# Unsupervised Robust Domain Adaptation without Source Data
–Supplementary Material–

## 1. Overview

We present additional discussion and detailed analysis of our method along with minute implementation details. We first provide information on the choice of hyper-parameters in Section 2. In Section 3, we analyze the sensitivity of one of the hyper-parameter. We also study the impact of the quality of pseudo-label on model performance in Section 6. The results from a controlled experiment of pseudo-label quality are presented in Table 3. In Section 5, we comment on adversarial and clean accuracy for each of the tasks on all the four datasets. Insights into the network architecture has been provided in Section 7. We also analyze the impact of contrastive loss with the help of a toy-example in Section 8.

## 2. Hyperparameters

Our proposed models depend on multiple hyper-parameters, which have been set carefully. We keep the hyper-parameters fixed for all the datasets in all the experiments. The batch size is set to 64, and the learning rates for the bottleneck along with the classifier are both $10^{-3}$. The ResNet backbone is trained with a learning rate of $10^{-5}$ as proposed in [6]. The training is completed with the help of Adam optimizer without weight decay. Our methods is trained using a combination of four weighted loss terms. The diversity loss is has a weight $\alpha = 1$, the pseudo-label cross-entropy is weighted by $\beta = 0.3$ and the contrastive loss is multiplied with a factor of $\gamma = 0.2$. The values for $\alpha, \beta$ are borrowed from [6], and the sensitivity analysis for $\gamma$ is shown in the following section.

## 3. Hyper-parameter sensitivity

We perform the sensitivity analysis of our method with respect to $\gamma$ by evaluating the test accuracy on adversarial images. We conduct the experiment for all six domain adaptation tasks on the Office-31 dataset. Table 1 shows the variation in accuracy for different values of $\gamma$. We conduct another experiment on a relatively large dataset (PACS) with even more variation on the values of $\gamma$. Table 2 shows that, on average, the adversarial accuracy is not impacted much by the variation in $\gamma$. Both the experiments empirically verify that changing the weight of the contrastive loss does not have a significant impact on the robust average accuracy. Therefore, our method is robust against changes of hyperparameters, making the transfer between datasets easy.

## 4. Baselines

Source-free unsupervised domain adaptation has gained interest only recently. We are the first to introduce adversarial robustness in the context of domain adaptation in the absence of source data. Most of the previous methods [2, 3, 5] require target sample generation, which are difficult to implement on large scale datasets like VisDA-C and would many samples for datasets with many classes (Office-home). Moreover, one also needs to modify them to account for robustness. Recall that robust transfer requires the ResNet backbone to be pre-trained robustly on ImageNet. This would require huge computational resources to re-train for methods that use a modified architecture for the feature encoder. Some other methods [12, 9] only deal with pixel-level corruptions and do not provide clear guidelines for other domain shift problems, which we tackle in this work. Thus, we choose to compare against two baselines, namely, ADDA [10] and SHOT [6] in both of which we do not suffer from any issues described above. ADDA is one of the landmark papers in UDA, and SHOT is a very recent work that outperforms most all of the previous work, making it a strong baseline.

## 5. Detailed Results

We provide the performance of each of the baselines and our methods for individual domain adaptation tasks within datasets. Tables 4 to 7 clearly shows that our method is better not only on average but also on each of the tasks compared to both the baselines. We also evaluate each of the methods on clean samples and report the accuracies in Tables 9 to 12. We present the case of sub-setting only ten classes from the Office-home dataset in Table 8. It affirms our hypothesis that only a standard source model is sufficient to transfer robustly on the target if we have few classes in total. Sample images from each of the dataset are show in Figures 1, 2, 3 and 4.

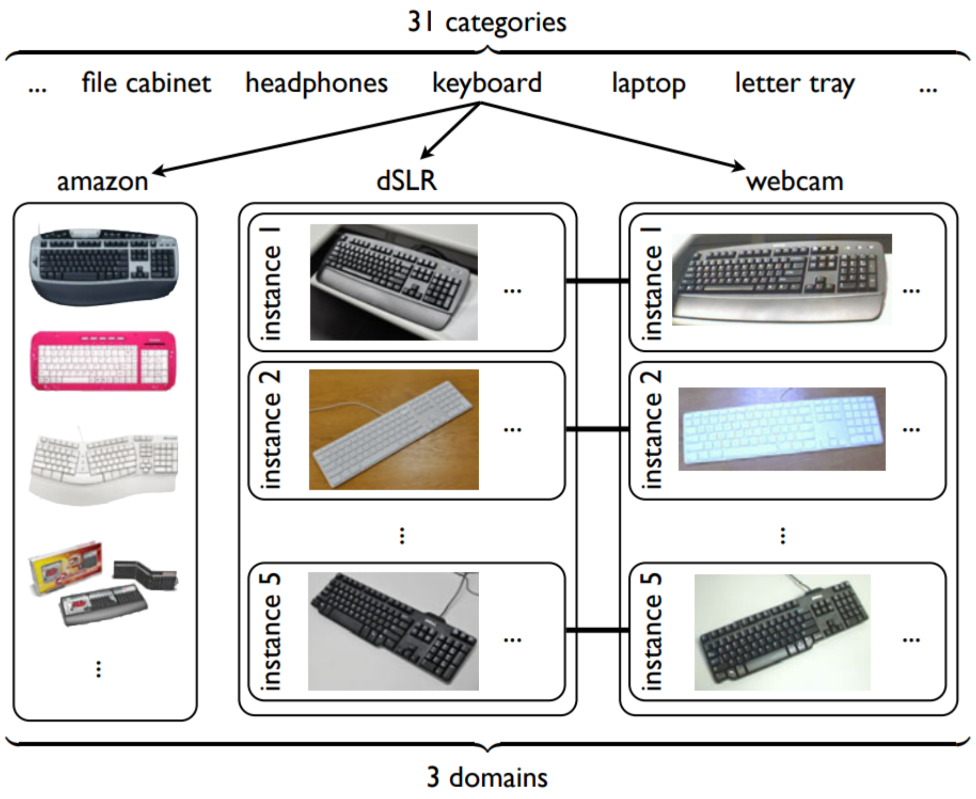


Figure 1: Sample images from the Office [8] dataset. Image courtesy [8]

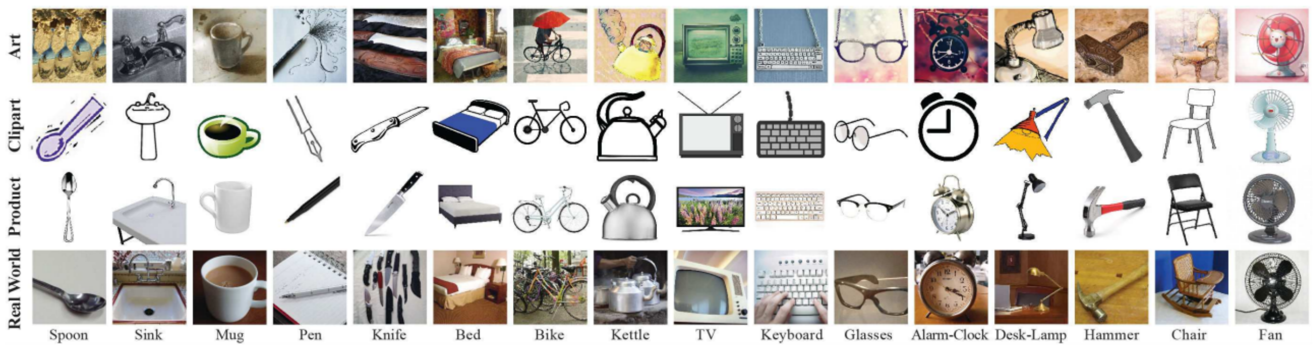


Figure 2: Sample images from the Office-home [11] dataset. Image courtesy [11]

## 6. Influence of Pseudo-labels Quality

We emphasize that pseudo-labels play a key role among other aspects in the proposed method. To further analyze the impact of pseudo-labels on the performance of our method, we conduct additional experiments. Recall that during the robust training on the target domain, we obtain pseudo-labels from the standard target model, which was trained in the previous step. To study their influence, we use the ground truth labels and create different sets of pseudo-labels by varying their accuracy with respect to the ground truth. We conduct the experiments on Office-31 datasets on all six domain adaptation tasks. Table 3 shows the results for 50%, 70% and 90% accurate pseudo-labels. Recall that pseudo-labels are required for three purposes, including contrastive loss, pseudo-label loss, and the generation of adversarial examples, each of with plays an important role. Thus, as we would expect, the accuracy improves with the quality of the pseudo-labels. Note that there is not only significant improvement in the average accuracy but also on each of the six tasks consistently, as shown in Table 3.

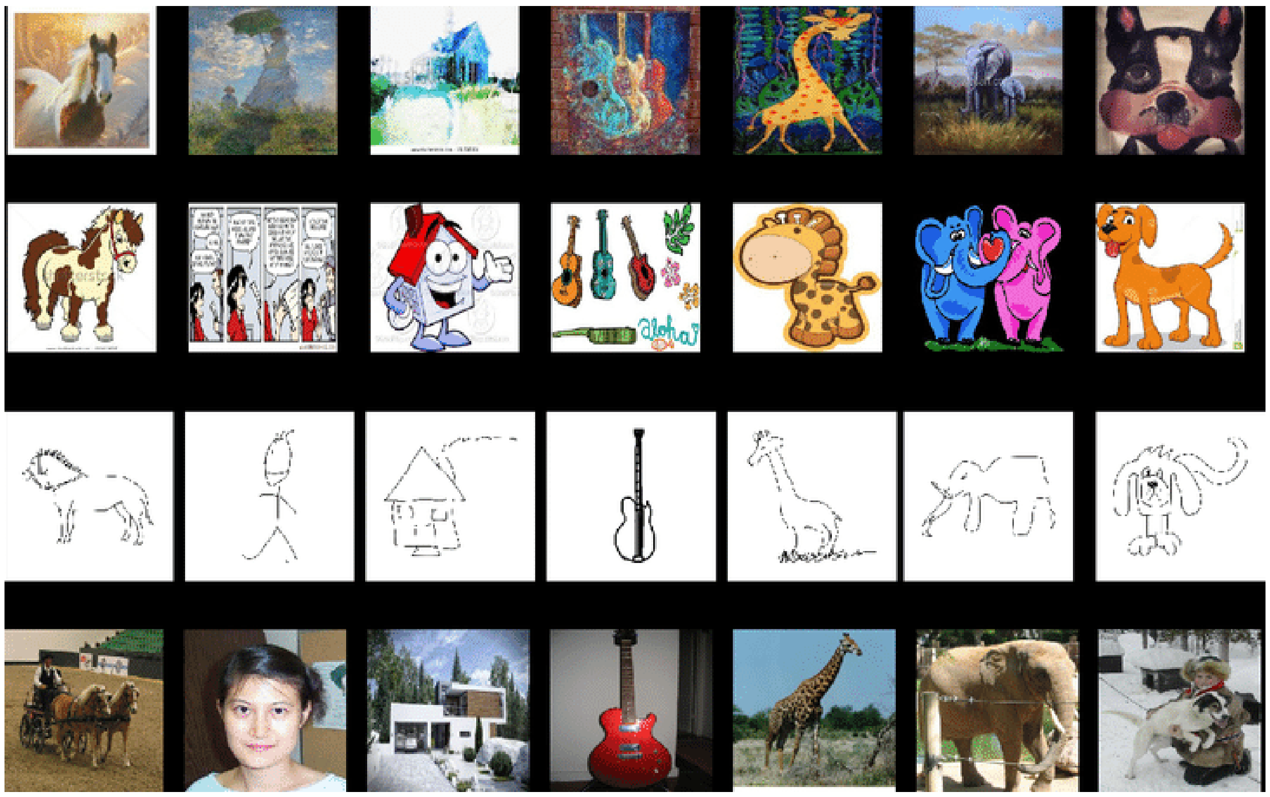


Figure 3: Sample images from the PACS [4] dataset. Image courtesy [4]

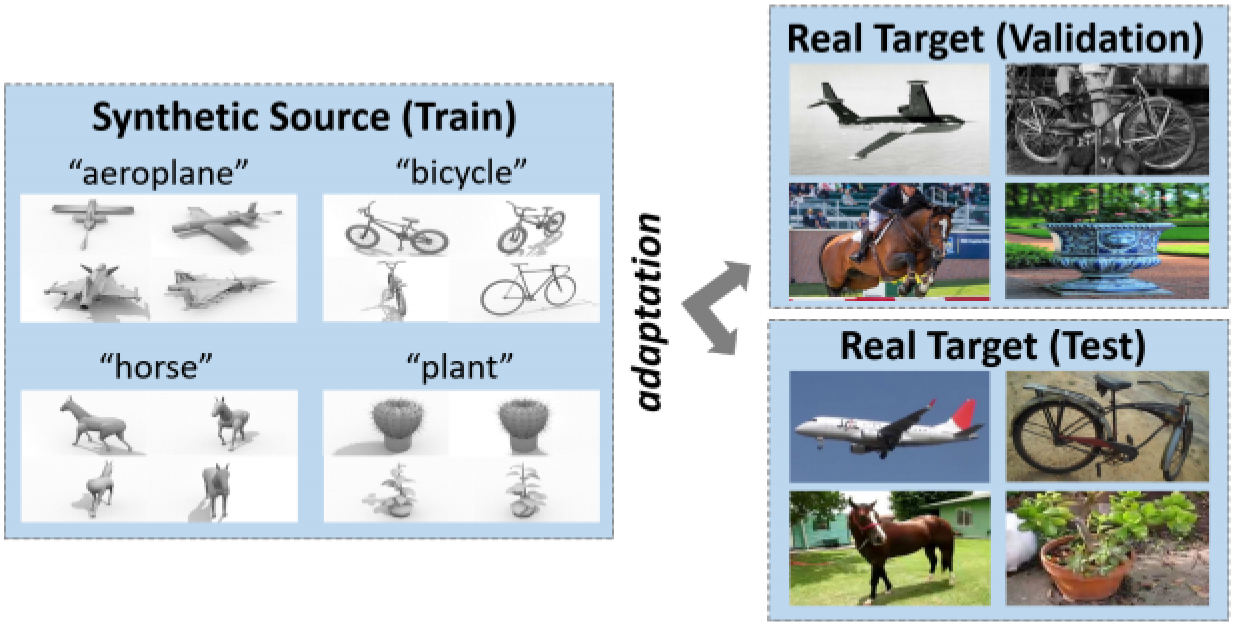


Figure 4: Sample images from the VisDA [7] dataset. Image courtesy [7]

| $\gamma$ | A → D | A → W | D → A | D → W | W → A | W → D | **Avg.** |
|---|---|---|---|---|---|---|---|
| 0.1 | 78.0 | 83.6 | 72.5 | **95.0** | 73.0 | 90.0 | 82.0 |
| 0.2 | 79.0 | 88.7 | 73.8 | 93.7 | 73.6 | **92.0** | 83.5 |
| 0.3 | **81.0** | **89.3** | **74.5** | **95.0** | **73.9** | **92.0** | 84.3 |

Table 1: Sensitivity with respect to hyper-parameter $\gamma$ on Office-31 dataset. The table indicates that in different weights to the contrastive loss can result in very similar performance.

| $\gamma$ | A → C | A → P | A → S | C → A | C → P | C → S | P → A | P → C | P → S | S → A | S → C | S → P | **Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.2 | 92.1 | 92.8 | **94.9** | 77.3 | **91.6** | **95.2** | 78.8 | **94.2** | 71.0 | **30.7** | **84.2** | 20.4 | 76.9 |
| 0.3 | 92.5 | 93.7 | 94.0 | **77.6** | **91.6** | 94.0 | **82.2** | 93.6 | 72.0 | 23.4 | 82.9 | 20.4 | 76.5 |
| 0.5 | **93.4** | **94.6** | 94.4 | **77.6** | 91.0 | 93.9 | 81.5 | 93.0 | **81.0** | 16.1 | 79.1 | **21.0** | 76.4 |

Table 2: Sensitivity with respect to hyper-parameter $\gamma$ on PACS dataset. We find that by modifying the weight of the contrastive loss the performance of our method does not change drastically.

## 7. Architecture

Figure 8 shows the network architecture we use in this work. The structure remains fixed in all the stages of the training in both the source and the target domain. The first block in Figure 8 refers to the ResNet50 backbone, which is trained on ImageNet in both standard and adversarial manner even before the source training. Following [1, 6] a feature bottleneck layer is introduced that consists of a linear (256 dimension) layer along with batch normalization. On top, we add a classification block containing a fully-connected layer and weight normalization as in [6] that outputs the class logits. Recall that the classification block is only trained during the source training and remains fixed during target training. However, the backbone and the bottleneck are tuned per the task. Note that even though we choose ResNet50 as the backbone, our method is not restrained by the backbone architecture.

## 8. Contrastive Loss

We make use of the contrastive loss for both standard and robust target training. Figure 5 shows a toy-example of the impact of contrastive loss. The circle positions represent images in the feature space, and the color represents its pseudo-label. We illustrate the pair selection in the feature space with the help of dotted lines. The idea of the contrastive loss is to pull together pair of points that belong to the same class and push apart others. In Figure 5 we show the change in relative distance of three pairs before (left) and after (right), minimizing the contrastive loss. Note that unlike other losses (e.g. Triplet loss), the contrastive loss does not require the additional cost of finding a suitable anchor. In our implementation, we apply contrastive loss on all possible pairs in a batch of 64 images simultaneously.

## 9. Feature Visualization

To understand the impact of pseudo-labels, we plot the encoder features with the help of PCA, followed by t-SNE. Figure 6 shows the adversarial images under three different scenarios. It clearly shows that cluster formation gets progressively better as we introduce pseudo-labels and switch from robust target model to standard target model for obtaining those pseudo-labels. We also plot the same scenarios but for clean samples instead in Figure 7. Thus, pseudo-labels not only help in improving the accuracy but also provide better clustering.

| Pseudo-label accuracy | A → D | A → W | D → A | D → W | W → A | W → D | **Avg.** |
|---|---|---|---|---|---|---|---|
| 50% | 69.0 | 76.1 | 64.7 | 88.1 | 68.6 | 89.0 | 75.9 |
| 70% | 76.0 | 87.4 | 76.4 | 93.1 | 72.5 | 89.0 | 82.4 |
| 90% | **79.0** | **91.8** | **79.3** | **97.5** | **80.0** | **93.0** | 86.8 |

Table 3: Impact of pseudo-label quality on Office-31. We can clearly observe that the "better" the pseudo-labels, the higher the accuracy on each of the tasks.

| 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.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDA robust | 22.3 | 18.2 | 34.2 | 21.4 | 34.9 | 35.0 | 16.0 | 26.5 | 40.4 | 28.4 | 36.9 | 47.2 | 30.1 |
| SHOT robust | 45.6 | 55.3 | 56.2 | 31.9 | 53.3 | 49.7 | 25.5 | 37.7 | 54.2 | 37.0 | 45.5 | 61.6 | 46.1 |
| Ours (Robust source) | 51.4 | 60.5 | 60.6 | 38.3 | 55.5 | 57.0 | 33.1 | 47.4 | 59.5 | **47.5** | 51.1 | 67.0 | 52.4 |
| Ours (Standard source) | 50.9 | 69.0 | 60.4 | 42.2 | 63.1 | 60.1 | 40.9 | 46.8 | 63.1 | 41.4 | 53.3 | 73.2 | 55.4 |
| Ours (Both) | **52.5** | **71.6** | **63.3** | **45.5** | **67.0** | **62.6** | **42.6** | **50.1** | **65.4** | 46.9 | **54.0** | **75.2** | **58.0** |

Table 4: Adversarial accuracy of robust models on Office-home. Our methods not only beat the baselines on average but also on each of the tasks. Also, our methods perform very similar to each other compared to the baselines.

| Method | A → D | A → W | D → A | D → W | W → A | W → D | **Avg.** |
|---|---|---|---|---|---|---|---|
| ADDA robust | 48.0 | 44.7 | 24.8 | 69.2 | 36.7 | 74.0 | 49.6 |
| SHOT robust | 61.0 | 64.2 | 52.5 | 90.6 | 52.0 | 87.0 | 67.9 |
| Ours (Robust source) | 64.0 | 66.7 | 62.2 | **95.0** | 59.2 | 90.0 | 72.8 |
| Ours (Standard source) | **80.0** | 86.2 | **74.5** | 90.6 | 71.1 | 85.0 | 81.2 |
| Ours (Both) | 79.0 | **88.7** | 73.8 | 93.7 | **73.6** | **92.0** | **83.5** |

Table 5: Adversarial accuracy of robust models on Office-31. Our methods not only beat the baselines on average but also on each of the tasks. Our method performs significantly better than both ADDA and SHOT on all of the tasks.

| Method | A → C | A → P | A → S | C → A | C → P | C → S | P → A | P → C | P → S | S → A | S → C | S → P | **Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDA robust | 63.8 | 65.0 | 5.3 | 55.1 | 65.0 | 8.9 | 50.2 | 58.2 | 10.7 | 32.7 | 52.9 | 31.4 | 41.6 |
| SHOT robust | 73.1 | 91.0 | 43.1 | 53.9 | 74.0 | 66.8 | 60.7 | 49.5 | 35.2 | 18.0 | 27.5 | 12.9 | 50.5 |
| Ours (Robust source) | 90.6 | **94.3** | 68.6 | 72.7 | 84.1 | 92.6 | 77.6 | 87.2 | 68.8 | 20.7 | 33.7 | 6.9 | 66.5 |
| Ours (Standard source) | **92.3** | 94.0 | 93.1 | 79.3 | **94.0** | 91.2 | **81.0** | 91.9 | 70.6 | **81.0** | **91.7** | **55.4** | **84.6** |
| Ours (Both) | 92.1 | 92.8 | **94.9** | 77.3 | 91.6 | **95.2** | 78.8 | **94.2** | **71.0** | 30.7 | 84.2 | 20.4 | 76.9 |

Table 6: Adversarial accuracy of robust models on PACS. The performance among three of our methods is close to each on relatively simpler (based on highest and lowest accuracy) tasks (e.g. A → C, A → P) but result into significantly different numbers on harder tasks (e.g. S → A, S → P).

| Method | plane | bcycl | bus | car | horse | knife | mcycl | person | plant | sktbrd | train | truck | **Per-class Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDA robust | 2.3 | 13.3 | 60.2 | 9.7 | 44.8 | **32.4** | 65.9 | 56.1 | 61.1 | 29.1 | 27.5 | 5.2 | 34.0 |
| SHOT robust | 48.0 | 26.4 | 23.9 | 26.3 | 33.9 | 5.7 | 33.4 | 10.7 | 16.5 | 15.5 | 47.8 | 11.4 | 25.0 |
| Ours (Robust source) | 78.3 | 42.7 | 61.1 | 60.0 | 70.2 | 4.4 | 62.1 | 55.4 | 54.9 | 31.1 | 78.0 | 19.0 | 51.4 |
| Ours (Standard source) | **88.0** | 56.9 | **71.0** | **68.9** | **83.3** | 0.5 | **80.6** | **68.7** | **79.8** | **82.1** | **82.7** | **38.2** | **66.7** |
| Ours (Both) | 86.1 | **61.8** | 69.4 | 67.5 | 82.8 | 1.1 | 76.3 | 67.4 | 75.3 | 79.7 | 80.8 | 31.5 | 65.0 |

Table 7: Adversarial accuracy of robust models on VisDA-C (Synthetic to Real). There is large variation in performance among different methods due to increased difficulty. Ours (standard source) performs the best on most of the classes.

| 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.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDA robust | 36.8 | 32.8 | 54.2 | 48.6 | 52.7 | 71.1 | 35.2 | 47.7 | 72.3 | 63.8 | 65.2 | 70.2 | 54.2 |
| SHOT robust | 69.0 | 82.4 | 91.6 | 61.9 | 75.6 | 91.0 | 52.4 | 61.3 | 88.0 | 66.7 | 65.2 | 87.0 | 74.3 |
| Ours (Robust source) | 89.0 | 87.8 | **95.2** | 74.3 | 90.8 | **95.8** | 69.5 | 79.4 | 93.4 | 75.2 | 85.8 | 93.1 | 85.8 |
| Ours (Standard source) | 89.0 | 93.1 | **95.2** | **78.1** | 91.6 | 95.2 | **78.1** | **91.6** | **95.2** | 76.2 | **91.6** | 93.1 | **89.0** |
| Ours (Both) | **91.0** | **95.4** | **95.2** | 74.3 | **94.7** | 94.6 | 74.3 | 87.1 | 94.6 | **77.1** | **91.6** | **94.7** | 88.7 |

Table 8: Adversarial accuracy of robust models on Office-home with only **10 classes**. Unlike the data with all 65 classes, here ours (standard source) better on average than ours (both) and is comparable on most tasks.

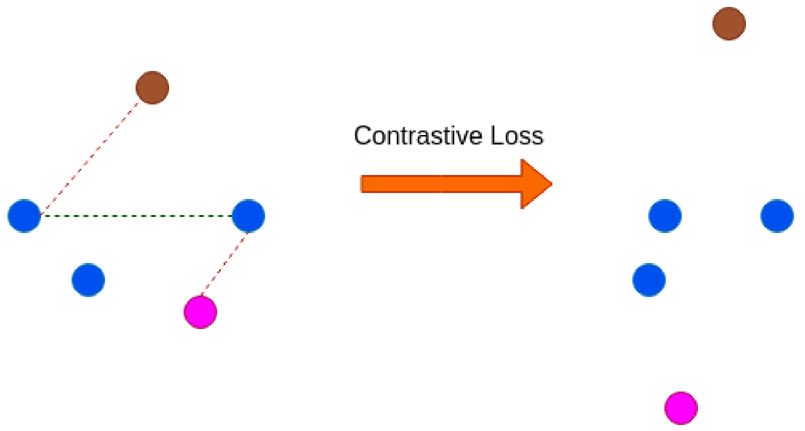


Figure 5: Depiction of contrastive loss in the feature space. On the left, the positive (same class) and negative (different class) pair of images are connected with a green and a red dotted line, respectively. The impact of the loss is shown on the right.

| 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.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDA robust | 26.8 | 23.1 | 39.3 | 30.2 | 41.9 | 44.2 | 24.9 | 31.7 | 50.1 | 42.0 | 43.4 | 54.5 | 37.7 |
| SHOT robust | 48.2 | 61.5 | 64.0 | 38.7 | 57.8 | 59.3 | 34.6 | 42.2 | 62.7 | 51.9 | 49.7 | 66.8 | 53.1 |
| Ours (Robust source) | 54.6 | 65.3 | 69.5 | 50.4 | 59.8 | 65.3 | 42.8 | 51.1 | 69.3 | 57.0 | 54.3 | 71.4 | 59.2 |
| Ours (Standard source) | 53.5 | 73.6 | 71.6 | 52.1 | 69.3 | 69.5 | 52.3 | 50.6 | 73.7 | 51.6 | 56.9 | 78.2 | 62.7 |
| Ours (Both) | **54.9** | **74.9** | **73.1** | **56.4** | **72.1** | **71.9** | **54.1** | **53.4** | **75.2** | **57.4** | **58.5** | **79.8** | **65.1** |

Table 9: Clean accuracy of robust models on Office-home. Ours (both) method beats all others not just on average but on all the tasks.

| Method | A → D | A → W | D → A | D → W | W → A | W → D | **Avg.** |
|---|---|---|---|---|---|---|---|
| ADDA robust | 60.0 | 59.7 | 28.9 | 74.2 | 40.8 | 82.0 | 57.6 |
| SHOT robust | 66.0 | 68.6 | 57.4 | 93.1 | 59.4 | **94.0** | 73.1 |
| Ours (Robust source) | 70.0 | 68.6 | 67.0 | **95.6** | 64.4 | 93.0 | 76.4 |
| Ours (Standard source) | 82.0 | 91.8 | **78.4** | 95.0 | 76.1 | 91.0 | 85.7 |
| Ours (Both) | **84.0** | **92.5** | 78.0 | **95.6** | **77.7** | **94.0** | **87.0** |

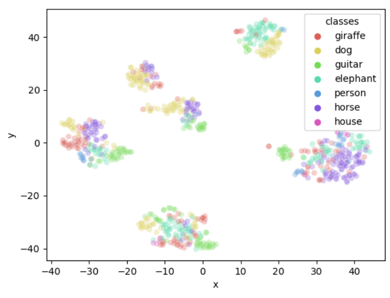
Table 10: Clean accuracy of robust models on Office-31. Our methods beats the baselines not just on average but also on most of the tasks.

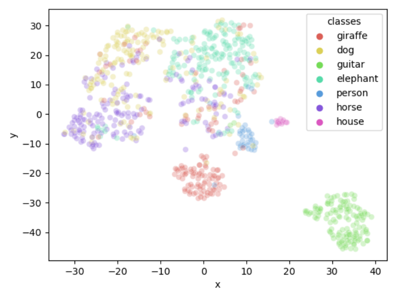
| Method | A → C | A → P | A → S | C → A | C → P | C → S | P → A | P → C | P → S | S → A | S → C | S → P | **Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDA robust | 72.7 | 79.6 | 37.8 | 74.1 | 77.2 | 51.4 | 65.6 | 65.9 | 50.8 | 44.4 | 58.8 | 39.5 | 59.8 |
| SHOT robust | 78.3 | 95.2 | 46.6 | 68.0 | 82.6 | 70.5 | 73.2 | 57.6 | 37.3 | 26.3 | 33.9 | 18.0 | 57.3 |
| Ours (Robust source) | 94.7 | **99.1** | 69.5 | 84.4 | 94.3 | 95.0 | 91.7 | 93.0 | 70.0 | 26.6 | 42.2 | 12.0 | 72.7 |
| Ours (Standard source) | 95.1 | 97.9 | 94.1 | **92.0** | **97.9** | 92.1 | 92.4 | 94.7 | 71.5 | **92.4** | **95.1** | **58.1** | **89.4** |
| Ours (Both) | **95.9** | **99.1** | **96.2** | 88.8 | **97.9** | **96.2** | **93.2** | **96.2** | **72.1** | 45.9 | 91.0 | 30.2 | 83.6 |

Table 11: Clean accuracy of robust models on PACS. Similar to adversarial accuracy (see Table 6) we find that ours (standard source) significantly outperforms others on relatively difficult tasks (e.g. S → A, S → P).

| Method | plane | bcycl | bus | car | horse | knife | mcycl | person | plant | sktbrd | train | truck | **Per-class Avg.** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDA robust | 2.9 | 26.4 | 77.2 | 24.3 | 59.1 | **42.3** | 78.6 | 66.8 | 72.3 | 35.8 | 48.6 | 21.0 | 46.3 |
| SHOT robust | 59.5 | 36.7 | 33.8 | 35.9 | 44.6 | 6.4 | 47.0 | 18.3 | 25.1 | 26.3 | 59.5 | 18.7 | 34.3 |
| Ours (Robust source) | 88.0 | 59.4 | 74.0 | 71.3 | 83.5 | 6.0 | 77.1 | 69.8 | 67.1 | 47.7 | 85.0 | 34.8 | 63.6 |
| Ours (Standard source) | **93.9** | 70.5 | **81.8** | **78.6** | 91.4 | 1.4 | **90.3** | **78.0** | **89.9** | **91.4** | **90.0** | **52.8** | **75.8** |
| Ours (Both) | 92.5 | **75.3** | 79.9 | 77.5 | **91.5** | 3.0 | 87.9 | 77.0 | 87.5 | 89.2 | 89.5 | 47.5 | 74.9 |

Table 12: Clean accuracy of robust models on VisDA. Performances of ours (Standard source) and ours (Both) are very close though ours (Standard source) does better on average.


(a) No pseudo-labels


(b) Ours (Robust source)

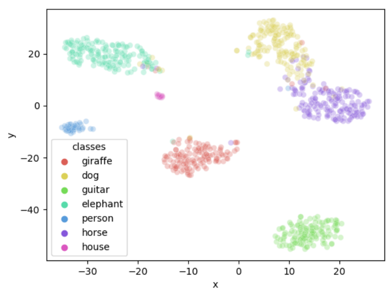
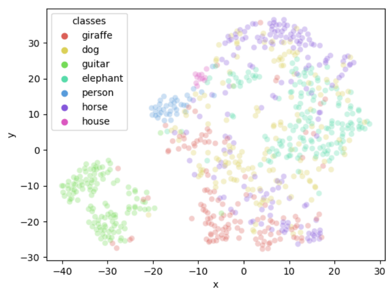
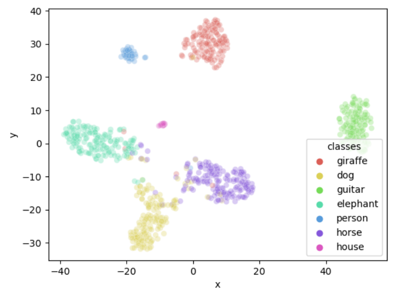

(c) Ours (standard source)

Figure 6: Impact of pseudo-labels on PACS with Cartoon as the source and Sketch as the target domain for **adversarial images**. The plot clear shows the importance of pseudo-labels. (a) Without the use of pseudo-labels the clusters are all mixed. (b) With pseudo-labels from robust model, some clusters tend to overlap. (c) With pseudo-labels from standard source model, well separated clusters are formed.


(a) No pseudo-labels


(b) Ours (Robust source)

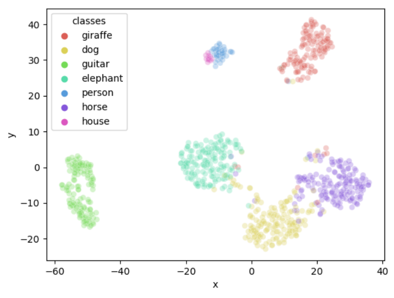

(c) Ours (Standard source)

Figure 7: Impact of pseudo-labels on PACS with Cartoon as the source and Sketch as the target domain for **clean images**. (a) Without the use of pseudo-labels, features of different classes tend to overlap. (b) With pseudo-labels, points features tend to cluster with little difference with respect to the model being robust or not.

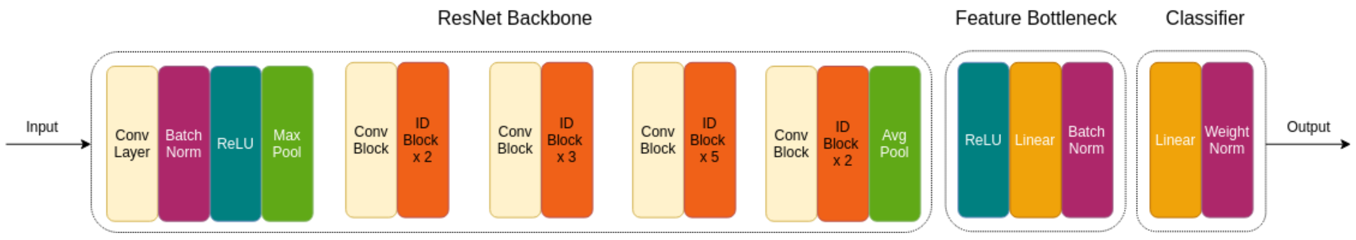


Figure 8: Detailed architecture of the network used in all the experiments. The network is divided into three blocks, namely, the backbone, the feature bottleneck and the classifier. The backbone together with the bottleneck forms the feature encoder.

## References

[1] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015. 4

[2] Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4544–4553, 2020. 1

[3] Vinod K Kurmi, Venkatesh K Subramanian, and Vinay P Namboodiri. Domain impression: A source data free domain adaptation method. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 615–625, 2021. 1

[4] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017. 3

[5] Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9650, 2020. 1

[6] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020. 1, 4

[7] X. Peng, B. Usman, N. Kaushik, D. Wang, J. Hoffman, and K. Saenko. Visda: A synthetic-to-real benchmark for visual domain adaptation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2102–21025, 2018. 3

[8] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 2

[9] Roshni Sahoo, Divya Shanmugam, and John Guttag. Unsupervised domain adaptation in the absence of source data. *arXiv preprint arXiv:2007.10233*, 2020. 1

[10] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017. 1

[11] 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. 2

[12] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. 1