DRI Dataset Extraction (Fig. 1B)
  - Sec. 2.3 Domain-specificity criterion
  - Sec. 2.4: Experimental Settings
- Sec. 3 Additional experimental results
  - Sec. 3.1 Extended comparisons (Table 2, 6)
  - Sec. 3.2 Vendor-side DSiT results (Table 3)
  - Sec. 3.3 Model adaptation setting (Table 4)
  - Sec. 3.4 Results on different backbones (Table 5)
  - Sec. 3.5 Analysis for augmentations (Table 8)
  - Sec. 3.6: Sensitivity analysis of DSiT (Table 9)
  - Sec. 3.7 Training time comparisons (Table 7)
  - Sec. 3.8: Statistical significance (Table 10)
  - Sec. 3.9: Target adaptation losses (Table 11)
  - Sec. 3.10: Effect of DRI grid-size (Figure 2)

1. Approach

Table 1 shows a complete list of the notations used in the paper. We summarize our full approach in Algorithm 1 and describe the details of the approach in this section.

Target adaptation losses. For the client-side target adaptation, we use the Information Maximization loss formulation from SHOT [8], which consists of two terms: entropy loss $\mathcal{L}_{im}$ and diversity loss $\mathcal{L}_{div}$. The entropy loss $\mathcal{L}_{im}$ ensures that the confidence of the model towards a label is high. The diversity loss $\mathcal{L}_{div}$ ensures that the model's predictions are well-balanced across all classes and prevents the model from producing degenerate solutions. We define the two terms as follows:

$$\mathcal{L}_{im} = -\mathop{\mathbb{E}}_{x_t \in \mathcal{X}} \sum_{k=1}^{K} \delta_k(f_g(z_c)) \log \delta_k(f_g(z_c)) \quad (1)$$

Table 1. List of all the notations used throughout the paper.

	Symbol	Description
Models	h	Backbone feature extractor
	f_g	Goal task classifier
	f_d	Domain classifier
Transformers	z_c	Class token of last layer
	z_d	Domain token of last layer
	N_P	Number of patch tokens
	W_Q	Query weights
	W_K	Key weights
	W_V	Value weights
	θ_Q	Query weights of all layers
	θ_K	Key weights of all layers
	θ_V	Value weights of all layers
Datasets	$\mathcal{D}_s$	Labeled source dataset
	$\mathcal{D}_t$	Unlabeled target dataset
	$\mathcal{A}_i$	i^{th} augmentation function
	$\mathcal{D}_s^{[i]}$	i^{th} augmented source dataset
	$\mathcal{D}_t^{[i]}$	i^{th} augmented target dataset
	(x_s, y_s)	Labeled source sample
	$(x_s^{[i]}, y_s, y_d)$	Augmented source sample
	x_t	Unlabeled target sample
	$(x_t^{[i]}, y_d)$	Target augmented sample
Spaces	$\mathcal{X}$	Input space
	$\mathcal{C}_g$	Label set for goal task
	$\mathcal{Z}_c$	Class token feature space
	$\mathcal{Z}_d$	Domain token feature space
	$\mathcal{Z}_1, \dots, \mathcal{Z}_{N_P}$	Patch token
Losses	$\mathcal{L}_{dom}$	Domain classification loss
	$\mathcal{L}_{cls}$	Task classification loss
	$\mathcal{L}_{im}$	Entropy loss
	$\mathcal{L}_{div}$	Diversity loss
Criterion	γ_{dom}	Domain specificity
	γ_{cls}	Task specificity
	γ_{all}	Inter-class-inter-domain similarity
	τ	Threshold

*Equal Contribution

$$\mathcal{L}_{div} = \sum_{k=1}^{K} \hat{p_k} \log \hat{p_k} = KL(\hat{p}, \frac{1}{K} 1_K) - \log K \tag{2}$$

where $\delta_k(a) = \frac{\exp(a_k)}{\sum_i \exp(a_i)}$ represents the k^{th} element in the softmax output of $a \in \mathbb{R}^K$, and z_c is the class-token from h for an input x_t. We optimize all parameters of the transformer backbone h, except the query weights θ_Q as follows,

$$\min_{\theta_h \setminus \theta_Q, f_g} \mathbb{E}_{\mathcal{D}_t} [\mathcal{L}_{im} + \mathcal{L}_{div}] \tag{3}$$

We also utilize the clustering method of SHOT [8] for self-supervised pseudo-labeling. First, we obtain the centroid of each class in the target domain via weighted k-means clustering,

$$c_k = \frac{\sum_{x_t \in \mathcal{X}} \delta_k(f_g(z_c)) z_c}{\sum_{x_t \in \mathcal{X}} \delta_k(f_g(z_c))} \tag{4}$$

The centroid characterizes the labels for the samples. In order to obtain a pseudo-label, we choose the closest centroid based on the cosine distance as follows,

$$\hat{y_c} = \arg\min_k D_c(z_c, c_k) \tag{5}$$

where D_c denotes the cosine-distance in the class-token feature space $\mathcal{Z}_c$ between the centroid and the input sample features z_c. As the model keeps training, the centroids are updated after every few iterations, and pseudo-labels are assigned according to the new centroids.

Preliminaries on Transformers. Recently, Vision Transformers (ViT) have been shown to improve significantly on several vision tasks [2]. Self-attention is one of the most important components in the transformer architecture. A ViT takes an image as input $x \in \mathcal{X} = \mathbb{R}^{H \times W \times C}$ in the form of patches of size (P, P), where H, W is the image size and C is the number of channels. The total number of patches is denoted as $N_P = H \times W / P^2$. For self-attention, each patch is projected into Q, K, V with a set of weights W_Q, W_K, W_V respectively. The self-attention [16] is computed as follows,

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{6}$$

where d_k is the dimension of the keys/queries.

2. Implementation Details

In this section, we describe our analysis and benchmark experiments, which includes the augmentation strategies, DRI dataset creation, backbone, and optimization details.

2.1. Domain Augmentations

To induce domain-specificity, we use five label-preserving augmentations to simulate virtual domains (Fig. 1A):

Algorithm 1 DSiT Training Algorithm

Vendor-side training

Input: Let $\mathcal{D}_s$ be source data , $\mathcal{D}_s^{[i]}$ be augmented DRI dataset for each augmentation $\mathcal{A}_i$, ImageNet pre-trained DeiT-B backbone h from [17], randomly initialized goal classifier f_g and randomly initialized domain classifier f_d.
for $iter < MaxIter$ **do**:
Goal task training
  for $iter < MaxTaskIters$ **do**:
    Sample batch from $\mathcal{D}_s$
    Compute $\mathcal{L}_{cls}$ using Eq. 2 (main paper)
    update $\theta_h \setminus \theta_Q, \theta_{f_g}$ by minimizing $\mathcal{L}_{cls}$
  end for
Domain classifier training
  for $iter < MaxDomainIters$ **do**:
    Sample batch of DRI from $\mathcal{D}_s^{[i]}$
    Compute $\mathcal{L}_{dom}$ using Eq. 1 (main paper)
    update θ_Q, θ_{f_d} by minimizing $\mathcal{L}_{dom}$
  end for
  ▷ The two steps are carried out alternatively
end for

Client-side training

Input: Target data $\mathcal{D}_t$, Target augmented DRI data $\mathcal{D}_t^{[i]}$, source-side pretrained backbone h, goal classifier f_g and domain classifier f_d.
for $iter < MaxIter$ **do**:
Goal Task Training
  for $iter < MaxTaskIters$ **do**:
    Sample batch from $\mathcal{D}_t$
    Compute $\mathcal{L}_{im}$ and $\mathcal{L}_{div}$ using Eq. 1, 2 (suppl.)
    update $\theta_h \setminus \theta_Q, \theta_{f_g}$ by minimizing $\mathcal{L}_{im} + \mathcal{L}_{div}$
  end for
Domain classifier training
  for $iter < MaxDomainIters$ **do**:
    Sample batch of DRI from $\mathcal{D}_t^{[i]}$
    Compute $\mathcal{L}_{dom}$ using Eq. 1 (main paper)
    update θ_Q, θ_{f_d} by minimizing $\mathcal{L}_{dom}$
  end for
  ▷ The two steps are carried out alternately
end for

a) FDA augmentation: We use FDA [18] to stylize an image with a fixed style-transfer set of images [5]. This is done by superimposing the amplitude spectrum of the style images onto the input image.

b) Weather augmentations: We employ frost and snow augmentations [7] to augment the input images.

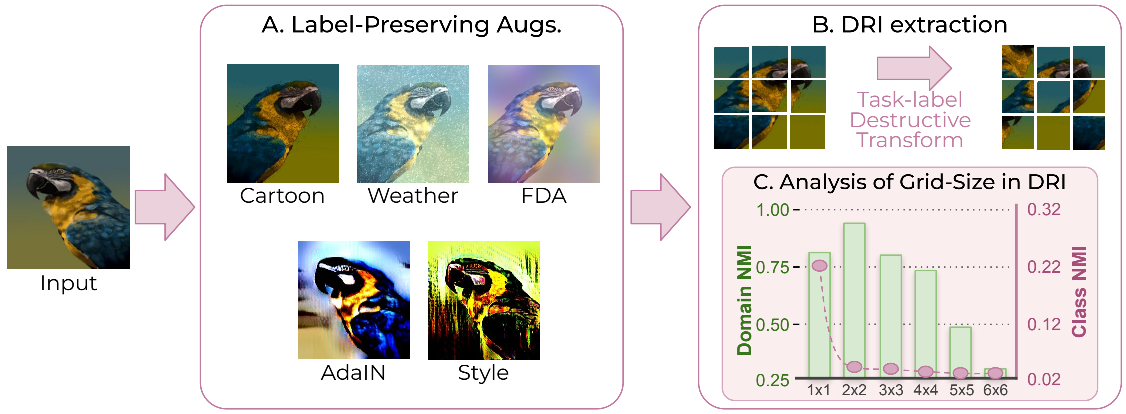

Figure 1. **A.** Label-preserving augmentations are first applied to the input to simulate novel domains. **B.** Then, the task-destructive transform of patch-shuffling is used to obtain the DRI image. **C.** We analyze the Domain-NMI and Class-NMI for different grid-sizes used in patch-shuffling. An example of 3×3 shuffling is shown in B.

c) AdaIN augmentation: In this augmentation [5], we alter the feature statistics through an instance normalization layer [15] that stylizes the images using the same reference style image set as in FDA.

d) Cartoon augmentation: We employ cartoonization-based augmentations [7] to convert inputs to cartoon-like images with reduced texture.

e) Style augmentation: We use stylization from Jackson et al. [6]. No controllable parameters are available and style is chosen without a reference style image.

2.2. DRI Dataset Extraction

The Domain-Representative Inputs (DRI) are created using augmentations as shown in Fig. 1. An input image is first augmented to simulate a virtual domain. Note that only one augmentation is used at a time. After this, the image is shuffled across patches to obtain a DRI image. The extent of patch shuffling is done such that the domain information is still intact, however the task-label information is lost. Following prior works [10], we use normalized mutual information (NMI) to assess the consistency between the feature clusters formed by a self-supervised learning algorithm on the transformed images and the class/domain labels (see Fig. 1C). To obtain NMI, the training images are first subjected to the class-destructive transformation to produce DRI images, and these images are then subjected to self-supervised learning to produce class and domain invariant features. In order to assign a domain or class label to each cluster, we finally apply clustering to the learned features. For the self-supervised learning, we employ SimCLR [1] on the DRI images and apply Gaussian mixture-based clustering to the learned features to obtain either domain or class labels for domain-NMI and class-NMI respectively.

From Figure 1, we see that the Domain NMI rises within a certain range as the number of grid partitions increases, whereas the Class-NMI sharply declines. These results demonstrate that domain-specific features can be learned by using an appropriate grid partition size. Hence, for all our experiments, we have used a grid shuffling size of 4×4 for representing DRI inputs.

2.3. Domain-specificity disentanglement criterion

As discussed in section 3.3 (main) paper we define the a domain-specificity disentanglement criterion based on three parameters: γ_{cls}: intra-class, inter-domain similarity, γ_{dom}: intra-domain inter-class and γ_{all} denotes the inter-class, inter-domain similarity. We define the criterion of domain-specificity disentanglement as follows:

$$\gamma_{dom} = \mathop{\mathbb{E}}_{\mathcal{D}_s \cup \mathcal{D}_t} \mathcal{D}_c(z_{c_1}, z_{c_2}), \text{ where } y_{c_1} \neq y_{c_2}, y_{d_1} = y_{d_2} \tag{7}$$

$$\gamma_{cls} = \mathop{\mathbb{E}}_{\mathcal{D}_s \cup \mathcal{D}_t} \mathcal{D}_c(z_{c_1}, z_{c_2}), \text{ where } y_{c_1} = y_{c_2}, y_{d_1} \neq y_{d_2} \tag{8}$$

$$\gamma_{dom} = \mathop{\mathbb{E}}_{\mathcal{D}_s \cup \mathcal{D}_t} \mathcal{D}_c(z_{c_1}, z_{c_2}), \text{ where } y_{c_1} \neq y_{c_2}, y_{d_1} \neq y_{d_2} \tag{9}$$

where $\mathcal{D}_c(z_{c_1}, z_{c_2})$ denotes cosine similarity between the class-token features of two inputs x_1, x_2 with corresponding class labels y_{c_1}, y_{c_2} and domain labels y_{d_1}, y_{d_2}.

How to choose the threshold τ? We empirically evaluated the metrics γ_{cls}, γ_{dom} and γ_{all} in Table 6 (main paper) and found that task-specificity γ_{cls} and domain-specificity γ_{dom} are closer for DSiT (*Ours*) than the SHOT-B baseline. Based on our observations, we choose a threshold of 0.05.

2.4. Experimental settings

Backbone details. For our experiments, we use DeiT-Base [14] which has 86M parameters, pretrained on ImageNet-1k dataset. DeiT-Base architecture consists of 12 layers, where each layer consists of multi-head self-attention with 12 heads. The input to the transformer is

Table 2. Single-Source Domain Adaptation (SSDA) results on the DomainNet dataset. * indicates results taken from [13].

ResNet-101 [4]	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	19.3	37.5	11.1	52.2	41.0	32.2
inf	30.2	-	31.2	3.6	44.0	27.9	27.4
pnt	39.6	18.7	-	4.9	54.5	36.3	30.8
qdr	7.0	0.9	1.4	-	4.1	8.3	4.3
rel	48.4	22.2	49.4	6.4	-	38.8	33.0
skt	46.9	15.4	37.0	10.9	47.0	-	31.4
Avg.	34.4	15.3	31.3	7.4	40.4	30.5	**26.6**

CDAN [9]	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	20.4	36.6	9.0	50.7	42.3	31.8
inf	27.5	-	25.7	1.8	34.7	20.1	22.0
pnt	42.6	20.0	-	2.5	55.6	38.5	31.8
qdr	21.0	4.5	8.1	-	14.3	15.7	12.7
rel	51.9	23.3	50.4	5.4	-	41.4	34.5
skt	50.8	20.3	43.0	2.9	50.8	-	33.6
Avg.	38.8	17.7	32.8	4.3	41.2	31.6	**27.7**

MIMFTL [3]	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	15.1	35.6	10.7	51.5	43.1	31.2
inf	32.1	-	31.0	2.9	48.5	31.0	29.1
pnt	40.1	14.7	-	4.2	55.4	36.8	30.2
qdr	18.8	3.1	5.0	-	16.0	13.8	11.3
rel	48.5	19.0	47.6	5.8	-	39.4	32.1
skt	51.7	16.5	40.3	12.3	53.5	-	34.9
Avg.	38.2	13.7	31.9	7.2	45.0	32.8	**28.1**

MDD+SCDA [19]	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	20.4	43.3	15.2	59.3	46.5	36.9
inf	32.7	-	34.5	6.3	47.6	29.2	30.1
pnt	46.4	19.9	-	8.1	58.8	42.9	35.2
qdr	31.1	6.6	18.0	-	28.8	22.0	21.3
rel	55.5	23.7	52.9	9.5	-	45.2	37.4
skt	55.8	20.1	46.5	15.0	56.7	-	38.8
Avg.	44.3	18.1	39.0	10.8	50.2	37.2	**33.3**

DeiT-B [14]	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	24.3	49.6	15.8	65.3	52.1	41.4
inf	45.9	-	45.9	6.7	61.4	39.5	39.9
pnt	53.2	23.8	-	6.5	66.4	44.7	38.9
qdr	31.9	6.8	15.4	-	23.4	20.6	19.6
rel	59.0	25.8	56.3	9.16	-	44.8	39.0
skt	60.6	20.6	48.4	16.5	61.2	-	41.5
Avg.	50.1	20.3	43.1	10.9	55.5	40.3	**36.7**

SHOT-B [8]	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	27.0	49.7	16.5	65.4	53.2	46.1
inf	46.4	-	45.9	7.4	60.6	40.1	40.1
pnt	54.6	25.7	-	8.1	66.3	49.0	40.7
qdr	33.3	6.8	15.5	-	23.8	24.0	20.7
rel	59.3	28.1	57.4	9.0	-	47.3	40.2
skt	64.0	26.5	55.0	18.2	63.8	-	45.5
Avg.	51.5	26.6	44.7	11.8	56.0	42.7	**38.9**

CDTrans* [17]	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	27.9	57.6	27.9	73.0	58.8	49.0
inf	58.6	-	53.4	9.6	71.1	47.6	48.1
pnt	60.7	24.0	-	13.0	69.8	49.6	43.4
qdr	2.9	0.4	0.3	-	0.7	4.7	1.8
rel	49.3	18.7	47.8	9.4	-	33.5	31.7
skt	66.8	23.7	54.6	27.5	68.0	-	48.1
Avg.	47.7	18.9	42.7	17.5	56.5	38.8	**37.0**

SSRT-B* [13]	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	33.8	60.2	19.4	75.8	59.8	49.8
inf	55.5	-	54.0	9.0	68.2	44.7	46.3
pnt	61.7	28.5	-	8.4	71.4	55.2	45.0
qdr	42.5	8.8	24.2	-	37.6	33.6	29.3
rel	69.9	37.1	66.0	10.1	-	58.9	48.4
skt	70.6	32.8	62.2	21.7	73.2	-	52.1
Avg.	60.0	28.2	53.3	13.7	65.3	50.4	**45.2**

DSiT (Ours)	clp	inf	pnt	qdr	rel	skt	Avg.
clp	-	27.2	51.8	23.1	70.2	54.7	45.4
inf	52.3	-	48.8	12.8	68.3	44.2	45.3
pnt	59.2	26.1	-	14.5	71.5	51.4	44.5
qdr	38.1	8.3	21.2	-	37.2	27.6	26.5
rel	60.4	28.0	57.8	13.1	-	49.7	41.8
skt	66.3	27.5	56.0	24.4	70.2	-	48.9
Avg.	55.3	23.4	47.1	17.6	63.5	45.5	**42.1**

Table 3. Vendor-side Performance of Single-Source Domain Adaptation (SSDA) on Office-Home, DomainNet and VisDA.

Training stage	Method	Office-Home (4 settings)	VisDA	DomainNet
Vendor-side	SHOT	74.8	67.4	36.7
	DSiT	76.6 **(+1.8)**	68.7 **(+1.3)**	37.8 **(+1.1)**
Client-side	SHOT	79.0	85.9	38.3
	DSiT	81.1 **(+2.1)**	87.5 **(+1.6)**	42.1 **(+3.8)**

Table 4. Single-Source Domain Adaptation (SSDA) on Office-Home (4 settings) for different vendor-side and client-side adaptation strategies.

#	Vendor-side	Client-side	Ar→Cl	Cl→Pr	Pr→Rw	Rw→Ar	Avg
1.	SHOT	SHOT	67.1	83.4	85.3	80.4	79.1
2.	SHOT	DSiT	68.4	85.7	86.8	81.8	80.7 **(+1.6)**
3.	DSiT	SHOT	67.1	83.0	84.9	81.0	79.0
4.	DSiT	DSiT	69.2	86.8	86.6	82.4	81.3 **(+2.3)**

an RGB image that is divided into 16 × 16-sized patches. Therefore, $P = 16$ and $N_P = 14$ for all our experiments. DeiT-B contains an additional distillation token, however, the rest of the architecture is the same as a ViT-B backbone.

Optimization details. For optimizing the training objectives, we use Stochastic Gradient Descent (SGD) with momentum of 0.9, and weight decay ratio of 1×10^{-4}. The learning rate is set to 5×10^{-3} for fine-tuning the domain classifier on the target domain. For the Goal Task training, we use a learning rate of 8×10^{-3} for OfficeHome and VisDA, 8×10^{-2} for Office-31, and 2×10^{-3} for DomainNet. The Goal Task Training and Domain-specific disentanglement Training for the vendor-side source domain are done for 20 epochs, of which 10 are used for warm-up with a warm-up factor of 0.01. In the client-side target adaptation, the goal task training (*task classifier training*) is carried out for 2 epochs, followed by the domain specificity disentanglement (*domain classifier training*) until a domain classification accuracy of 80% is achieved. These two steps are carried out alternatively for an effective 40 epochs of task-classifier training, the same as CDTrans [17]. We use an NVIDIA RTX A5000 GPU with 64GB RAM and 24GB GPU memory to train our models. Our code takes a total training time of approximately 5 hours for Target adaptation training for the Office-Home dataset.

3. Additional Experimental Results

3.1. Extended Comparisons

Comparisons on single-source domain adaptation (SSDA): In Tables 2, we show additional comparisons for our method with existing SSDA works on the DomainNet benchmark. We achieve significant improvements over existing works, especially on CDTrans [17] despite it being a non-source-free method. It is worth noting that CDTrans uses the entire domain during the training and evaluation steps, while we train on the *train* split and evaluate on the *test* split, same as SSRT [13].

Multi-target domain adaptation (MTDA): In Table 6, we provide a quantitative comparison with the prior arts for multi-target domain adaptation on Office-Home. The

Table 5. Single-Source Domain Adaptation (SSDA) on Office-Home on ViT-S Backbone. SF indicates *source-free* adaptation. "-S" denotes Small ViT Backbone.

Method	SF	Office-Home				
		Ar→Cl	Cl→Pr	Pr→Rw	Rw→Ar	Avg.
CDTrans-S [17]	✗	60.7	75.6	84.4	77.0	74.4
SHOT-S	✓	**56.3**	73.7	81.3	76.7	71.9
DSiT-S (*Ours*)	✓	55.3	**77.4**	**83.0**	**76.9**	**73.1** (+1.2)
SHOT-B	✓	69.07	85.31	88.13	83.89	81.6
DSiT-B (*Ours*)	✓	**71.84**	**87.18**	**88.11**	**83.4**	**82.6** (+1.0)

Table 6. Multi-Target Domain Adaptation (MTDA) on Office-Home. SF indicates *source-free* adaptation. ResNet-based methods (top) and Transformer-based methods (bottom).

Method	SF	Office-Home				
		Ar→	Cl→	Pr→	Rw→	Avg.
MT-MTDA [11]	✗	64.6	66.4	59.2	67.1	64.3
CDAN+DCL [9]	✗	63.0	66.3	60.0	67.0	64.1
D-CGCT [12]	✗	70.5	71.6	66.0	71.2	69.8
D-CGCT-B [12]	✗	77.0	78.5	77.9	80.9	78.6
SHOT-B*	✓	75.4	79.3	73.6	77.1	76.4
DSiT-B (*Ours*)	✓	**77.3**	**83.4**	**75.6**	**76.8**	**78.3** (+1.9)

performance improvement is quite prominent (+2.0%) over the source-free prior art (SHOT-B), and the proposed approach also yields comparable performance to non-source-free prior arts, D-CGCT and CDAN+DCL [12], which mainly focus on domain invariant features.

3.2. Vendor-side DSiT Performance

Our DSiT approach incorporates a novel Domain-Specificity Training (DST) that improves the vendor-side performance over the standard source-only baseline (shown in Table 3). Further, we observe significant gains from vendor to client-side (4.5% and 4.3% for Office Home and DomainNet, Table 3). This shows that our vendor-side DST positively aids client-side DST.

Table 7. Training time comparison of our approach DSiT vs SHOT on Office-Home (Rw→Ar)

Method	**Training time (in min)**					**Inf. time (ms)**	**Acc.**
	Src. train	Src. DST	Tgt. adapt	Tgt. DST	Total time		
SHOT-B	12	-	17	-	29	3.6	80.4
Ours	12	109	-	258	270	3.6	**82.4**

3.3. Performance in a Model Adaptation Setting

Our DSiT approach works well even for a model adaptation setting, where we perform DST only on client-side without any specialized training on the vendor-side (#2, Table 4). However, we get the best results when DST is done on both vendor and client-side (#4, Table 4). We also observe that SHOT target adaptation (TA) with our vendor-side DSiT model (#2) gives the same performance as the baseline (#1). This indicates that our proposed TA (#4) is able to better leverage our vendor-side model to yield improved adaptation performance.

Table 8. Analysis for $\mathcal{A}$-distance of three augmentations on 4 settings of Office-Home (SSDA).

Aug.	**Ar→Cl**	**Cl→Pr**	**Pr→Rw**	**Rw→Ar**
FDA	0.857	0.730	0.829	0.286
Original	**1.049**	**0.852**	**0.834**	**0.504**
AdaIN	1.072	0.648	0.842	0.136

Table 9. Sensitivity Analysis on Single-Source Domain Adaptation (SSDA) on Office-Home. (4 settings)

Epochs	**Ar → Cl**	**Cl → Pr**	**Pr → Rw**	**Rw → Ar**	**Avg.**
1	64.1	79.8	84.7	79.6	77.0
2	69.2	86.1	86.6	82.4	81.1
3	69.6	86.7	87.3	82.4	81.5
5	69.5	86.3	87.1	82.5	81.3

3.4. Performance on different backbones

We report results in Table 5 for DeiT-S backbone (with 22M parameters) pre-trained on ImageNet and observe that our approach improves over the baseline SHOT-S baseline by 1.2%. Note that "-S" denotes Small. We also report the results over ViT-B backbone which is trained on ImageNet-21K dataset. Over ViT-B, our approach shows an improvement of 1.0% over the SHOT baseline.

3.5. Experimental analysis for augmentations

In Table 8, we show the $\mathcal{A}$-distance (domain-gap) between augmented source and target domains on Office-Home using the class token of a source-trained DeiT-B. The domain gap for FDA is lower than the original source-target while it is higher for AdaIN. This validates Fig. 4 (main paper) which illustrated that augmented domains may be closer or farther than original domains.

Table 10. Significance experiments of DSiT-B (Ours) on Single-Source DA (SSDA) on Office-Home (4 settings).

Ar → Cl	**Cl → Pr**	**Pr → Rw**	**Rw → Ar**	**Avg.**
69.2 ±0.1	86.1 ±0.3	86.6 ±0.3	82.4 ±0.6	81.1 ±0.1

3.6. Sensitivity Analysis of Alternate Training

In our approach, we perform alternate rounds of training of the domain and the task classifier. We usually train the task classifier for a few epochs, followed by the domain classifier training. In this analysis, we vary the number of epochs of task classifier training from 1 to 5 epochs in an alternate round and observe its impact on task accuracy. Table 9 shows that the task accuracy increases at 2 epochs and is maximum at 3 epochs of training. In all our experiments, we report results with 2 epochs of task classifier training.

3.7. Training time comparisons

We provide detailed training and inference time comparisons of our method with SHOT-B in Table 7. The DST training time is higher due to augmented images being computed at training time, which can be easily reduced by loading pre-computed augmented images. We point out that the inference time remains the same, highlighting the fact that the same DeiT-Base architecture is used for both methods.

3.8. Statistical Significance

We report mean and standard deviation over 3 runs for three Office-Home settings in Table 10. We observe that the standard deviation (0.1 to 0.3) is very low w.r.t. to our gains ($\sim$2%) over SHOT-B.

Table 11. Ablation study for the three components of the client-side adaptation on 4 settings of Office-Home. *PL* indicates pseudo-labeling

Method	Ar → Cl	Cl → Pr	Pr → Rw	Rw → Ar	Avg.
Source Baseline	62.5	79.4	84.3	79.2	76.4
$\mathcal{L}_{im}$	62.1	79.7	80.1	73.9	74.0
$\mathcal{L}_{im} + \mathcal{L}_{div}$	68.2	86.0	86.6	81.3	80.5
$\mathcal{L}_{im} + \mathcal{L}_{div} + PL$	69.2	86.1	86.6	82.4	81.1

3.9. Effect of target adaptation losses

We perform an ablation study on 4 settings of the Office-Home dataset to analyze the influence of each component of the target adaptation objective described in Section 1, and present the results in Table 11. Target adaptation with the entropy loss $\mathcal{L}_{im}$ alone shows sub-optimal results, even when compared to the source-trained baselines, which is also observed by [8]. Adding the diversity loss $\mathcal{L}_{div}$ shows comparatively better performance, indicating that balancing the classifier's predictions across all classes is essential. Lastly, the self-supervised pseudo-labeling *PL* also improves the performance further, demonstrating its importance towards the client-side adaptation.

3.10. Effect of DRI grid-size

Here, we study the effect of varying the DRI grid size for the domain classifier training to determine its influence on the target accuracy (see Figure 2). The target accuracy gradually increases upon increasing the grid size from 1 to 4, with the best results being observed at grid size 4. Beyond this point, the performance begins to drop, which can be attributed to the excessive destruction of information caused by over-partitioning of the images. To achieve a balance between the effect of the task-destructive transformation and

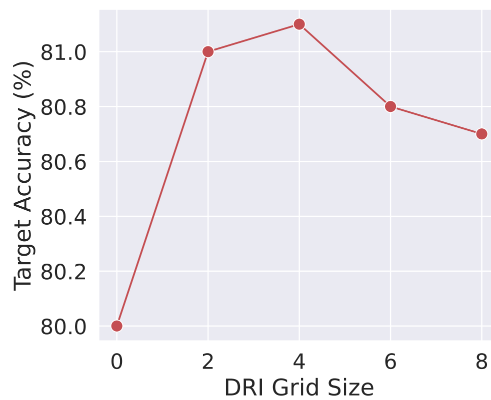

Figure 2. Sensitivity analysis on DRI grid-size for Single-Source DA on Office-Home (4 settings)

its impact on the target accuracy, we use a grid size of 4×4 for all our experiments.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 3

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[3] Jian Gao, Yang Hua, Guosheng Hu, Chi Wang, and Neil M Robertson. Reducing distributional uncertainty by mutual information maximisation and transferable feature learning. In *ECCV*, 2020. 4

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[5] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 3

[6] Philip TG Jackson, Amir Atapour Abarghouei, Stephen Bonner, Toby P Breckon, and Boguslaw Obara. Style augmentation: data augmentation via style randomization. In *CVPR workshops*, 2019. 3

[7] Jung. imgaug. In *https://github.com/aleju/imgaug*, 2020. 2, 3

[8] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020. 1, 2, 4, 6

[9] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2017. 4, 5

[10] Yu Mitsuzumi, Go Irie, Daiki Ikami, and Takashi Shibata. Generalized domain adaptation. In *CVPR*, 2021. 3

[11] Le Thanh Nguyen-Meidine, Atif Belal, Madhu Kiran, Jose Dolz, Louis-Antoine Blais-Morin, and Eric Granger. Unsupervised multi-target domain adaptation through knowledge

distillation. In *WACV*, 2021. 5

[12] Subhankar Roy, Evgeny Krivosheev, Zhun Zhong, Nicu Sebe, and Elisa Ricci. Curriculum graph co-teaching for multi-target domain adaptation. In *CVPR*, 2021. 5

[13] Tao Sun, Cheng Lu, Tianshuo Zhang, and Haibin Ling. Safe self-refinement for transformer-based domain adaptation. In *CVPR*, 2022. 4

[14] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 3, 4

[15] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017. 3

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[17] Tongkun Xu, Weihua Chen, Pichao Wang, Fan Wang, Hao Li, and Rong Jin. CDTrans: Cross-domain transformer for unsupervised domain adaptation. In *ICLR*, 2022. 2, 4, 5

[18] Yanchao Yang and Stefano Soatto. FDA: Fourier domain adaptation for semantic segmentation. In *CVPR*, 2020. 2

[19] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *ICML*, 2019. 4