Supplementary: Robust Unsupervised Domain Adaptation through Negative-View Regularization

```
from torchvision import transforms
from PIL import Image

color_jitter = transforms.ColorJitter()
data_transforms = transforms.Compose([transforms.RandomResizedCrop(),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.RandomApply(color_jitter),
                                      transforms.RandomGrayscale()]
                                    )
gaussian = GaussianBlur()
norm = transforms.Normalize()
pil2tensor = transforms.ToTensor()

img = Image.open("example.jpg")

temp = data_transforms(img)
temp = gaussian(temp)
temp = pil2tensor(temp)
result = norm(temp)
```

Figure 1. A pseudo-code of pos_aug. For brevity, most arguments are omitted.

A. Methodology

A.1. Proposed negative augmentation-based loss function: Pseudo-code of pos_aug

As mentioned in our paper, we make use of a composition of positive augmentations in [1] which attests to the significance of the composition in contrastive learning, and it is available on PyTorch-based SimCLR repository[1]. A pseudo-code of pos_aug can be found in Figure 1

B. New dataset: Retail-71

B.1. Details: Collecting target-domain samples

To build target-domain dataset, we utilized a shelf with 6 slots (*i.e.*, 2 rows and 3 columns) filled with products, and three FHD webcams for multi top view system. The webcams are 2.7 meters high from the floor, and look down vertically on the shelf. Furthermore, they are connected to a single desktop which receives a series of FHD frames from the webcams, and the frames go through a detection model, YOLOv5m[2]. Figure 2 shows the camera system we utilized. Notably, when a person grabs a product with his/her hand, the appearance of the product is shown in the FHD frames, and the detection model detects the position and size of the product, cropping out the product. All the crops of the detected products are automatically saved in the desktop. Some examples are exhibited in Figure 3. Note that the

[1] PyTorch-based SimCLR
[2] A repository of YOLOv5

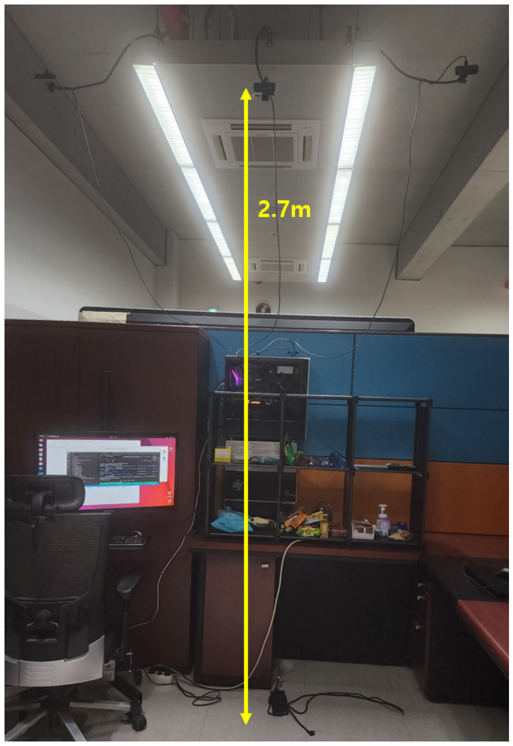


Figure 2. A camera system to acquire target-domain samples.

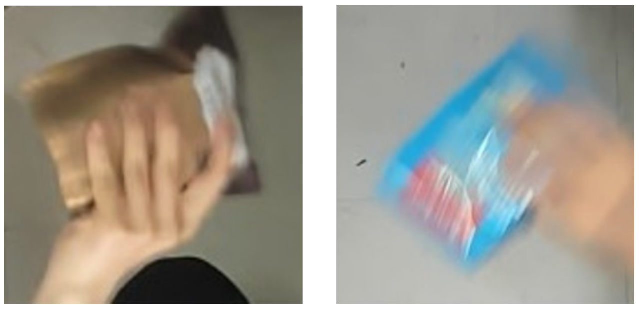


Figure 3. Some examples of target-domain samples.

person is standing still with the product for a few seconds, moves his/her hands slowly and carefully looks around the product for the next few seconds to collect the easy-case images. After that, the person partially occludes the product with his/her hands or strongly shakes it with various strengths so that the cases of severe hand occlusion and hard motion blur are provoked. In this way, we assembled about 1000 images for each product.

620 images per class are randomly sampled and used as target-domain samples for training. The remaining samples, 300-400 samples for each class, are treated as materials for building test sets. We manually select 50 easy-difficulty samples (*i.e.*, relatively clean images), 50 hard-difficulty samples (*i.e.*, images showing harsh motion blur), and 50 medium-difficulty samples (*i.e.*, images exhibiting moderate level of motion blur) for each class. Note that the difficulty of test set is chiefly at the mercy of the degree of motion blur, and the detail criterion can be seen in Subsection B.3.

As a result, the target-domain dataset for training broods total 44,020 samples, and the test set possesses 150 images per class, thus total 10,650 samples. Technically, the test set is composed with three subsets: the easy-difficulty, medium-difficulty, and hard-difficulty test set, and each subset contains total 3,550 samples (*i.e.*, 50 samples for each product). The specific criteria for each difficulty can be found in supplementary.

B.2. Training detection model used to collect target-domain samples

As mentioned in preceding paragraph, we utilize YOLOv5m as a detection model. The role of YOLOv5m is to crop out the product images, regardless of the class of product. We collected the data samples to train the detection model by using the multi top view webcams (Figure 2). To be more specific, one or two persons stand in front of a shelf and catch one or two products with their hands. The persons shake the products, varying the strength. During shaking, the webcams transmit FHD frames of the scene to the desktop, and the frames are saved in the desktop. After that, we labeled all the saved FHD frames in YOLO format, using a tool *labelImg*[3] (Refer to Figure 4. Note that only a single class of object (*i.e.*, product) is used when labeling the frames. As a result, we constructed a training set and validation set which consist of 4,801 and 2,845 frames, respectively. Finally, we trained a YOLOv5m, employing the repository of YOLOv5, and the resultant YOLOv5m achieved $mAP^{50}_{Val} = 0.9875$ and $mAP^{50-95}_{Val} = 0.7168$. The YOLOv5m is used as our product detector.

[3] A repository of labelImg

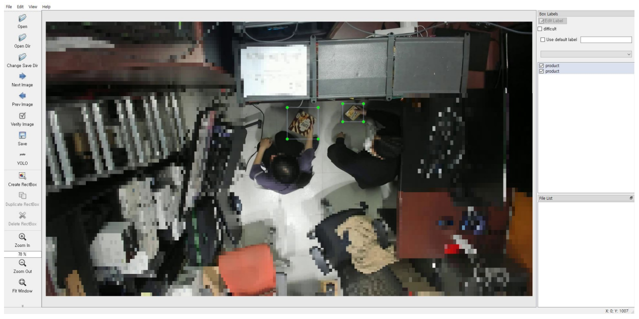


Figure 4. An example of labeling with a tool *labelImg*.

B.3. Details: Criterion for each difficulty in test set

The difficulty of test set is primarily determined depending on the degree of motion blur which is a main factor affecting the features of product images. Although hand occlusion is one of the principal factors, the networks trained with UDA methods typically have trouble predicting the class of images which contain severe motion blur rather than hand occlusion. It is because motion blur hurts the features of products *entirely* even in the case of product images in which the severity of hand occlusion is low, whereas hand occlusion *partially* occludes the features of product image. For each difficulty of test set, We complied with the following guidelines when collecting the test samples.

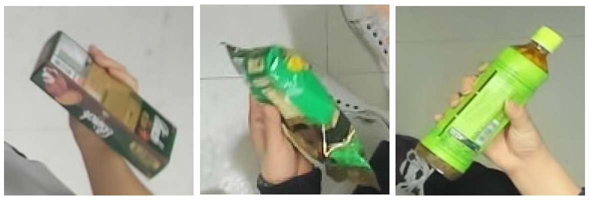
Easy difficulty. It requires the clean images without motion blur. Even if motion blur appears in images, the hardly blurry images are also allowed. Importantly, the texts or pictures printed in the products have to be recognizable to some extent.

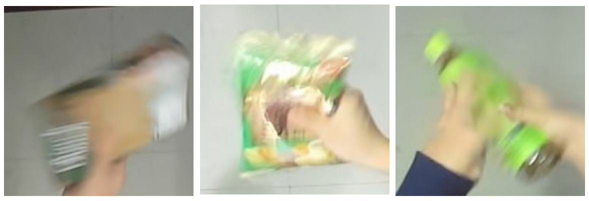
Medium difficulty. The images are so blurry that the texts and pictures printed in the products are unrecognizable. Furthermore, there are some additional acceptable cases in which the edge of product is blurry but its form is vaguely recognizable.

Hard difficulty. The hard-difficulty cases ought to be the images in which the products are extremely blurry to the extent that the shape of products is unrecognizable and distorted. Some examples of each difficulty are shown in Figure 5.

C. Details: Rule-based synthesis: Building an intermediate domain of Retail-71

As mentioned in our paper, the domain discrepancy between source and target domain in Retail-71 results from the myriad of factors, principally hand occlusion and motion blur. Additionally, the images on source domain typ-


(a) Easy difficulty.


(b) Medium difficulty.

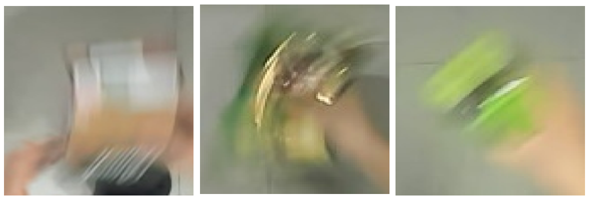

(c) Hard difficulty.

Figure 5. The examples of test images. Each row shows identical difficulty, and the images in each column are from the same class.

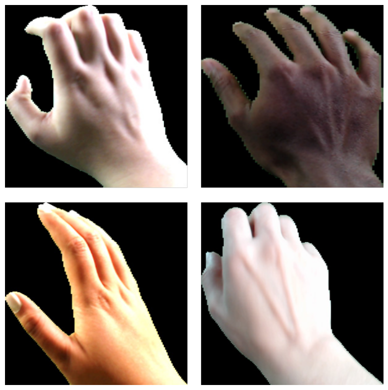


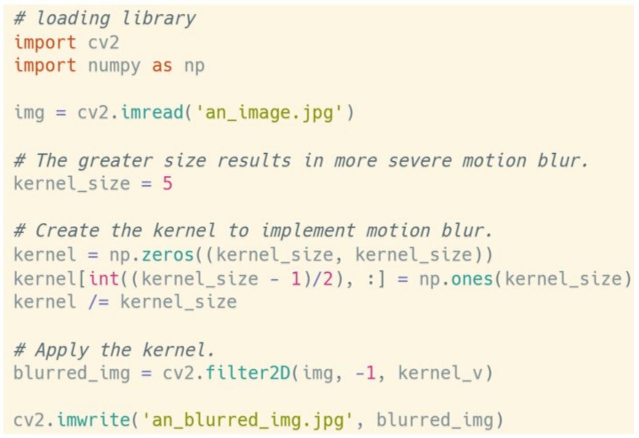
Figure 6. Examples of hand images mined from Ego2Hands [5].

ically show lower resolution than the ones on target domain because the devices used to build each domain are different each other, causing device noise. We propose a rule-based synthesis (RS) for further smooth domain alignment on Retail-71 through building an intermediate domain. The synthesis is applied to source-domain images, and no ground-truth labels are required. Note that there are a variety level of motion blur and hand occlusion in target-domain samples, hence we vary the degree of RS when applying the rule-based synthesis to source-domain samples. Technically, we construct three intermediate-domain datasets (*i.e.*, easy difficulty (E), medium difficulty (M), and hard difficulty (H)) from an original source-domain dataset (O). For the medium-difficulty intermediate-domain dataset, we also apply zero padding, and the resultant intermediate dataset is denoted as MP. Some examples are shown in Figure 8. There are four factors in RS: (1) hand attachment, (2) putting on motion blur, (3) applying a certain level of noise, and (4) zero padding. The following paragraphs deal with the factors.

Hand attachment. It needs Plenty of hand images to manufacture the images of hand-occluded products from source-domain samples. We designated Ego2Hands [5] as the source of hand images. It provides about 180k hand images, each of which has a format of RGBA. In the fourth channel (*i.e.*, channel A), 1 and 0 mean full opacity and transparency, respectively, which helps to separate hand in each image from the background in aids of automated algorithm. Following to removing backgrounds, we manually sampled 392 hands with various appearance, filtering the similar hands as shown in Figure 6.

After that, the certain number of hands are sampled, resized, rotated, and attached to source-domain samples. Concretely, for each source-domain sample, the certain number of hands have to be prepared. The default size of each hand crop is determined based on the length of the longer side of product image. For instance, if the height of the product image (product height) is longer than the product width, the hand crop is resized with its $\text{hand\_height}_{target} = \text{product\_height} \times 0.75$, $\text{hand\_width}_{target} = \text{hand\_height}_{target} \times \frac{\text{hand\_width}}{\text{hand\_height}}$. However, if $\text{hand\_width}_{target} > \text{product\_width}$, the target hand height and target hand width are recalculated: $\text{hand\_width}_{target} = \text{product\_width} \times 1.2$, $\text{hand\_height}_{target} = \text{hand\_width}_{target} \times \frac{\text{hand\_height}}{\text{hand\_width}}$. Importantly, to vary the hand size, we introduce a factor 'size_rate' and multiply it to $\text{hand\_width}_{target}$ and $\text{hand\_height}_{target}$. The same rule is applied in the case that the product width is longer than the product height, except for switching between height and weight.

Next, each hand crop resized with $\text{hand\_width}_{target}$ and $\text{hand\_height}_{target}$ is attached to the product image. Before attachment, the product image goes through zero padding, where 10% of product height is padded to the top and bottom of product image, and 10% of product width is padded to the left and right side of product image. For example, if the product image has a size of $H = 100$ and $W = 50$, the top and bottom side is respectively zero-padded with the thickness 10, and the left and right side is respectively zero-padded with the thickness 5, resulting in the padded product image with a size of $H_{pad} = 120$ and $W_{pad} = 60$. After padding, each hand crop is attached to the padded product image, and the position to which the hands are attached is randomly determined. Before finishing the step of attachment, the padded area of the padded product image is cut



```
# loading library
import cv2
import numpy as np

img = cv2.imread('an_image.jpg')

# The greater size results in more severe motion blur.
kernel_size = 5

# Create the kernel to implement motion blur.
kernel = np.zeros((kernel_size, kernel_size))
kernel[int((kernel_size - 1)/2), :] = np.ones(kernel_size)
kernel /= kernel_size

# Apply the kernel.
blurred_img = cv2.filter2D(img, -1, kernel_v)

cv2.imwrite('an_blurred_img.jpg', blurred_img)
```

Figure 7. Pseudo-code for applying motion blur to image.

off, regaining the size of the original product image.

Note that there are two factors enabling to adjust the level of difficulty in hand attachment: size_rate and the number of hands.

Putting on motion blur. The next step of hand attachment is to put on motion blur. To implement motion blur, we make use of simple kernel and convolutional operation (Refer to Figure 7). Greater size the kernel has, more blurry image are generated. Hence, the kernel_size gives the freedom to control the degree of motion blur.

Applying noise. Following putting on motion blur, we apply little noise to source-domain images to imitate the device noise from webcams which are used to collect target-domain samples. Technically, we employ White Gaussian Noise (WGN), and adjust the degree of noise by changing its standard deviation value.

The setting of factors. Given an original source-domain dataset (O) of Retail-71, we apply hand attachment, motion blur, and noise to generate intermediate-domain datasets. As aforementioned, there are three difficulties: E, M, and H. For easy difficulty (E), the number of hands is randomly determined between 1 and 2, and size_rate is randomly sampled from a uniform distribution (min=0.5, max=0.9) for each hand crop. When it comes to motion blur, kernel_size is set as 0 or sampled from a range of integer (min=5, max=14). The standard deviation of WGN is 5. In the case of medium difficulty (M), the number of hands is randomly determined between 2 and 3, and size_rate is randomly sampled from a uniform distribution (min=0.9, max=1.1) for each hand crop. The value of kernel_size is randomly sampled from either a range of integer (min=5, max=14) or a range of integer (min=25, max=39). The standard deviation of WGN is 5. Lastly, for hard difficulty (H), the number of

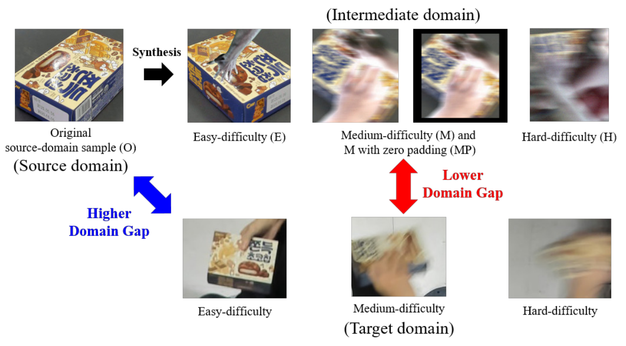


Figure 8. The rule-based synthesis on Retail-71. The goal of the synthesis is to build intermediate domains which have lower domain gap with target domain. We build four domains: E, M, MP, and H. When a neural network is learned by UDA method on Retail-71, the intermediate domains can be utilized by replacing the source-domain training set with the combination of source-domain and intermediate-domain dataset (*e.g.* replace O with O+E+M).

hands is randomly sampled from a range of integer (min=4, max=7), and size_rate is randomly selected from a uniform distribution (min=1.1, max=1.5) for each hand crop. The kernel_size value is randomly sampled from either a range of integer (min=25, max=39) or a range of integer (min=70, max=79). The standard deviation of WGN is 10.

Zero padding. Lastly, when it comes to middle-difficulty images (*i.e.*, M in Figure 8), we zero-pad them to center-align the image pixels and preserve the aspect ratio of original image against resizing which is included in data preprocessing in dataloader. Notably, the edge of products in many target-domain samples have a margin with the edge of image (see Figure 3), while the edge of products in source-domain ones has no margin with and even amputated by the edge of image (see (O) in Figure 8). That is why we apply zero padding to middle-difficulty intermediate-domain samples. Concretely, denoting the size of product image as $(\text{Height}, \text{Width}) = (H, W)$, we firstly make the image to the form of square with zero-padding, and then resize the image to $(0.9H, 0.9W)$. Finally, we zero-pad the resized image and make it having the size (H, W). For example, for the product image with size $(80, 100)$, we zero-pad the top and bottom side of the image with thickness=10, making the image size to $(100, 100)$. After that, the image is resized to the size of $(90, 90)$, and zero padding is applied to the top, bottom, left, and right side with thickness of 5. The size of the resultant image is $(100, 100)$. MP in Figure 8 displays an example.

D. Experimental results

D.1. Details: Datasets

Office-31 [11]. It contains 31 classes of office supplies and three domains: DSLR (D), Webcam (W), and Amazon (A). DSLR and Webcam have 498 and 795 samples, respectively, while Amazon is composed of 2,817 samples. There are total $3 \times 2 = 6$ adaptation scenarios in Office-31.

Office-Home [12]. It has 65 classes and total four domains: Art (Ar), Clipart (Cl), Product (Pr), and Real World (Rw). The smallest domain Ar consists of 2,427 samples, while Cl, Pr, and Rw contain a similar number of samples: 4,365, 4,439, and 4,357 images, respectively. The total number of scenarios is $4 \times 3 = 12$.

VisDA-2017 [7]. Unlike aforementioned two datasets, there are only two domains which are Real and Synthetic, and only one scenario (*i.e.*, Synthetic $\rightarrow$ Real). Real and Synthetic domain possess 152,397 and 55,388 samples, respectively. In addition, the per-class evaluation is performed, and the final accuracy is determined as the average of all per-class accuracies.

Retail-71. It has two domains and one adaptation scenario. The number of classes is 71, and there are 10,650 and 44,020 samples in source and target domain, respectively. On Retail-71, we report five accuracies: a validation accuracy, three test accuracy values for the three difficulties (*i.e.*, easy-difficulty, medium-difficulty, and hard-difficulty accuracy), and a final test accuracy.

D.2. Details: Experimental settings

Backbone. We predominantly exploit ViT-Base except for Retail-71 on which ViT-Small and ViT-Tiny are used. Note that we report UDA performance of both ViT-Small and ViT-Tiny on Retail-71.

Resources. We used 3 NVIDIA TITAN RTX GPUs, each of which has a RAM size of 23.65GB. Specifically, training one ViT-Base requires 3 GPUs, while one ViT-Small and one ViT-Tiny only need two GPUs and a single GPU, respectively.

Baseline method. We appoint SDAT [9], one of the state-of-the-art (SOTA) methods, as a baseline method. In SDAT, MCC [4] is attached over CDAN [6], and a smoothness enhancing loss based on [2] is utilized. The batch size is set to 96 rather than 24 or 32 so that each minibatch reflects the distribution of dataset. The learning rate value is 0.007, 0.01, 0.002, and 0.015 on Office-31, Office-Home, VisDA-2017, and Retail-71, respectively. When it comes to other training hyperparameters, we follow the original ones [9].

Hyperparameter of our regularizer loss. We attach our negative augmentation-based regularizer to the baseline SDAT. In first, when it comes to the negative augmentation, we basically use P-Shuffle with patch size 32 and image size 224, as mentioned in Methodology section. Next, there are two coefficients in our regularizer function, which are a trade-off coefficient α and a temperature τ. The value of α is 0.5 except for Office-31 on which 0.9 is used. As for the value of β, Office-31 and VisDA-2017 use 0.7 and 0.1, respectively, while it is 0.5 on both Office-Home and Retail-71.

D.3. Details: Evaluation on Retail-71 with Rule-based Synthesis

Focusing on Rule-based Synthesis (RS), we report evaluation results in Table 1. We introduced RS into original source-domain samples (O) and generated intermediate-domain samples (E, M, MP, and H), varying the degree of RS. After that, we made diverse combinations of source and intermediate samples and introduced them into our experiments by replacing original source-domain dataset with the combinations. For ViT-Small, it causes its best test accuracy 95.9% to mix O with E and MP, while a mixture of O and M shows the best performance 94.5% for ViT-Tiny. Interestingly, usage of H exhibits relatively small performance boost or even degradation. It is because the hard-difficulty RS which makes H uses more and bigger hand images, and leads to excessive hand occlusion, overly hiding and injuring the features of products in images (See the example of H in Figure 8). That is, utilizing the images (*i.e.*, H) with the features overly destroyed may confuse and disrupt the neural network, which leads to small performance boost or degradation.

D.4. Ablation study

The hyperparameters of our regularizer loss NVC specified in Subsection D.2 are empirically determined, and we show the results of the parameter study on some datasets in this Subsection. Interestingly, the immoderate values of α sometimes give rise to mode collapse, especially in Table 6 and 8. Our NVC regularizer pushes original target-domain samples and their positive views from all the negative views, technically causes original samples and negative views push *each other* away in a contrastive manner. Importantly, original samples may also be pushed out of negative samples. Hence, bigger α increases the gradient signal from NVC loss, and the increased gradient possibly disturbs the distribution of target samples (*i.e.*, devastates the clusters in the distribution), leading to mode collapse.

Method	Backbone	Val.	Easy Test	Medium Test	Hard Test	Avg. Test
Source Only	ViT-Small	49.0	72.4	43.5	18.7	44.9
Source Only+**RS (O+E+MP)**	ViT-Small	55.8	77.4	52.1	27.5	52.3
SDAT†	ViT-Small	95.2	97.7	95.7	87.5	93.7
SDAT†+**RS (O+E+MP)**	ViT-Small	96.0	98.6	96.6	89.4	94.9
SDAT†+**Ours**	ViT-Small	96.0	98.6	96.5	88.3	94.4
SDAT†+**Ours+RS (O+E)**	ViT-Small	95.9	98.5	96.3	89.1	94.6
SDAT†+**Ours+RS (O+M)**	ViT-Small	96.3	98.7	97.0	90.1	95.2
SDAT†+**Ours+RS (O+MP)**	ViT-Small	96.4	98.7	96.7	89.9	95.1
SDAT†+**Ours+RS (O+H)**	ViT-Small	94.7	98.3	95.9	86.0	93.4
SDAT†+**Ours+RS (O+E+M)**	ViT-Small	97.0	99.1	97.6	90.4	95.7
SDAT†+**Ours+RS (O+E+MP)**	ViT-Small	**97.0**	**99.1**	**97.5**	**90.9**	**95.9**
SDAT†+**Ours+RS (O+E+H)**	ViT-Small	95.4	98.5	96.4	88.0	94.3
SDAT†+**Ours+RS (O+E+M+H)**	ViT-Small	95.5	98.6	96.2	87.5	94.1
SDAT†+**Ours+RS (O+E+MP+H)**	ViT-Small	95.7	98.4	96.6	87.8	94.3
SDAT†+**Ours+RS (O+E+M+MP+H)**	ViT-Small	95.6	98.5	96.5	88.5	94.5
Source Only	ViT-Tiny	41.3	58.9	37.3	19.0	38.4
Source Only+**RS (O+M)**	ViT-Tiny	48.2	67.1	45.5	25.7	46.1
SDAT†	ViT-Tiny	93.4	95.8	93.7	83.2	90.9
SDAT†+**RS (O+M)**	ViT-Tiny	95.1	98.4	95.9	86.4	93.6
SDAT†+**Ours**	ViT-Tiny	94.8	97.8	95.2	85.2	92.7
SDAT†+**Ours+RS (O+E)**	ViT-Tiny	94.2	97.6	95.1	85.3	92.7
SDAT†+**Ours+RS (O+M)**	ViT-Tiny	**95.8**	**99.1**	**96.8**	**87.7**	**94.5**
SDAT†+**Ours+RS (O+MP)**	ViT-Tiny	95.6	98.9	96.8	87.6	94.5
SDAT†+**Ours+RS (O+H)**	ViT-Tiny	94.8	98.8	96.3	85.4	93.5
SDAT†+**Ours+RS (O+E+M)**	ViT-Tiny	95.9	99.1	96.7	87.0	94.3
SDAT†+**Ours+RS (O+E+MP)**	ViT-Tiny	95.5	98.9	96.7	86.9	94.2
SDAT†+**Ours+RS (O+E+H)**	ViT-Tiny	95.1	99.0	96.4	86.4	93.9
SDAT†+**Ours+RS (O+E+M+MP)**	ViT-Tiny	95.4	98.7	96.6	86.4	93.9
SDAT†+**Ours+RS (O+E+M+H)**	ViT-Tiny	95.3	99.0	96.4	86.2	93.9
SDAT†+**Ours+RS (O+E+M+MP+H)**	ViT-Tiny	95.2	99.1	96.6	86.4	94.0

Table 1. Accuracy (%) details on Retail-71 for UDA. We used a variety of combination of source-domain dataset (O) and intermediate-domain datasets (E, M, MP, and H). As mentioned in paper, there are various levels of motion blur and hand occlusion in target samples, and thus to imitate it, we vary the degree of RS (see Figure 8 when building intermediate domains. The characters in each parenthesis denote the composition of source dataset and intermediate datasets, e.g. 'O+E+MP' means that the mixture of O, E, and MP. The composition plays the role of source-domain dataset in training. The composition 'O+E+MP' and 'O+M' result in the best performance of ViT-Small and ViT-Tiny, respectively, and they are reported in paper as representative performances on Retail-71. The identical compositions leading to best performances ('O+E+MP' for ViT-Small and 'O+M' for ViT-Tiny) are also used to SDAT† and Source Only. † indicates that it is reproduced with batch size 96.

α	0.0	0.1	0.3	0.5	0.7	0.9	1.0	1.5
SDAT†+**Ours**	84.9	85.2	85.4	85.4	85.3	**85.6**	85.1	84.7

Table 2. Parameter study of the trade-off coefficient α with ViT-Base on a scenario W$\rightarrow$A of Office-31. Note that the case of $\alpha = 0.0$ is identical to vanilla SDAT. The batch size is 96, and the temperature τ is fixed to 0.7.

τ	0.1	0.3	0.5	0.7	0.9	1.0	1.5	2.0
SDAT†+**Ours**	81.7	84.1	84.7	**85.6**	85.1	85.1	85.2	85.3

Table 3. Parameter study of the temperature τ with ViT-Base on a scenario W$\rightarrow$A of Office-31. The batch size is 96, and the trade-off coefficient α is fixed to 0.9.

D.5. Average Negative Confidence Score

In our paper, we analyzed our trained neural networks based on not only Negative Accuracy but also Average Negative Confidence Score. In this subsection, we give a supplementary explanation of Average Negative Confidence Score. As displayed in our paper, the definition of Average Negative Confidence Score is in Equation 1, in which $\mathbb{1}\!\!\!/$ is an indicator function which has a value 1 only when

α	0.0	0.1	0.3	0.5	0.7	0.9	1.0	1.5
SDAT[†]+**Ours**	74.1	74.8	74.5	**75.1**	72.8	71.8	70.4	63.5

Table 4. Parameter study of the trade-off coefficient α with ViT-Base on a scenario Ar→Cl of Office-Home. Note that the case of $\alpha = 0.0$ is identical to vanilla SDAT. The batch size is 96, and the temperature τ is fixed to 0.5.

τ	0.1	0.3	0.5	0.7	0.9	1.0	1.5	2.0
SDAT[†]+**Ours**	70.9	71.9	**75.1**	74.7	74.0	74.7	74.7	74.3

Table 5. Parameter study of the temperature τ with ViT-Base on a scenario Ar→Cl of Office-Home. The batch size is 96, and the trade-off coefficient α is fixed to 0.5.

α	0.0	0.1	0.3	0.5	0.7	0.9	1.0	1.5
SDAT[†]+**Ours**	95.2	95.6	95.8	**96.0**	30.7	5.7	3.7	2.2

Table 6. Parameter study of the trade-off coefficient α with ViT-Small on Retail-71. We report the validation accuracy. Note that the case of $\alpha = 0.0$ is identical to vanilla SDAT. The batch size is 96, and the temperature τ is fixed to 0.5.

τ	0.1	0.3	0.5	0.7	0.9	1.0	1.5	2.0
SDAT[†]+**Ours**	5.0	5.9	**96.0**	95.0	95.8	94.9	94.7	95.5

Table 7. Parameter study of the temperature τ with ViT-Small on Retail-71. We report the validation accuracy. The batch size is 96, and the trade-off coefficient α is fixed to 0.5.

α	0.0	0.1	0.3	0.5	0.7	0.9	1.0	1.5
SDAT[†]+**Ours**	93.4	94.5	**94.8**	**94.8**	94.4	93.6	93.1	39.2

Table 8. Parameter study of the trade-off coefficient α with ViT-Tiny on Retail-71. We report the validation accuracy. Note that the case of $\alpha = 0.0$ is identical to vanilla SDAT. The batch size is 96, and the temperature τ is fixed to 0.5.

τ	0.1	0.3	0.5	0.7	0.9	1.0	1.5	2.0
SDAT[†]+**Ours**	92.7	94.3	**94.8**	94.7	94.6	94.5	94.4	94.5

Table 9. Parameter study of the temperature τ with ViT-Tiny on Retail-71. We report the validation accuracy. The batch size is 96, and the trade-off coefficient α is fixed to 0.5.

the equation in its parenthesis is hold, otherwise 0.

$$\begin{aligned} avg\_conf_{neg} &= \frac{1}{|\mathcal{B}^{TN}_{test}|} \sum_{x_i^{tn} \in \mathcal{B}^{TN}_{test}} \sum_{c=1}^{C} G(F(x_i^{tn}))_c \mathbb{1}(c = y_i^t) \\ &= \frac{1}{|\mathcal{B}^{TN}_{test}|} \sum_{x_i^{tn} \in \mathcal{B}^{TN}_{test}} \sum_{c=1}^{C} (\hat{y}_i^{tn})_c \mathbb{1}(c = y_i^t) \end{aligned} \tag{1}$$

Given a labeled target-domain test set $\mathcal{B}^T_{test} = \{(x_i^t, y_i^t)\}_{i=1}^{N_{test}}$, all the negative views can be obtained through $\mathcal{B}^{TN}_{test} = \{(x_i^{tn}, y_i^t)\}_{i=1}^{N_{test}} = neg\_aug(\mathcal{B}^T_{test})$. When each negative view x_i^{tn} is fed into neural network, the confidence score corresponding to the class y_i^t of x_i^{tn} can be obtained, and we term it as *negative confidence score*. Thus, Average Negative Confidence Score avg_conf_{neg} is calculated by averaging over all the negative confidence score.

Original target-domain samples have not only local features but also global contexts, whereas negative views only have local features. In ideal case, when processing original samples, neural network such as ViT attends the global contexts and captures semantically meaningful features as well as local features, yet the network fails to find the meaningful features when digesting negative views lacking of global contexts. In other words, given negative views, the network ideally fail to recognize the class of each negative view, making predictions in the form of uniform distribution (*i.e.*, all the class confidence score is equal to $1/C$). Therefore, all the value of *negative confidence score* is ideally $1/C$, hence the ideal value of avg_conf_{neg} is also $1/C$. Note that the higher avg_conf_{neg} of a network implies more heavy dependence of the neural network on local features.

D.6. Results of Negative Accuracy and Average Negative Confidence Score on various datasets

We only reported the result of Negative Accuracy and Average Negative Confidence Score on Retail-71, and thus we present the other results on different datasets in this subsection. The results are in Table 10, 11, 12, 13, 14, and 15. The baseline SDAT shows high value of both Negative Accuracy and Average Negative Confidence Score, whereas SDAT+Ours records the values relatively close to the ideal values on Office-Home and VisDA-2017 as well as Retail-71. It indicate that our NVC regularizer effectively instructs the neural network to reduce its dependence on local features.

Nevertheless, on Office-31, there is a tendency contrary to the one on other datasets in that SDAT+Ours shows higher values than the baseline SDAT. In first, we pay attention to the fact that Office-31 contains a smaller amount of images than Office-Home, VisDA-2017, and Retail-71 (See Subsection D.1). In second, referring to Section 'Benchmark evaluation' in our paper, we note that 'Source Only'

Method	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
SDAT†	79.7	77.6	64.5	79.9	63.7	83.5	74.8
SDAT†+**Ours**	96.6	91.2	78.5	87.5	63.6	92.0	84.9

Table 10. Negative Accuracy (%) of ViT-Base trained on Office-31 for UDA. † indicates that it is reproduced with batch size 96. In ideal case, the negative accuracy is the same as the accuracy of random guess: $1/(\text{\# of classes}) = 1/31 = 0.0323 = 3.23(\%)$ [8].

accuracy values on Office-31 are higher than the ones on other datasets, implying that Office-31 has less domain shift than other datasets (*i.e*., more easier than other datasets). Moreover, the smaller dataset has less image diversity than the larger one, arousing overfitting, and we venture a guess that on the smaller and easier dataset, it is closer the optima to focus on some local patches of images rather than to extract the contextual information. Consequently, we speculate that our NVC loss makes ViT-based network further concentrating to some local patches relatively helpful and necessary to predict class rather than learning to catch the contextual information, on Office-31, and it leads to high Nagative Accuracy and Average Negative Confidence Score.

D.7. Visualization: T-SNE on various datasets

We show more diverse T-SNE figures on various datasets. For each dataset and backbone, We prepare four models: ImageNet-pretrained model, a model trained only with labeled source-domain dataset, a model learned by a baseline SDAT, and a model trained with SDAT+Ours. That is, we plot four T-SNE for each dataset and backbone. Figure 9, 10, 11, and 12 show all the results.

When it comes to ImageNet-pretrained ViT, it can differentiate original target samples and their negative views to some extent, and we infer that the pretrained ViT is trained on ImageNet-1K [10] dataset, a large-scale dataset, in a supervised manner, so that the ViT acquires an ability to capture the global contexts in images. For short, ImageNet-pretrained ViT is able to extract the semantically meaningful features in image because of a variety of training samples with supervision.

For Source Only, T-SNE shows the domain gap between two domains, especially on Retail-71, and the distributions of original target samples and negative views are aligned, compared with T-SNE of ImageNet-pretrained models. We speculate that it is attributed to catastrophic forgetting [3], *i.e*. the model forgets its knowledge acquired from ImageNet and loses its ability to capture the global context, with learning on other datasets such as Retail-71.

In the case of SDAT, the figures of T-SNE show the alleviated domain gap between source and target domain, yet the distribution of negative views still overlaps with the one of target samples. It indicates the features from target samples are similar to the ones from negative views, and thus the ViT heavily relies on the features from local patches rather than on the relations between them.

On the other hand, the figures from SDAT+Ours show that the distribution of negative views is almost completely separated from the distribution of target samples. Thus, our NVC loss successfully lowers the reliance on local features. Even so, in Figure 12h, some negative views are attached to the clusters even though negative views are differentiated from target samples to some extent. As discussed in Subsection D.6, we suspect that our NVC loss rather prompts the ViT to further focus on some helpful local patches of images on the small and easy dataset, and only some images lacking of the *helpful* local patches are successfully alienated from their corresponding negative views (See the yellow island in the center of Figure 12h).

Method	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg.
SDAT†	32.1	62.4	64.8	47.3	50.7	54.8	41.1	29.8	63.9	49.0	28.9	66.0	49.2
SDAT†+**Ours**	6.6	10.8	9.0	8.7	6.8	8.1	3.1	4.9	6.0	3.4	6.6	47.4	10.1

Table 11. Negative Accuracy (%) of ViT-Base trained on Office-Home for UDA. † indicates that it is reproduced with batch size 96. In ideal case, the negative accuracy is the same as the accuracy of random guess: $1/(\text{\# of classes}) = 1/65 = 0.0154 = 1.54(\%)$ [8].

Method	Plane	Bcycl	Bus	Car	Horse	Knife	Mcyle	Persn	Plant	Sktb	Train	Truck	Mean
SDAT†	88.9	58.0	48.6	36.6	41.4	62.4	78.6	77.5	91.4	58.8	72.8	52.2	63.9
SDAT†+**Ours**	0.03	0.0	0.0	5.0	0.0	8.1	0.07	2.2	39.1	0.0	27.3	0.0	6.8

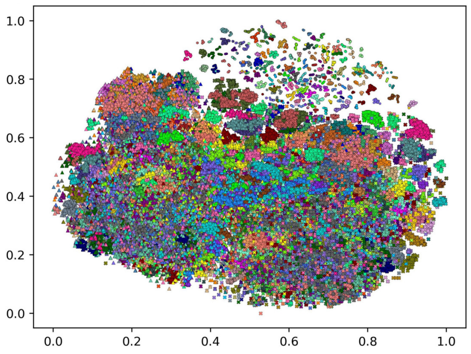
Table 12. Negative Accuracy (%) of ViT-Base trained on VisDA-2017 for UDA. † indicates that it is reproduced with batch size 96. In ideal case, the negative accuracy is the same as the accuracy of random guess: $1/(\text{\# of classes}) = 1/12 = 0.0833 = 8.33(\%)$ [8].

Method	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
SDAT†	0.7582	0.7394	0.6433	0.5228	0.6294	0.4359	0.6215
SDAT†+**Ours**	0.9571	0.9038	0.7680	0.6629	0.5803	0.7094	0.7636

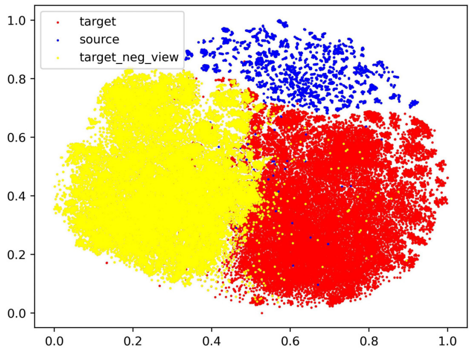
Table 13. Average Negative Confidence Score of ViT-Base trained on Office-31 for UDA. † indicates that it is reproduced with batch size 96. In ideal case, when the learned ViT receives negative views, it predicts the uniform distribution which has its length equal to the number of classes and in which each probability (*i.e.*, confidence score) is $1/(\text{\# of classes}) = 1/31 = 0.0323$. Hence, the average negative confidence score of ground-truth class is ideally $1/(\text{\# of classes}) = 1/31 = 0.0323$.

D.8. Visualization: Additional figures of attention map

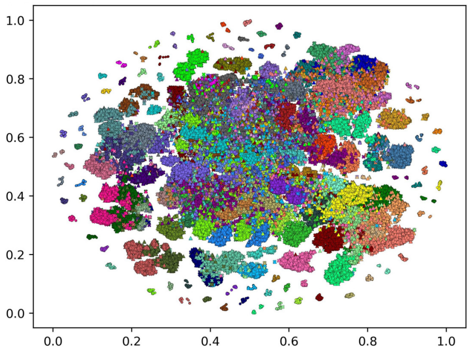
In the paper, a few visualization results of attention map are shown due to the page limit. Thus, we exhibit more results on Retail-71 and Office-Home in this section. In some examples (Figure 13, 14, 15, 16, and 17, the cases of Ours successfully attend to the object and take the semantically meaningful, global features into the class predictions, unlike the ones of SDAT. Meanwhile, referring to other examples (Figure 18, 19, and 20), both SDAT and Ours almost similarily attend to the given object or SDAT even shows more clear attention. Nevertheless, SDAT shows its incorrect predictions, whilst Ours correctly classifies the given objects. It is because Ours relies on not only local features but also the semantically meaningful features and the relations between local patches, unlike SDAT mainly focusing on local features. In conclusion, the visualization results show that our NVC loss leads the ViT to capture the global contexts in images, boosting UDA performance.



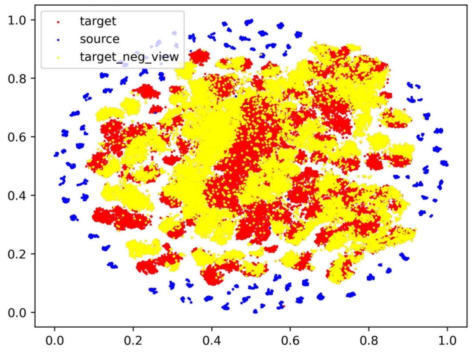
(a) ImageNet-pretrained model.



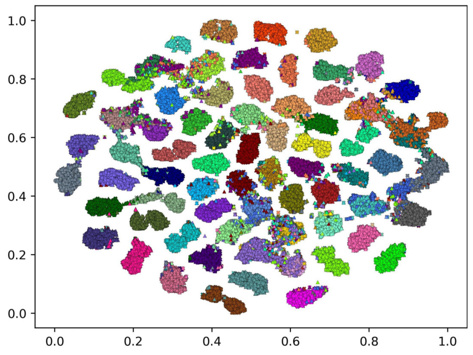
(b) ImageNet-pretrained model.



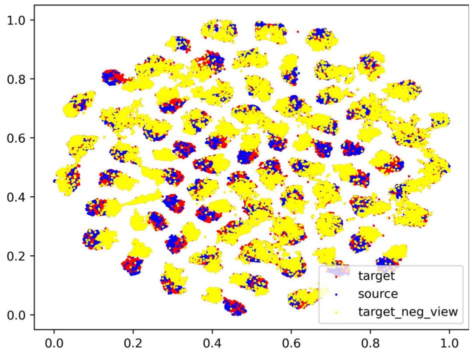
(c) Source Only.



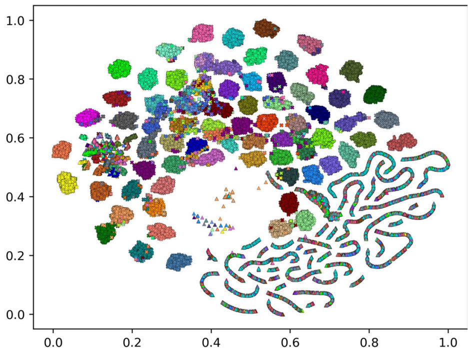
(d) Source Only.



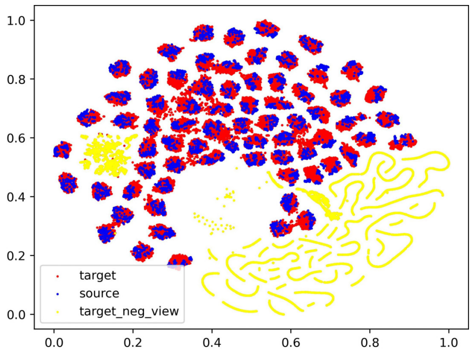
(e) SDAT.



(f) SDAT.

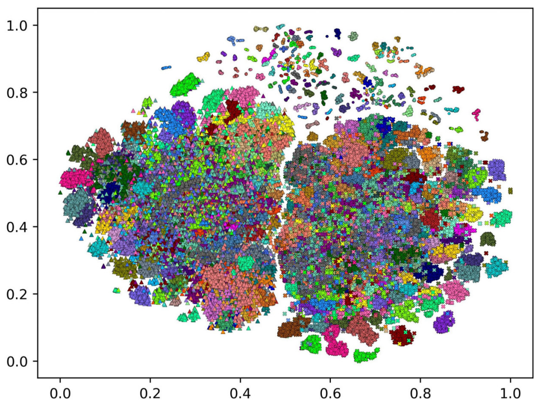


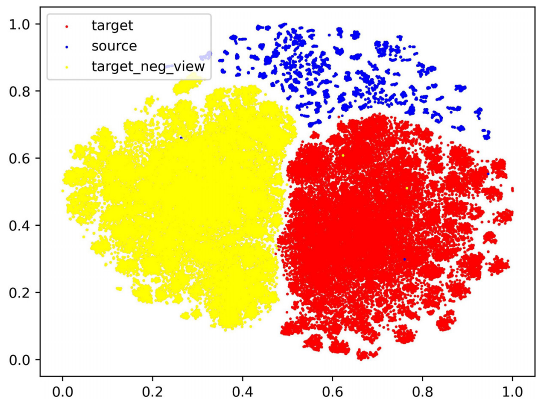
(g) SDAT+Ours.



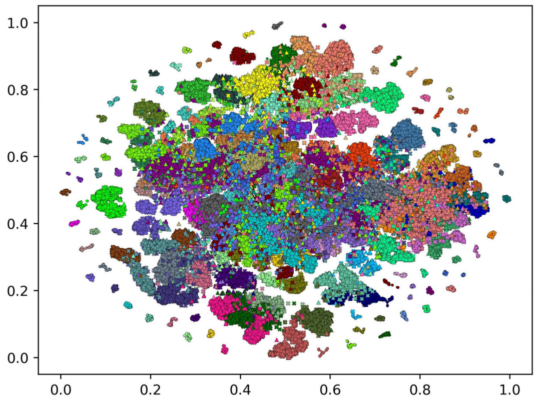
(h) SDAT+Ours.

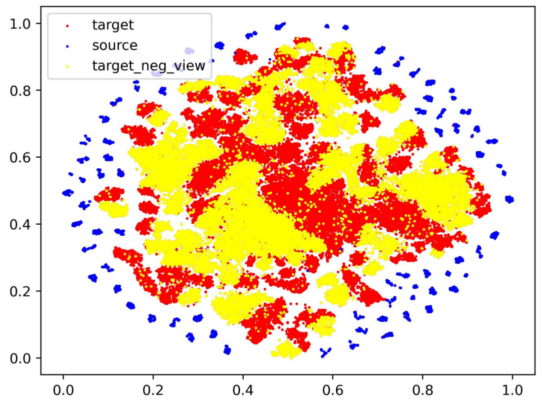
Figure 9. T-SNE of ViT-Tiny pretrained on ImageNet-1k, ViT-Tiny trained only with source-domain samples, and ViT-Tiny learned by SDAT and SDAT+Ours on Retail-71. Each color in (a), (c), (e), and (g) is corresponding to each class, while each color in (b), (d), (f), and (h) indicates each domain. They also include the negative views of target-domain samples which are marked in yellow in (b), (d), (f), and (h). The plot (e) and (f) exhibits that ViT learned by SDAT heavily depends on local features rather than global contexts in that the distribution of negative views is similar to the one of original target samples. On the other hand, ours successfully captures the contextual relations between local patches, pushing negative views away.


(a) ImageNet-pretrained model.

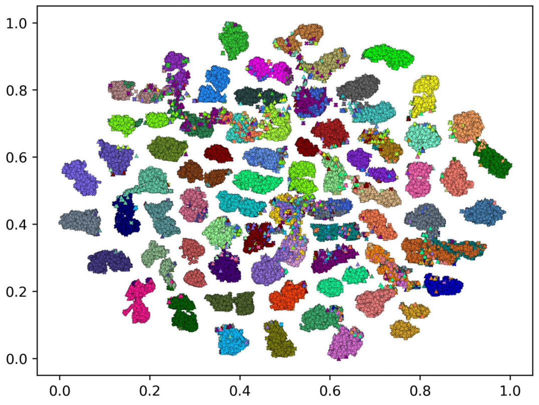


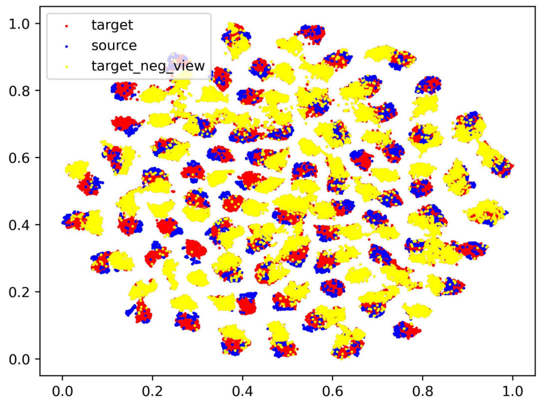
(b) ImageNet-pretrained model.


(c) Source Only.

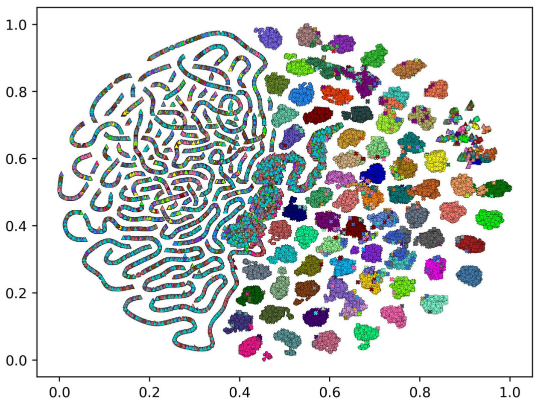


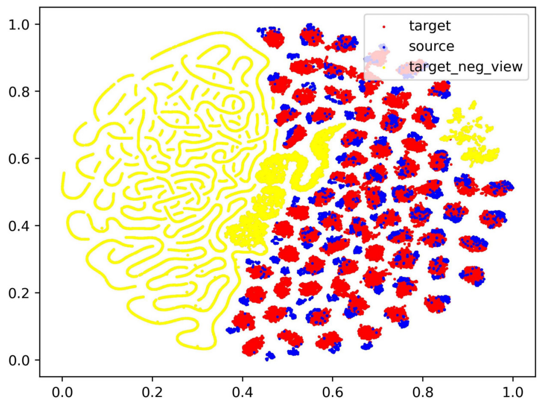
(d) Source Only.


(e) SDAT.



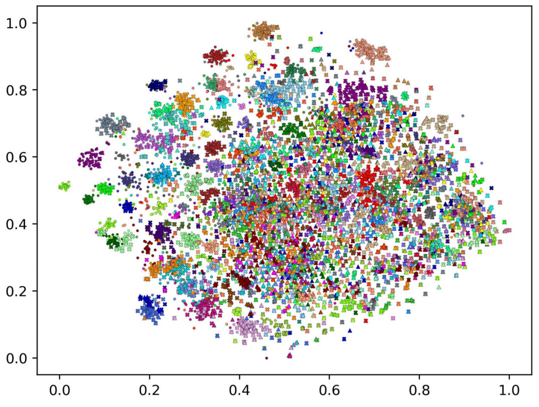
(f) SDAT.


(g) SDAT+Ours.

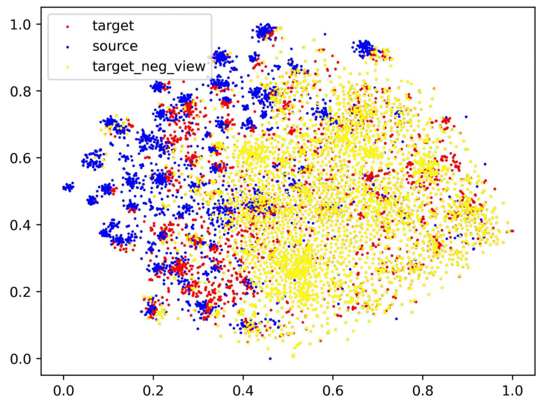


(h) SDAT+Ours.

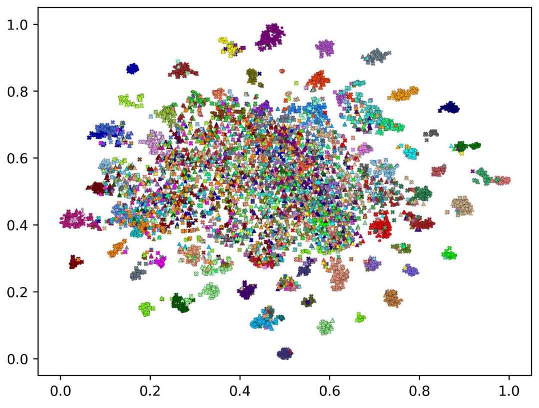
Figure 10. T-SNE of ViT-Small pretrained on ImageNet-1k, ViT-Small trained only with source-domain samples, and ViT-Small learned by SDAT and SDAT+Ours on Retail-71.



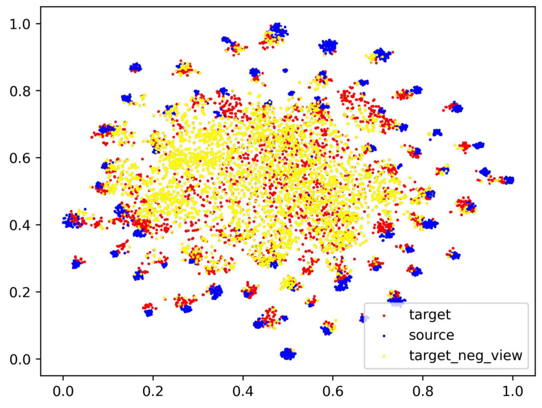
(a) ImageNet-pretrained model.



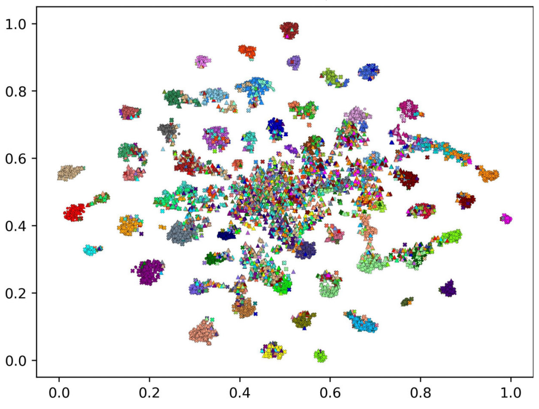
(b) ImageNet-pretrained model.



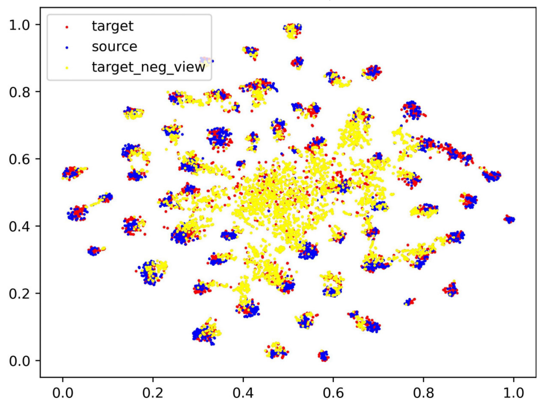
(c) Source Only.



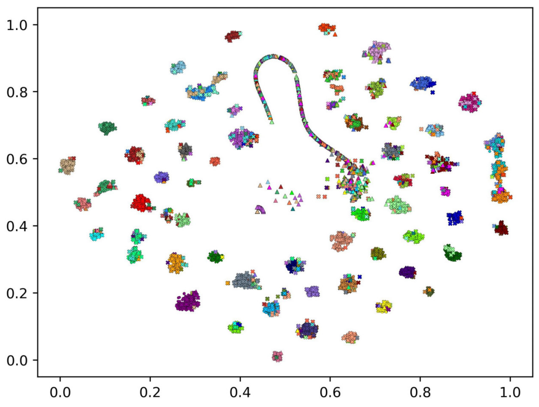
(d) Source Only.



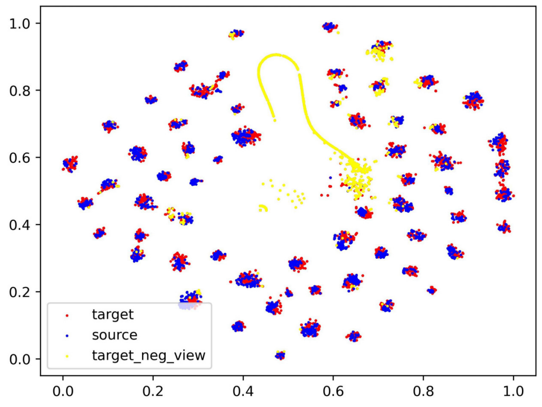
(e) SDAT.



(f) SDAT.

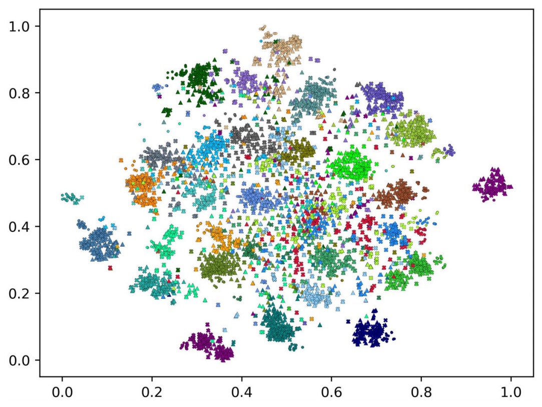


(g) SDAT+Ours.

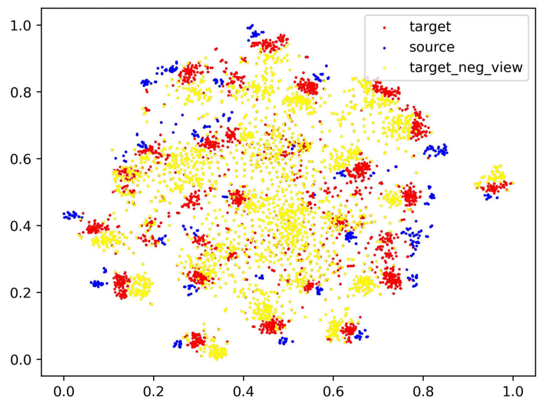


(h) SDAT+Ours.

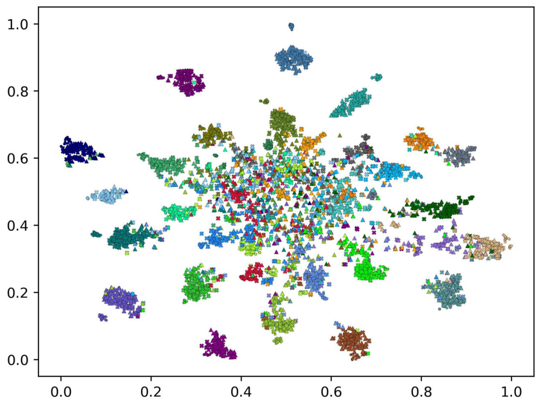
Figure 11. T-SNE of ViT-Base pretrained on ImageNet-1k, ViT-Base trained only with source-domain samples, and ViT-Base learned by SDAT and SDAT+Ours on 'Ar→Cl' scenario of Office-Home.



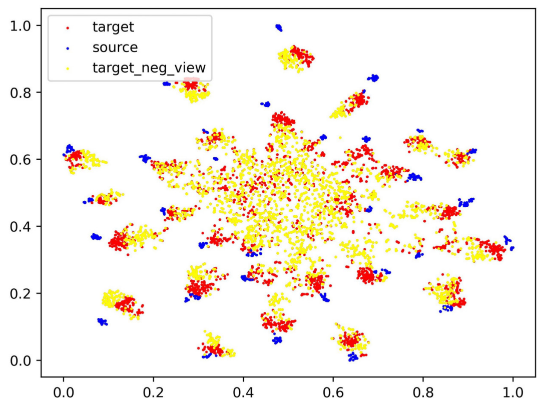
(a) ImageNet-pretrained model.



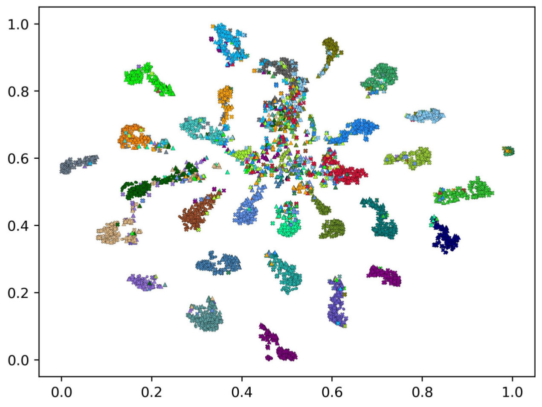
(b) ImageNet-pretrained model.



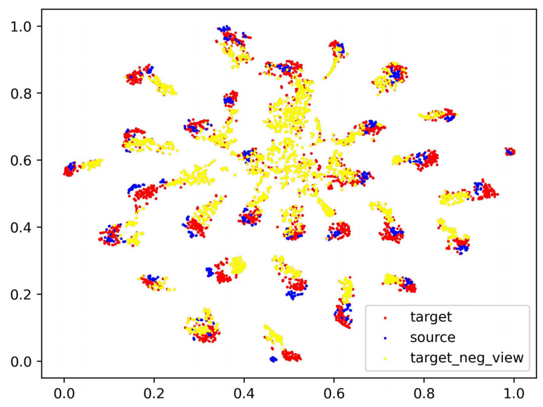
(c) Source Only.



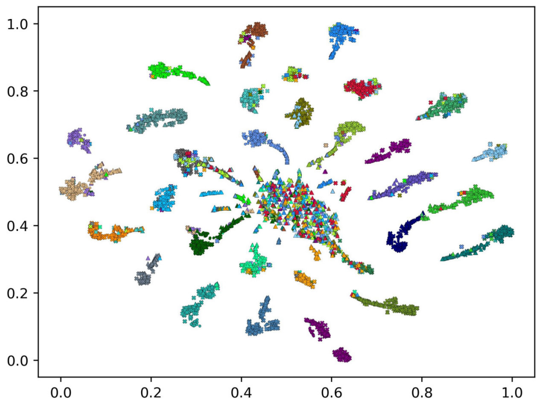
(d) Source Only.



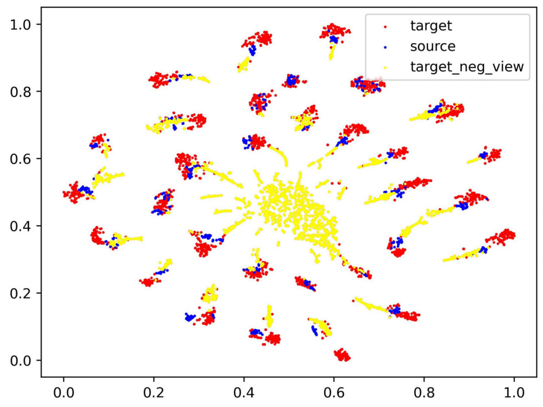
(e) SDAT.



(f) SDAT.



(g) SDAT+Ours.



(h) SDAT+Ours.

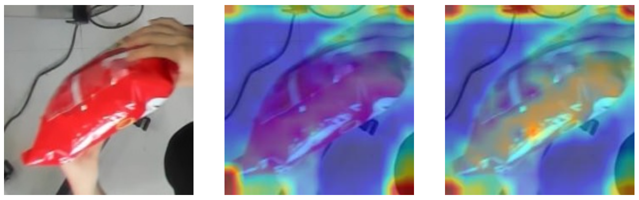
Figure 12. T-SNE of ViT-Base pretrained on ImageNet-1k, ViT-Base trained only with source-domain samples, and ViT-Base learned by SDAT and SDAT+Ours on 'W→A' scenario of Office-31.

Method	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg.
SDAT†	0.3044	0.6159	0.6374	0.4455	0.4875	0.5277	0.4014	0.2882	0.6252	0.4823	0.2834	0.6529	0.4793
SDAT†+**Ours**	0.0665	0.0966	0.0780	0.0763	0.0678	0.0716	0.0363	0.0514	0.0604	0.0404	0.0540	0.4649	0.0970

Table 14. Average Negative Confidence Score of ViT-Base trained on Office-Home for UDA. † indicates that it is reproduced with batch size 96. In ideal case, when the learned ViT receives negative views, it predicts the uniform distribution which has its length equal to the number of classes and in which each probability (*i.e.*, confidence score) is $1/(\# \text{ of classes}) = 1/65 = 0.0154$. Hence, the average negative confidence score of ground-truth class is ideally $1/(\# \text{ of classes}) = 1/65 = 0.0154$.

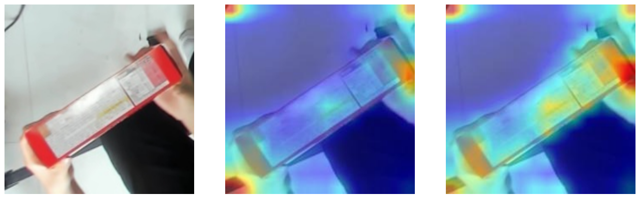
Method	Mean Confidence Score
SDAT†	0.5977
SDAT†+**Ours**	0.0852

Table 15. Mean Negative Confidence Score of ViT-Base trained on VisDA-2017 for UDA. † indicates that it is reproduced with batch size 96. In ideal case, when the learned ViT receives negative views, it predicts the uniform distribution which has its length equal to the number of classes and in which each probability (*i.e.*, confidence score) is $1/(\# \text{ of classes}) = 1/12 = 0.0833$. Hence, the mean negative confidence score of ground-truth class is ideally $1/(\# \text{ of classes}) = 1/12 = 0.0833$.



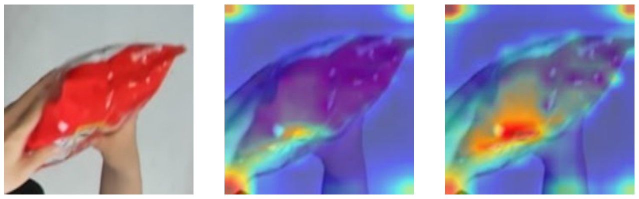
(a) Original Image Ground Truth: 016 (Lotte kkokkalcorn). (b) SDAT (Incorrect) Prediction: 011 (Noon-eul gamja). (c) Ours (Correct) Prediction: 016 (Lotte kkokkalcorn).

Figure 15. Visualization of attention maps from ViT-Tiny learned by SDAT and SDAT+Ours on Retail-71.



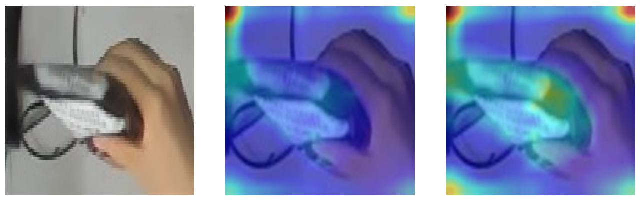
(a) Original Image Ground Truth: 001 (Fresh Berry Strawberry). (b) SDAT (Incorrect) Prediction: 004 (Pringles Original). (c) Ours (Correct) Prediction: 001 (Fresh Berry Strawberry).

Figure 13. Visualization of attention maps from ViT-Tiny learned by SDAT and SDAT+Ours on Retail-71.



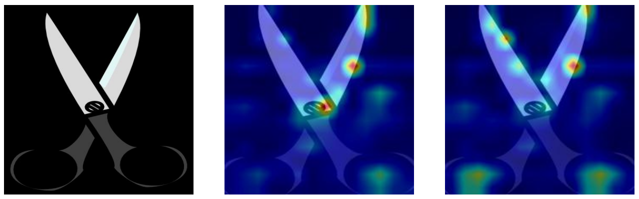
(a) Original Image Ground Truth: 016 (Lotte kkokkalcorn). (b) SDAT (Incorrect) Prediction: 011 (Noon-eul gamja). (c) Ours (Correct) Prediction: 016 (Lotte kkokkalcorn).

Figure 16. Visualization of attention maps from ViT-Tiny learned by SDAT and SDAT+Ours on Retail-71.



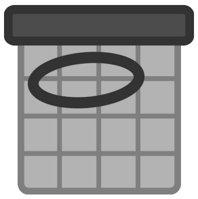
(a) Original Image Ground Truth: 019 (Choco pretzels). (b) SDAT (Incorrect) Prediction: 051 (Loacker wehaseu vanilla). (c) Ours (Correct) Prediction: 019 (Choco pretzels).

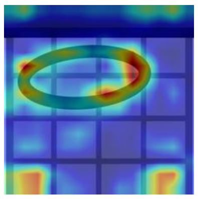
Figure 14. Visualization of attention maps from ViT-Tiny learned by SDAT and SDAT+Ours on Retail-71.



(a) Original Image Ground Truth: Scissors. (b) SDAT (Incorrect) Prediction: Knives. (c) Ours (Correct) Prediction: Scissors.

Figure 17. Visualization of attention maps from ViT-Base learned by SDAT and SDAT+Ours on a scenario Ar→Cl of Office-Home.


(a) Original Image Ground Truth: Calendar.


(b) SDAT (Incorrect) Prediction: Notebook.

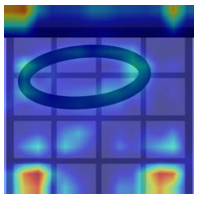
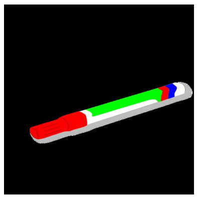
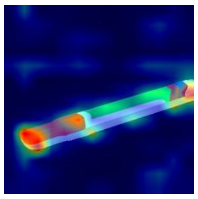

(c) Ours (Correct) Prediction: Calendar.

Figure 18. Visualization of attention maps from ViT-Base learned by SDAT and SDAT+Ours on a scenario Ar→Cl of Office-Home.


(a) Original Image Ground Truth: Marker.


(b) SDAT (Incorrect) Prediction: ToothBrush.

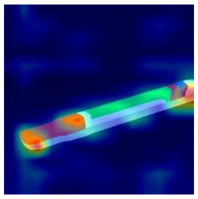
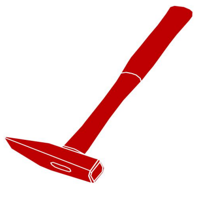
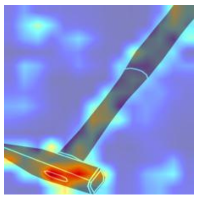

(c) Ours (Correct) Prediction: Marker.

Figure 19. Visualization of attention maps from ViT-Base learned by SDAT and SDAT+Ours on a scenario Ar→Cl of Office-Home.


(a) Original Image Ground Truth: Hammer.


(b) SDAT (Incorrect) Prediction: Mop.

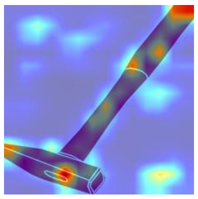

(c) Ours (Correct) Prediction: Hammer.

Figure 20. Visualization of attention maps from ViT-Base learned by SDAT and SDAT+Ours on a scenario Ar→Cl of Office-Home.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1

[2] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. 5

[3] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. 8

[4] Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 464–480. Springer, 2020. 5

[5] Fanqing Lin, Brian Price, and Tony Martinez. Ego2hands: A dataset for egocentric two-hand segmentation and detection, 2021. 3

[6] Mingsheng Long, ZHANGJIE CAO, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 5

[7] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017. 5

[8] Yao Qin, Chiyuan Zhang, Ting Chen, Balaji Lakshminarayanan, Alex Beutel, and Xuezhi Wang. Understanding and improving robustness of vision transformers through patch-based negative augmentation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 16276–16289. Curran Associates, Inc., 2022. 8, 9

[9] Harsh Rangwani, Sumukh K Aithal, Mayank Mishra, Arihant Jain, and Venkatesh Babu Radhakrishnan. A closer look at smoothness in domain adversarial training. In *International Conference on Machine Learning*, pages 18378–18399. PMLR, 2022. 5

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. 8

[11] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. volume 6314, pages 213–226, 09 2010. 5

[12] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017. 5