Surprisingly Simple Semi-Supervised Domain Adaptation with Pretraining and Consistency

Samarth Mishra[1] **Kate Saenko**[1,2] **Venkatesh Saligrama**[1]
[1]Boston University [2]MIT-IBM Watson AI Lab
{samarthm, saenko, srv}@bu.edu

Abstract

Most modern unsupervised domain adaptation (UDA) approaches are rooted in domain alignment, *i.e.*, learning to align source and target features to learn a target domain classifier using source labels. In semi-supervised domain adaptation (SSDA), when the learner can access few target domain labels, prior approaches have followed UDA theory to use domain alignment for learning. We show that the case of SSDA is different and a good target classifier can be learned without needing explicit alignment. We use self-supervised pretraining and consistency regularization to achieve well separated target clusters, aiding in learning a low error target classifier, allowing our method to outperform recent state of the art approaches on large, challenging benchmarks like DomainNet and VisDA-17. Code for our experiments can be found at https://github.com/venkatesh-saligrama/PAC.

1 Introduction

The problem of visual domain adaptation arises when a learner must leverage labeled source domain data to classify instances in the target domain, where it has limited access to ground-truth annotated labels. An example of this is the problem of learning to classify real-world images based on hand-sketched depictions. The problem is challenging because discriminative features that are learnt while training to classify source domain instances may not be meaningful or sufficiently discriminative in the target domain. As described in prior works, this situation can be viewed as arising from a "domain-shift", where the joint distribution of features and labels in the source domain does not follow the same law in the target domain.

We propose a novel method for semi-supervised domain adaptation (SSDA), where the learner, in addition to unlabeled target examples, is granted access to a few labeled target domain examples for training. Our method is based on enhancing clusterability of target domain features independent of the source domain. Prior SSDA methods [28, 15, 16] draw upon approaches developed within the context of visual domain adaptation using labelled source images and unlabelled target images [1, 21, 30, 34, 38]. These works in turn are based on adversarial domain alignment, and draw upon Ben-David *et al*. [2]'s elegant theory of domain adaptation. Generalization bounds in Ben-David *et al*. [2] suggest that, good adaptation is possible if the divergence between induced source and target domain feature distributions is small.

While domain alignment is a meaningful goal in the absence of labels, we believe that the situation is dramatically different in the presence of a few target domain labels. For our work, we draw inspiration from Castelli and Cover [5], who showed that in binary classification problems, a learner with no knowledge of underlying feature distributions, can benefit exponentially from labelled examples. Specifically, with u unlabelled examples and ℓ labelled ones, they showed the learners probability of error approaches Bayes error as

$$P_{err} - P_{bayes} = O\left(\frac{1}{u}\right) + \exp\left(-D(q_1, q_2)\ell + o(\ell)\right) \tag{1}$$

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

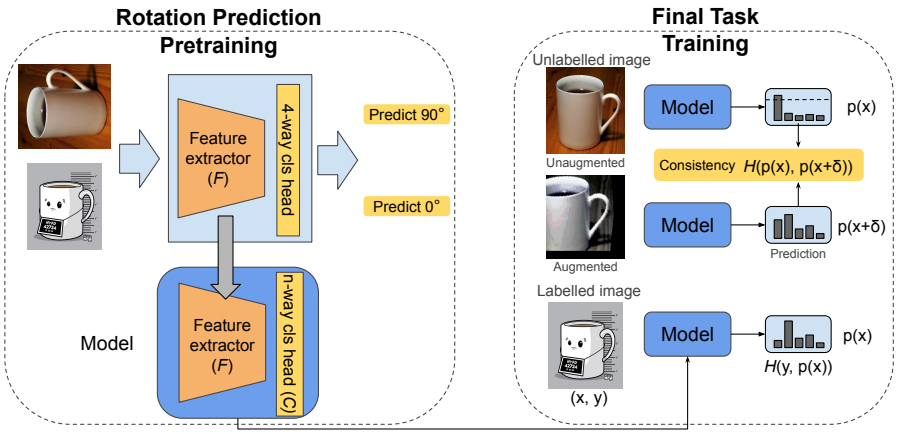

Figure 1: A diagram of our Pretraining and Consistency (PAC) approach. We first train our backbone for the self-supervised task of predicting rotations (left). This backbone is then used as a warm start for our classification model, which uses labelled data with the cross entropy criterion and consistency regularization for the unlabelled data (right).

where $D(q_1, q_2)$ or the Bhattacharya distance is a measure of separation between the class distributions q_1 and q_2. Notice that the error probability decays exponentially with the product of inter-class distance and number of labels. As such, our key insight is that we can forego domain alignment if we learn representations that result in compact, well-separated clusters in the target domain. The identity of these clusters can then be deduced from few labels.

Our contributions in this paper are two-fold: (1) We propose a novel semi-supervised domain adaptation method PAC (pretraining and consistency), based on label consistency and rotation prediction for pretraining, which performs comparably or better than state of the art on SSDA across multiple datasets. In contrast to prior works, we forego domain alignment, and pose objectives that improve target clusterability. (2) We perform ablative analysis on individual components of our method, illustrating their behavior, and their impact on performance. Our analysis provides an understanding of these components and shows how they can be combined with other techniques.

2 Pretraining and Consistency (PAC)

Notating some variables: Available to the model are two sets of labelled images : $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$, the labelled source images and $\mathcal{D}_t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t}$, the few labelled target images, and additionally the set of unlabelled target images $\mathcal{D}_u = \{x_i^u\}_{i=1}^{n_u}$. The goal is to predict labels for these images in $\mathcal{D}_u$. The final classification model consists of two components : the feature extractor F and the classifier C. F generates features $F(\boldsymbol{x})$ for an input image $\boldsymbol{x}$ which the classifier uses produce output class scores $C(F(\boldsymbol{x})) \in \mathbb{R}^K$, where K is the number of categories that the images in the dataset could belong to. In our experiments, F is a convolutional network and produces features with unit ℓ_2-norm, *i.e.* $||F(x)||_2 = 1$ (following [28]). C consists of one or two fully connected layers.

An overview of PAC is shown in Fig 1. Our final model is trained in two stages:

Pretraining with Rotation Prediction. We first train our feature extractor F with the self-supervised task of predicting image rotations (Fig 1 (left)) on both the source and target datasets, *i.e.*, all images in $\mathcal{D}_s, \mathcal{D}_t$ and $\mathcal{D}_u$. Without using image category labels, we train a 4-way classifier to predict one out of 4 possible angles ($0°$, $90°$, $180°$, $270°$) that an input image has been rotated by. This backbone is then used as the initialization for training the final classifier in the next stage.

Consistency Regularized Classifier Training. Consistency regularization promotes the final model $C \circ F$ to produce the same output for both an input image $\boldsymbol{x}$ and a perturbed version $\boldsymbol{x} + \delta$. We introduce these perturbations using image level augmentations — RandAugment [8] along with additional color jittering. Given an unlabelled image $\boldsymbol{x} \in \mathcal{D}_u$, we first compute the model's predicted class distributions

$$p_x = p(y|\boldsymbol{x}; F, C) = \text{softmax}(C(F(\boldsymbol{x})))$$
$$q_x = p(y|\boldsymbol{x} + \delta; F, C) = \text{softmax}(C(F(\boldsymbol{x} + \delta)))$$

p_x is then confidence thresholded using a threshold τ and the following is used as the consistency regularization loss.

$$\mathcal{L}_{CR}(\boldsymbol{x}) = \mathbb{1}[\max_{k \in K} p_x(k) \geq \tau] H(p_x, q_x) \tag{2}$$

where $\mathbb{1}$ is an indicator function and $H(p_x, q_x) = \sum_{k \in K} -p_x(k) \log(q_x(k))$ is cross-entropy. Note that $p_x(\cdot)$ has been used to index into the K elements of p_x. Intuitively, an unperturbed version of image $\boldsymbol{x}$ is used to compute *pseudo-targets* for the perturbed version $\boldsymbol{x} + \delta$, which is only used when the pseudo-target has high confidence ($\max_k p_x(k) \geq \tau$). We also note here that the target p_x is treated as a constant for gradient computation with respect to the network parameters. For the labelled examples from $\mathcal{D}_s$ and $\mathcal{D}_t$, we use the same perturbations but with ground truth labels as targets.

The model is optimized using minibatch-SGD, with minibatches M_s, M_t and M_u sampled from $\mathcal{D}_s, \mathcal{D}_t$ and $\mathcal{D}_u$ respectively. The final optimization criterion used is

$$\mathcal{L} = \frac{1}{|M_s|} \sum_{(\boldsymbol{x},y) \in M_s} H(\bar{y}, \boldsymbol{x}) + \frac{1}{|M_t|} \sum_{(\boldsymbol{x},y) \in M_t} H(\bar{y}, \boldsymbol{x}) + \frac{1}{|M_u|} \sum_{\boldsymbol{x} \in M_u} \mathcal{L}_{CR}(\boldsymbol{x})$$

where $\bar{y} \in \mathbb{R}^K$ is the one-hot representation of $y \in [K]$ or $\bar{y}(i) = \mathbb{1}[i = y]$ and $H(\bar{y}, \boldsymbol{x})$ has been overloaded to mean $H(\bar{y}, C(F(\boldsymbol{x})))$.

3 Experiments

Datasets. From DomainNet[25], we used 4 domains (Clipart, Paintings, Real and Sketch) following [28], each with 126 classes resulting in an avg 36500 images per domain. In VisDA-17 [26] the source domain (the VisDA "train" split) consists of 152,398 synthetic images from 12 categories, and the target domain (the VisDA "validation" split) consists of 55,388 real images. We used 1 and 3-shot settings for evaluation, which contained 1 and 3 labelled examples in the target domain. Additionally, for VisDA-17, we evaluated with settings where 1 and 5% of the target examples were labelled.

Implementation Details. All our experiments were implemented in PyTorch [24] using W&B [4] for tracking experiments. For evaluations on the DomainNet dataset, we used an Alexnet and a Resnet-34 [14] backbone, while on VisDA-17 we evaluated our method with a ResNet-34 backbone. 1 or 2 fully connected layers we used for the classifier C. Optimization was done via SGD with a momentum 0.9. Complete details of all experiments are in the supplementary.

3.1 Results

Comparison to other approaches. We compare PAC with different recent SSDA approaches : MME [28], BiAT [15], Meta-MME [19], APE [16], LIRR [18], and CDAC [20] using results reported by these papers. Besides this, we also include in the tables, baseline approaches using adversarial domain alignment—DANN [11], ADR [29] and CDAN [21], that were evaluated by Saito *et al*. [28]. The baseline "S+T" is a method that simply uses all labelled data available to it to train the network using cross-entropy loss.

In Table 1, we compare the accuracy of PAC with different recent approaches on DomainNet. Remarkably our simple approach outperforms other SSDA approaches by 3-5% on this benchmark with different backbones (with the exception of CDAC, with which it is competitive on most scenarios). In Table 2, besides our method, we report results of S+T and MME on VisDA-17, that we replicated from the implementation of [28]. We see that PAC shows strong performance, with close to 10% improvement in accuracy over MME in the 3-shot scenario, and 4% improvement over LIRR.

Ablative analysis. In Table 3, we see what rotation prediction pretraining and consistency regularization do for final target classification performance separately. The two components provide boosts to the final performance individually, with the combination of both performing best. We see that in most cases consistency regularization helps performance significantly, especially in the 3-shot scenarios.

For more analyses including a feature space comparison of methods, effects of different pretraining and perturbation methods and PAC's sensitivity to different number of target examples, see the supplementary.

Net	Method	R to C		R to P		P to C		C to S		S to P		R to S		P to R		MEAN	
		1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot
Alexnet	S+T	43.3	47.1	42.4	45.0	40.1	44.9	33.6	36.4	35.7	38.4	29.1	33.3	55.8	58.7	40.0	43.4
	DANN	43.3	46.1	41.6	43.8	39.1	41.0	35.9	36.5	36.9	38.9	32.5	33.4	53.6	57.3	40.4	42.4
	ADR	43.1	46.2	41.4	44.4	39.3	43.6	32.8	36.4	33.1	38.9	29.1	32.4	55.9	57.3	39.2	42.7
	CDAN	46.3	46.8	45.7	45.0	38.3	42.3	27.5	29.5	30.2	33.7	28.8	31.3	56.7	58.7	39.1	41.0
	MME	48.9	55.6	48.0	49.0	46.7	51.7	36.3	39.4	39.4	43.0	33.3	37.9	56.8	60.7	44.2	48.2
	Meta-MME	-	56.4	-	50.2	-	51.9	-	39.6	-	43.7	-	38.7	-	60.7	-	48.7
	APE	47.7	54.6	49.0	50.5	46.9	52.1	38.5	42.6	38.5	42.2	33.8	38.7	**57.5**	**61.4**	44.6	48.9
	BiAT	54.2	58.6	49.2	50.6	44.0	52.0	37.7	41.9	**39.6**	42.1	37.2	42.0	56.9	58.8	45.5	49.4
	CDAC	**56.9**	**61.4**	**55.9**	**57.5**	**51.6**	**58.9**	**44.8**	**50.7**	**48.1**	**51.7**	**44.1**	**46.7**	**63.8**	**66.8**	**52.1**	**56.2**
	PAC	**55.4**	**61.7**	**54.6**	**56.9**	**47.0**	**59.8**	**46.9**	**52.9**	38.6	**43.9**	**38.7**	**48.2**	56.7	59.7	**48.3**	**54.7**
Resnet-34	S+T	55.6	60.0	60.6	62.2	56.8	59.4	50.8	55.0	56.0	59.5	46.3	50.1	71.8	73.9	56.8	60.0
	DANN	58.2	59.8	61.4	62.8	56.3	59.6	52.8	55.4	57.4	59.9	52.2	54.9	70.3	72.2	58.4	60.7
	ADR	57.1	60.7	61.3	61.9	57.0	60.7	51.0	54.4	56.0	59.9	49.0	51.1	72.0	74.2	57.6	60.4
	CDAN	65.0	69.0	64.9	67.3	63.7	68.4	53.1	57.8	63.4	65.3	54.5	59.0	73.2	78.5	62.5	66.5
	MME	70.0	72.2	67.7	69.7	69.0	71.7	56.3	61.8	64.8	66.8	61.0	61.9	76.1	78.5	66.4	68.9
	Meta-MME	-	73.5	-	70.3	-	72.8	-	62.8	-	68.0	-	63.8	-	79.2	-	70.1
	APE	70.4	76.6	70.8	72.1	**72.9**	**76.7**	56.7	63.1	64.5	66.1	63.0	67.8	76.6	**79.4**	67.8	71.7
	BiAT	73.0	74.9	68.0	68.8	71.6	74.6	57.9	61.5	63.9	67.5	58.5	62.1	**77.0**	78.6	67.1	69.7
	CDAC	**77.4**	**79.6**	**74.2**	**75.1**	**75.5**	**79.3**	**67.6**	**69.9**	**71.0**	**73.4**	**69.2**	**72.5**	**80.4**	**81.9**	**73.6**	**76.0**
	PAC	**74.9**	**78.6**	**73.0**	**74.3**	72.6	76.0	**65.8**	**69.6**	**67.9**	**69.4**	**68.7**	**70.2**	76.7	79.3	**71.4**	**73.9**

Table 1: Accuracy on the DomainNet dataset (%) for one-shot and three-shot settings on 4 domains, R: Real, C: Clipart, P: Painting, S: Sketch. PAC, though simple, is strong enough to be competitive with or outperform other state of the art approaches on most scenarios. Top 2 accuracies in each column are highlighted in bold

Method	Overall Accuracy			
	1-shot	3-shot	1-pct	5-pct
S+T	57.7	59.9	76.2	82.9
MME	69.7	70.7	80.5	84.1
LIRR	-	-	81.7	84.5
LIRR+CosC	-	-	82.3	85.1
PAC	**75.2**	**80.4**	**86.0**	**88.9**

Table 2: Results on VisDA-17. PAC outperforms MME and LIRR, both current state of the art approaches, by a sizeable margin. CosC stands for cosine classifier and "1-pct" and "5-pct" denote scenarios where 1% and 5% of the target images are labelled respectively.

Rot^n	CR	Target Accuracy			
		Alexnet		Resnet-34	
		1-shot	3-shot	1-shot	3-shot
		29.1	33.3	46.3	50.1
✓		35.1	37.9	54.1	56.1
	✓	32.5	45.9	64.3	68.9
✓	✓	38.7	48.2	68.7	70.2

Table 3: Ablation study for pretraining predicting rotations (Rot^n) and consistency regularization (CR) on the *real* to *sketch* scenario of Domainnet using both Alexnet and Resnet-34 backbones.

4 Conclusion

We showed that consistency regularization and pretraining using rotation prediction are powerful techniques in SSDA. Our method, using simply a combination of these without requiring any domain alignment, could outperform recent state of the art on this task, most of which use adversarial alignment. With our approach we demonstrated that domain alignment is not a necessity for SSDA, and that achieving well-separated target clusters allows for low classifier error with a few labelled examples. We presented a thorough analysis of both the aforementioned techniques showing how they can improve target clustering and why they are better than other options for similar approaches. We hope our analysis can help inform future SSDA work.

Acknowledgements. This work was supported by the Hariri Institute at Boston University.

References

[1] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[3] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137–144, 2006.

[4] Lukas Biewald. Experiment tracking with weights and biases, 2020.

[5] Vittorio Castelli and Thomas M Cover. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Transactions on information theory*, 42(6):2102–2117, 1996.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[7] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

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

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[12] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Pin Jiang, Aming Wu, Yahong Han, Yunfeng Shao, Meiyu Qi, and Bingshuai Li. Bidirectional adversarial training for semi-supervised domain adaptation.

[16] Taekyung Kim and Changick Kim. Attract, perturb, and explore: Learning a feature alignment network for semi-supervised domain adaptation. *arXiv preprint arXiv:2007.09375*, 2020.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Bo Li, Yezhen Wang, Shanghang Zhang, Dongsheng Li, Kurt Keutzer, Trevor Darrell, and Han Zhao. Learning invariant representations and risks for semi-supervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1104–1113, 2021.

[19] Da Li and Timothy Hospedales. Online meta-learning for multi-source and semi-supervised domain adaptation. *arXiv preprint arXiv:2004.04398*, 2020.

[20] Jichang Li, Guanbin Li, Yemin Shi, and Yizhou Yu. Cross-domain adaptive clustering for semi-supervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2505–2514, 2021.

[21] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018.

[22] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[23] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[25] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[26] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017.

[27] 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.

[28] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8050–8058, 2019.

[29] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial dropout regularization. In *International Conference on Learning Representations*, 2018.

[30] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[31] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[33] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019.

[34] 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.

[35] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *(IEEE) Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] Wikipedia contributors. Bhattacharyya distance, 2020.

[37] Jiaolong Xu, Liang Xiao, and Antonio M López. Self-supervised domain adaptation for computer vision tasks. *IEEE Access*, 7:156694–156706, 2019.

[38] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael I Jordan. Bridging theory and algorithm for domain adaptation. *arXiv preprint arXiv:1904.05801*, 2019.

Supplementary

A Feature Space Comparison of Methods.

In Fig 2 we plot the 2-D TSNE [22] embeddings for features generated by 5 differently trained Alexnet backbones. The embeddings are plotted for all points from 5 randomly picked classes. The source domain points which are light colored circles, come from *real* images of Office-Home and the target domain points which are dark colored markers come from *clipart* images. The labelled target examples are marked with X's. The two plots on the left compare differently pretrained backbones and the three on the right use backbones at the end of different SSDA training processes. In the plots we can see that pretraining the backbone for rotation prediction starts to align and cluster points according to their classes a little better than what a backbone pretrained just on Imagenet can do. Out of the final classifiers on the right, we see that both PAC and MME create well separated classes in feature space allowing for the classifier to have decision boundaries in low-density regions. MME explicitly minimizes conditional entropy which may draw samples even further apart from the classifier boundaries, as compared to our method which simply tries to ensure that the classifier does not separate an example and its perturbed version.

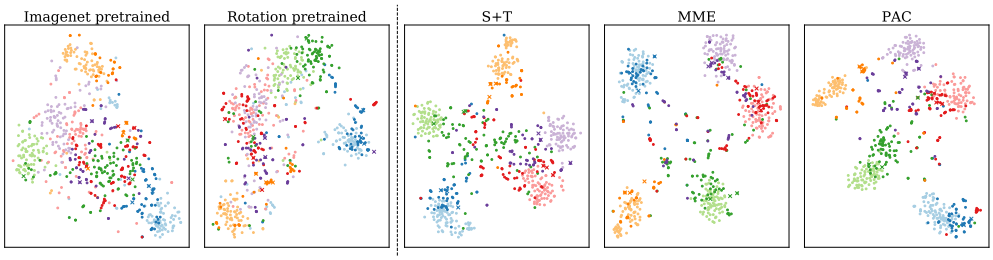

Figure 2: 2-D TSNE embeddings of features from 5 randomly chosen classes. Lighter colors represent source domain points and darker ones represent corresponding target domain points. The dark colored X's are labelled target domain points (3 per class). Pretraining with rotation prediction begins to cluster points of the same class a little better over a backbone pretrained only on Imagenet. Our method on the right, separates classes just as well as MME. Refer to Section 3.1 for discussion. (Best viewed with color and under zoom)

In Table 4, we quantitatively analyze the features using different metrics : The $\mathcal{A}$-distance is a distance metric between the two domains in feature space computed using an SVM trained to classify domains as done in [3]. The higher the error of the SVM domain classifier, the lower is the $\mathcal{A}$-distance. The other two metrics are accuracies of distance based nearest neighbor (NN) classifiers in feature space. The first one, "NN Acc. (Target)" is the accuracy of a classifier that assigns any unlabelled target example, the class label of the target labelled examples closest to it on average in the feature space. "NN Acc. (Source)" similarly uses only the source examples, all of which are labelled, to compute the class label for an unlabelled target example. Finally BD is an empirical inter-class Bhattacharyya distance estimate (details in supp.). Comparing the pretrained backbones, we see that rotation pretraining improves the feature space both by bringing closer the features across the two domains (as indicated by the low $\mathcal{A}$-distance) and aligning them so that features within a class are closer (indicated by the higher NN accuracies). When it comes to final feature spaces of the SSDA methods, we see that MME, being a domain alignment method, reduces $\mathcal{A}$-distance more than PAC. However, PAC is able to better maintain the class-defined neighborhood of features, as indicated by the higher accuracies. Also, PAC has higher inter-class divergence (BD), which leads to lower error according to Eq. (1) in the main paper. This shows that in SSDA, for learning good target classification, low divergence between domains is not a necessity.

B PAC performance with different target shots.

In Figure 3, we plot the target accuracy of 4 methods on the *real* to *clipart* adaptation scenario of Office-Home, for different number of labelled target examples. The method "CR" represents the consistency regularization part of PAC, meaning it starts with an Imagenet pretrained backbone, same

Backbone	$\mathcal{A}$-dist	NN Acc. (Target)	NN Acc. (Source)	BD
Imagenet pt.	1.57	26.8	26.0	-
Rotation pt.	1.28	36.2	34.6	-
S+T	1.49	43.0	43.3	2.01
MME	1.24	51.2	51.5	4.03
PAC	1.45	56.4	56.8	13.43

Table 4: Feature space metrics for different Alexnet backbones on the *real* to *clipart* adaptation scenario of Office-Home. We see that rotation prediction helps improve the initial feature space. Also amongst the SSDA methods, PAC maintains a better class defined neighborhood both within and across domains, even though the two domains are not aligned as closely as in the case of MME.

as S+T and MME [28]. We see that with its domain alignment approach, MME performs well at 0 shots. However, along with pretraining using rotation prediction, which has some alignment effect, PAC does not lag far behind. As the number of labelled examples increase, we see all methods enjoy a significant boost in performance, where the error has an exponential relation to the number of labelled examples as indicated by Eq. 1 (main paper). Since PAC and CR have better feature space clustering, *i.e.*, they have a higher inter-class divergence D, they see a bigger reduction in error.

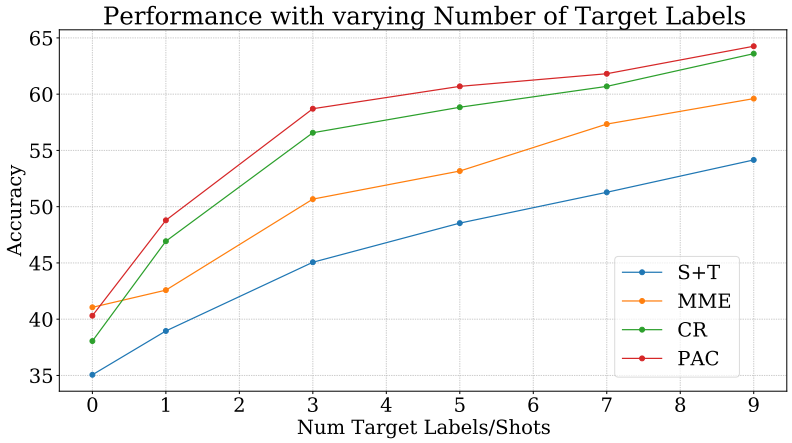

Figure 3: Performance with different number of labelled target examples. MME benefiting from domain alignment performs best at 0 shots. With more labelled examples, there is an exponential decrease in target error (Eq. 1), with PAC and CR benefiting most due to better target clustering, *i.e.*, high inter-class divergence D

C More questions

How does pretraining with rotation prediction compare to a constrastive method? Contrastive pretraining methods [13, 6, 7] have been shown to attain remarkable performance in learning features from unlabelled images that are useful for tasks like image recognition and object detection. We evaluate how momentum contrast (MoCo) [13] and SimSiam [7] perform for pretraining our feature extractor on both source and target images, compared with rotation prediction. Table 5 compares most of the same metrics as Table 4 with the addition of final model (training with labels and consistency regularization) performance on target classification. We see that, both MoCo and SimSiam improve the imagenet pretrained features to some extent helping the final classification performance. However, this improvement is not as high as in the case of rotation prediction pretraining. We see that MoCo has marginally better class-defined structure across domains, but a poorer structure in the target domain indicated by the accuracies of the distance based NN classifiers. Interestingly, we see that nearest neighbors classifiers perform much poorer in the case of SimSiam pretraining as compared to

Backbone pretraining	$\mathcal{A}$-dist	NN Acc. (Target)	NN Acc. (Source)	Final Acc.
Imagenet	1.57	26.8	26.0	54.1
MoCo	1.31	31.4	34.8	56.3
SimSiam	1.53	19.3	17.1	56.6
Rotation	1.28	36.2	34.6	58.8

Table 5: Feature space metrics and final method performance for differently pretrained Alexnet backbones on the *real* to *clipart* adaptation scenario of Office-Home. MoCo and SimSiam help over Imagenet pretraining, but not as much as rotation prediction.

other cases, while the pretraining still benefits final SSDA accuracy. This might indicate that this pretraining helps with optimization in some manner while not clustering the initial feature space as much.

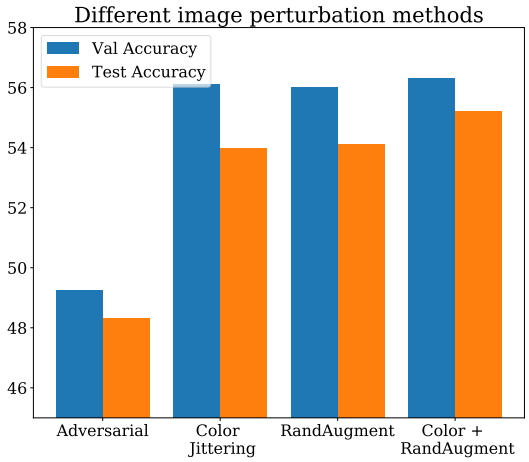

Figure 4: Performance of our method with different augmentation/perturbation methods on real to clipart adaptation of Office-Home. Adversarial perturbation helps, but not as much as image augmentation approaches do. A combination of color jittering and RandAugment performed the best.

Which perturbation technique is best for consistency? We compared three different image augmentation approaches : RandAugment [8] involves a list of 14 different augmentation schemes like translations, rotations, shears, color/brightness enhancements etc., 2 out of which are chosen randomly anytime an image is augmented. We also evaluated color jittering, since common objects in our datasets are largely invariant to small changes in color. Finally we tried a combination of both, and found that this performed best for our method. Fig 4 shows the comparison of the final target accuracies achieved using an Alexnet backbone on the *real* to *clipart* adaptation scenario of Office-Home. Besides perturbations based on augmentation, we also evaluated adversarial image perturbation via virtual adversarial training (VAT) [23]. When using VAT, we found improvements over the simple "S+T" method (48.3% using VAT vs 44.6% without), but as seen from Fig 4, we found this was much lower than image augmentation approaches. This is quite likely because image augmentation imposes a more meaningful neighborhood on images where class labels do not change, while adversarial perturbation does not have this guarantee.

Can pretraining and consistency help other methods? An indication towards the affirmative is seen when we train MME with pretraining and consistency on the 3-shot *real* to *sketch* scenario of DomainNet using a Resnet-34 backbone. The results are shown in Table 6, where we can see that pretraining and consistency both individually help MME's performance, and their combination helps it the most.

Rot^n	CR	Accuracy
		61.9
✓		65.8
	✓	70.4
✓	✓	71.5

Table 6: Pretraining and consistency with MME.

It was explained how pretraining improves initial feature space, but prior work has also used "pretext" tasks like rotation prediction alongside classification for training [37, 33]. How does pretraining compare to that? A comparison of this can be found in table 7, which reports the 3-shot SSDA target accuracies of the two methods on the DomainNet dataset. As can be seen, pretraining using rotation prediction provides more of a performance benefit as compared to using rotation prediction as an auxiliary task like [37, 33]. The latter can help regularize final target classifier training, but likely does not have the benefits that pretraining provides the method via a better initial feature space for training.

Method	C2S	P2C	P2R	R2C	R2P	R2S	S2P	Mean
S+T + Rot^n pred	54.7	59.5	74.1	60.4	62.3	51.8	59.2	60.3
S+T (Rot^n pred pretrained backbone)	59.1	65.3	74.0	64.1	63.9	56.1	61.7	63.5

Table 7: Comparison of rotation prediction for pretraining vs as an auxilliary training task using target accuracies on 3-shot SSDA on different scenarios of DomainNet.

What if pretraining uses rotation prediction only on target? We train the backbone only on target domain data for pretraining with rotation prediction, and then train it like PAC using consistency regularization. On the 3-shot *real* to *clipart* SSDA scenario of Office-Home using an Alexnet backbone, this achieves a final target accuracy of 57.5% compared to 58.9% of PAC. This is indicative of target-only rotation prediction helping the initial feature extractor, but not as much as in the case when source domain data is used along with it.

Rot^n	CR	Accuracy (with source)	Accuracy (only target)
	✓	56.6	35.5
✓	✓	58.9	36.7

Table 8: Ablating source domain information.

How big is the role of source domain data in final target performance? To see this, we train our method with no access to source domain data. This is similar to the semi-supervised learning problem. Target accuracy with only 3 labelled target examples and access to all other unlabelled examples, on the *clipart* domain of Office-Home using an Alexnet backbone, are in the last column of Table 8. For reference, the accuracies of our method with source domain data from the *real* domain (*i.e.* R2C adaptation scenario) are provided in the 3^{rd} column.

D PAC sensitivity to confidence threshold

Our consistency regularization approach uses soft targets based on outputs of the classifier only in cases where the confidence of labelling is high. In Fig 5, we compare the sensitivity of our method to this threshold. We see that higher confidence thresholds up to 0.9 help final target classification performance.

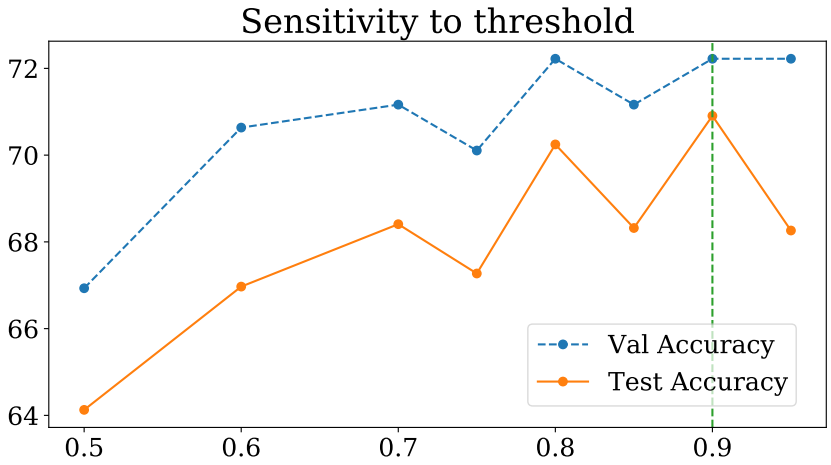

Figure 5: Sensitivity of our method to different thresholds used for consistency regularization. Accuracies reported are on the 3-shot *real* to *sketch* scenario of DomainNet using a Resnet-34 backbone.

E Results on Office and Office-Home

Office-Home [35] is a dataset with 65 categories of objects found in typical office and home environments. It has 4 different visual domains (Art, Clipart, Product, and Real), and we evaluate our methods on all 12 different adaptation scenarios. The 4 domains have close to 3800 images on average. Office [27] dataset has objects of 31 different categories in 3 different domains—amazon, webcam and dslr, with approx. 2800, 800 and 500 images respectively. Following [28] we evaluated only on the 2 cases with amazon as the target domain, since the other two domains have a lot fewer images.

Table 10 shows the results of PAC on the different scenarios of Office-Home, the average accuracy over all these scenarios was also reported in Table 3 in the main paper. Table 9 shows the accuracy of PAC on two scenarios of Office. We see that PAC performs comparably to state of the art. It lags behind a little in the 1-shot scenarios as compared to 3-shot ones.

F Experiment details

All our experiments were implemented in PyTorch [24] using W&B [4] for managing experiments.

F.1 PAC experiments

We used three different backbones for evaluation in different experiments—Alexnet [17], VGG-16 [32] and Resnet-34 [14]. Our backbones before being trained using the rotation prediction task, are pretrained on the Imagenet [9] dataset, same as other methods used for comparison. While using an Alexnet or VGG-16 feature extractor, we use 1 fully connected layer as the classifier, and while using the Resnet-34 backbone, we use a 2-layer MLP with 512 intermediate nodes. The classifier C uses a temperature parameter set to 0.05 to sharpen the distribution it outputs using a softmax. For consistency regularization, the confidence threshold τ was set to 0.9 across all experiments, having validated on the *real* to *sketch* scenario of DomainNet.

Same as [28], we train the models using minibatch-SGD, with s source examples, s labelled target examples and $2s$ unlabelled target examples that the learner "sees" at each training step. $s = 24$ for the VGG and Resnet backbones, while $s = 32$ for Alexnet. The SGD optimizer used a momentum parameter 0.9 and a weight decay (coefficient of ℓ_2 regularizer on parameter norm) of 0.0005. For all experiments, the parameters of the backbone are updated with a learning rate of 0.001, while the parameters of the classifier are updated with a learning rate 0.01. Both of these are decayed as training progresses using a decay schedule similar to [10]. Learning rate at step i (η_i) is set as below:

Network	Method	D to A		W to A	
		1-shot	3-shot	1-shot	3-shot
Alexnet	S+T	50.0	62.4	50.4	61.2
	DANN	54.5	65.2	57.0	64.4
	ADR	50.9	61.4	50.2	61.2
	CDAN	48.5	61.4	50.2	60.3
	ENT	50.0	66.2	50.7	64.0
	MME	55.8	67.8	57.2	67.3
	APE	-	69.0	-	67.6
	BiAT	54.6	68.5	57.9	68.2
	CDAC	**63.4**	**70.1**	**62.8**	**70.0**
	PAC	54.7	66.3	53.6	65.1
VGG	S+T	68.2	73.3	69.2	73.2
	DANN	70.4	74.6	69.3	75.4
	ADR	69.2	74.1	69.7	73.3
	CDAN	64.4	71.4	65.9	74.4
	ENT	72.1	75.1	69.1	75.4
	MME	**73.6**	**77.6**	**73.1**	**76.3**
	PAC	72.4	75.6	70.2	76.0

Table 9: Results on Office. We evaluate using the two scenarios where the target domain is *amazon*

Network	Method	R to C	R to P	R to A	P to R	P to C	P to A	A to P	A to C	A to R	C to R	C to A	C to P	Mean
One-shot														
Alexnet	S+T	37.5	63.1	44.8	54.3	31.7	31.5	48.8	31.1	53.3	48.5	33.9	50.8	44.1
	DANN	42.5	64.2	45.1	56.4	36.6	32.7	43.5	34.4	51.9	51.0	33.8	49.4	45.1
	ADR	37.8	63.5	45.4	53.5	32.5	32.2	49.5	31.8	53.4	49.7	34.2	50.4	44.5
	CDAN	36.1	62.3	42.2	52.7	28.0	27.8	48.7	28.0	51.3	41.0	26.8	49.9	41.2
	ENT	26.8	65.8	45.8	56.3	23.5	21.9	47.4	22.1	53.4	30.8	18.1	53.6	38.8
	MME	42.0	69.6	**48.3**	**58.7**	37.8	**34.9**	52.5	**36.4**	**57.0**	**54.1**	**39.5**	**59.1**	49.2
	BiAT	-	-	-	-	-	-	-	-	-	-	-	-	**49.6**
	PAC	**49.6**	**69.8**	45.9	57.5	**42.5**	30.4	**53.1**	35.8	51.9	48.2	26.0	57.6	47.4
VGG	S+T	39.5	75.3	61.2	71.6	37.0	52.0	63.6	37.5	69.5	64.5	51.4	65.9	57.4
	DANN	52.0	75.7	62.7	72.7	45.9	51.3	64.3	44.4	68.9	64.2	52.3	65.3	60.0
	ADR	39.7	76.2	60.2	71.8	37.2	51.4	63.9	39.0	68.7	64.8	50.0	65.2	57.3
	CDAN	43.3	75.7	60.9	69.6	37.4	44.5	67.7	39.8	64.8	58.7	41.6	66.2	55.9
	ENT	23.7	77.5	64.0	74.6	21.3	44.6	66.0	22.4	70.6	62.1	25.1	67.7	51.6
	MME	49.1	78.7	**65.1**	**74.4**	46.2	**56.0**	68.6	**45.8**	**72.2**	**68.0**	**57.5**	**71.3**	**62.7**
	PAC	**56.4**	**78.8**	64.6	73.1	**54.7**	55.3	**69.8**	43.5	69.5	65.3	45.3	69.6	62.2
Three-shot														
Alexnet	S+T	44.6	66.7	47.7	57.8	44.4	36.1	57.6	38.8	57.0	54.3	37.5	57.9	50.0
	DANN	47.2	66.7	46.6	58.1	44.4	36.1	57.2	39.8	56.6	54.3	38.6	57.9	50.3
	ADR	45.0	66.2	46.9	57.3	38.9	36.3	57.5	40.0	57.8	53.4	37.3	57.7	49.5
	CDAN	41.8	69.9	43.2	53.6	35.8	32.0	56.3	34.5	53.5	49.3	27.9	56.2	46.2
	ENT	44.9	70.4	47.1	60.3	41.2	34.6	60.7	37.8	60.5	58.0	31.8	63.4	50.9
	MME	51.2	73.0	50.3	61.6	47.2	40.7	63.9	43.8	61.4	59.9	44.7	64.7	55.2
	APE	51.9	74.6	51.2	61.6	47.9	**42.1**	**65.5**	44.5	60.9	58.1	44.3	64.8	55.6
	BiAT	-	-	-	-	-	-	-	-	-	-	-	-	56.4
	CDAC	54.9	**75.8**	**51.8**	**64.3**	51.3	**43.6**	65.1	47.5	**63.1**	**63.0**	**44.9**	**65.6**	**56.8**
	PAC	**58.9**	72.4	47.5	61.9	**53.2**	39.6	63.8	**49.9**	60.0	54.5	36.3	64.8	55.2
VGG	S+T	49.6	78.6	63.6	72.7	47.2	55.9	69.4	47.5	73.4	69.7	56.2	70.4	62.9
	DANN	56.1	77.9	63.7	73.6	52.4	56.3	69.5	50.0	72.3	68.7	56.4	69.8	63.9
	ADR	49.0	78.1	62.8	73.6	47.8	55.8	69.9	49.3	73.3	69.3	56.3	71.4	63.1
	CDAN	50.2	80.9	62.1	70.8	45.1	50.3	74.7	46.0	71.4	65.9	52.9	71.2	61.8
	ENT	48.3	81.6	65.5	76.6	46.8	56.9	73.0	44.8	**75.3**	**72.9**	59.1	**77.0**	64.8
	MME	56.9	**82.9**	65.7	**76.7**	53.6	**59.2**	75.7	54.9	**75.3**	**72.9**	**61.1**	76.3	67.6
	PAC	**63.5**	82.3	**66.8**	75.8	**58.6**	57.1	**75.9**	**56.7**	72.2	70.5	57.7	75.3	**67.7**

Table 10: Results on all adaptation scenarios of Office-Home.

$$\eta_i = \frac{\eta_0}{(1 + 0.0001 \times i)^{0.75}}$$

For experiments on the Office and Office-Home dataset, we trained PAC using both an Alexnet and a VGG-16 backbone, and the models were trained for 10000 steps with the stopping point chosen using best validation accuracy.

For the experiments on DomainNet, we use both Alexnet and Resnet-34 backbones, while for VisDA-17, we use only Resnet-34. All models in these experiments were trained for 50000 steps, using validation accuracy for determining the best stopping point.

F.2 Pretraining

As mentioned above, we pretrain our models for rotation prediction starting from Imagenet pretrained weights. A comparison of PAC with a backbone trained with rotation prediction starting from imagenet pretraining (final target accuracy = 58.9%) vs one that does not use any imagenet pretraining (final target accuracy = 43.7%), revealed that there is important feature space information in imagenet pretrained weights that rotation prediction could not capture on its own. This comparison was done using an Alexnet on the *real* to *clipart* adaptation scenario of Office-Home.

Following Gidaris *et al*. [12], we trained the model on all 4 rotations of a single image in each minibatch. Each minibatch contained s images each from source and target domains, which translates to $4s$ images considering all rotations. The Alexnet backbones are trained using a learning rate of 0.01 and $s = 128$. The Resnet-34 and VGG backbones are both trained using $s = 16$ and a learning rate of 0.001. We found that beyond a certain point early on in training, the number of steps of training for rotation prediction did not make a big difference to the final task accuracy, and finally the chosen number of training steps was 4000 for Alexnet, 2000 for VGG-16 and 5000 for Resnet-34 backbones.

F.3 Other Experiments

MoCo pretraining. Using the Alexnet backbone, we trained momentum contrast [13] for 5000 training steps, where in each step the model saw 32 images each from the *real* and the *clipart* domains of Office-Home. The queue length used for MoCo was 4096 and the momentum parameter was 0.999.

SimSiam pretraining. Using the Alexnet backbone, we trained SimSiam [7] for 200 epochs (or 6800 training steps) on a mix of the source (*real*) and target (*clipart*) sets of Office-Home, with a batch size of 256.

Virtual Adversarial Training. For adding a VAT criterion to our model, we closely followed the VAT criterion in VADA [31]. We used a radius of 3.5 for adversarial perturbations and a coefficient of 0.01 for the VAT criterion, which is the KL divergence between the outputs of the perturbed and the unperturbed input from the target domain.

Empirical Bhattacharyya Distance Estimate. We use this estimate to compare target domain inter-class separation in Table 4. For computing an approximation, we made the assumption that features for each class in the target domain are distributed as gaussians with identity covariance and used the closed form Bhattacharyya Distance (BD) between two multivariate gaussians [36]. The estimate then reduces to:

$$BD = \frac{1}{\binom{K}{2}} \sum_{\substack{i,j \in [K] \\ i \neq j}} \frac{1}{8} \left|\left|\mu_i - \mu_j\right|\right|_2^2$$

where μ_i is the mean of class i features of images in the target domain ($\mathcal{D}_t \cup \mathcal{D}_u$).