A Reproducible and Realistic Evaluation of Partial Domain Adaptation Methods

Anonymous Author(s)
Affiliation
Address
email

Abstract

Unsupervised Domain Adaptation (UDA) aims at classifying unlabeled target images leveraging source labeled ones. In this work, we consider the Partial Domain Adaptation (PDA) variant, where we have extra source classes not present in the target domain. Most successful algorithms use model selection strategies that rely on target labels to find the best hyper-parameters and/or models along training. However, these strategies violate the main assumption in PDA: only unlabeled target domain samples are available. The main goal of this work is to provide a realistic evaluation of PDA methods with the different model selection strategies under a consistent evaluation protocol. We evaluate 7 representative PDA algorithms on 2 different real-world datasets using 7 different model selection strategies. Our two main findings are: *(i)* without target labels for model selection, the accuracy of the methods decreases up to 30 percentage points; *(ii)* only one method and model selection pair performs reasonably well on both datasets. Experiments were performed with our PyTorch framework, BenchmarkPDA, which we open source.

1 Introduction

Domain adaptation. Deep neural networks are highly successful in image recognition for in-distribution samples (He et al., 2016). However, they tend to not generalize as well on images with different background or colors not seen during training. Unfortunately, enriching the training set with new samples from different domains is challenging as labeling data is both an expensive and time-consuming task. Thus, researchers have focused on unsupervised domain adaptation (UDA) where we have access to unlabelled samples from a different domain, known as the target domain. The goal of UDA is to classify these unlabeled samples by leveraging the knowledge given by the labeled samples from the source domain (Pan & Yang, 2010). In this work, we consider a variant called *partial domain adaptation* (PDA) where the target domain classes form a subset of the source domain classes (Cao et al., 2018) The extra source classes, not present in the target domain, make the PDA problem more difficult: simply aligning the source and target domains forces a negative transfer where target samples are matched to outlier source-only labels.

Realistic evaluations. Most recent PDA methods report an increase of the target accuracy up to 15 percentage points on average when compared to the baseline approach that uses only source domain samples. However target labels are used for model selection, violating the main UDA assumption. In their absence, the effectiveness of PDA methods remains unclear and model selection constitutes a yet to be solved problem as we show in this work. Model selection strategies without target labels have been designed in a research environment under different experimental protocol. In this work, we study their applicability to real-world settings and evaluate 7 different PDA methods with 7 different model selection strategies on 2 different datasets. We reimplemented all our considered PDA methods under the same experimental protocol for fair comparison. We list below our major findings:

Submitted to the Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

- The accuracy attained by models selected without target labels can decrease up to 30 percentage points compared to the one reported using target labels.
- Out of the 49 model selection strategies and PDA methods pairs considered, only one gave consistent results over all tasks on both datasets.
- Random seed plays an important role in the selection of hyper-parameters. Selected parameters are not stable across different seeds and the standard deviation between accuracies on the same task can be up to 8.4% even when relying on target labels for model selection.
- Under a more realistic scenario where some target labels are available, 100 random samples only leads to a drop of 1 percentage point in accuracy (when compared to using all target samples).

Outline. The paper is organized as follows. In Section 2 we describe the training procedures and evaluation protocols used to evaluate all methods fairly. In Section 3, we discuss the results of the different benchmarked methods and the performance of the different model selection strategies. Finally in Section 4, we give some recommendations for future work in partial domain adaptation.

2 Experimental Protocol

In this section, we discuss our choices regarding the training details, datasets and neural network architecture. To perform our experiments we developed and openly released a PyTorch (Paszke et al., 2019) framework named *BenchmarkPDA*:

`https://anonymous.4open.science/r/BenchmarkPDA-7F73`

It is the standard in the literature when proposing a new method to report directly the results of its competitors from the original papers (Cao et al., 2018; Xu et al., 2019). In addition, some methods also differ in the network architecture implementation (Xu et al., 2019; Gu et al., 2021) and other experimental details leading to unfair comparisons between the different methods. Therefore we chose to implement all methods with the same commonly used neural network architecture, optimizer, learning rate schedule and evaluation protocol. We discuss the details below.

2.1 Methods, Datasets, Training and Evaluation Details

Model Selection Strategies. We consider 7 different strategies: S-ACC, DEV, ENT, SND, ORACLE, 1-SHOT, 100-RND, described in detail Appendix A. We use them both for hyper-parameter tuning and selecting the best model along training. Implementation details are discussed in Appendix C.2.

Methods. We implemented 7 PDA methods by adapting the code from the Official GitHub repositories of each method: Source Only, PADA (Cao et al., 2018), SAFN (Xu et al., 2019), BA3US (Jian et al., 2020), AR (Gu et al., 2021), JUMBOT (Fatras et al., 2021), MPOT (Nguyen et al., 2022). We give an overview of the methods in Appendix B and provide the links to the repositories in Appendix C.1.

Datasets. Our first dataset is OFFICE-HOME (Venkateswara et al., 2017). It has four domains: Art (A), Clipart (C), Product (P) and Real-World (R) with 65 object categories in each of them. The second dataset is VISDA (Peng et al., 2017) with synthetic images as source domain and real-world images as target domain. The domains have 12 object categories. For the partial setting, we follow Cao et al. (2018) and select the first, in alphabetical order, 25 and 6 categories for OFFICE-HOME and VISDA, respectively, in each partial target domain.

Architecture. Our network is composed of a feature extractor with a linear classification layer on top of it. The feature extractor is a ResNet50 (He et al., 2016), pre-trained on ImageNet (Deng et al., 2009), with its last linear layer removed and replaced by a linear bottleneck layer of dimension 256.

Optimizer. We use the SGD (Robbins & Monro, 1951) with momentum of 0.9, a weight decay of $5e^{-4}$ and Nesterov acceleration. As the bottleneck and classifer layers are randomly initialized, we set their learning rates to be 10 times that of the pre-trained ResNet50 backbone. We schedule the learning rate with a strategy similar to the one in Ganin et al. (2016): $\chi_p = \frac{\chi_0}{(1+\mu i)^{-\nu}}$, where i is the current iteration, $\chi_0 = 0.001$, $\gamma = 0.001$, $\nu = 0.75$. See Appendix C.4 for more details.

Evaluation Protocol. For the hyper-parameters chosen with each model selection strategy, we run the methods for each task 3 times, each with a different seed (2020, 2021, 2022). We tried to control for the randomness across methods by setting the random seeds at the beginning of training.

DATASET	METHOD	S-ACC	ENT	DEV	SND	1-SHOT	100-RND	ORACLE
OFFICE-HOME	S. ONLY	60.38±0.5	60.73±0.2	60.22±0.3	59.55±0.3	58.92±0.4	60.34±0.4	61.87±0.3
	PADA	63.08±0.3	59.74±0.5	52.72±2.8	62.36±0.4	62.00±0.5	63.22±0.1	63.72±0.3
	SAFN	62.09±0.2	61.37±0.3	62.03±0.4	62.59±0.1	49.30±0.7	62.36±0.2	63.30±0.2
	BA3US	**68.32±1.1**	73.36±0.6	62.25±7.1	**75.37±0.8**	65.56±7.6	75.19±0.4	75.98±0.3
	AR	65.68±0.3	70.58±0.4	**64.32±0.9**	70.25±0.2	70.56±0.7	70.34±0.2	72.73±0.3
	JUMBOT	62.89±0.2	**74.61±0.8**	61.28±0.1	72.29±0.2	**74.95±0.1**	**75.74±0.3**	77.15±0.4
	MPOT	66.24±0.1	64.46±0.1	61.37±0.2	46.92±0.4	68.28±0.2	73.06±0.3	**77.31±0.5**
VISDA	S. ONLY	55.15±2.4	55.24±3.2	55.07±1.2	55.02±2.9	55.72±2.2	58.16±0.6	59.48±0.4
	PADA	47.48±4.8	32.32±4.9	43.43±5.3	56.83±1.0	53.15±2.9	54.38±2.7	54.57±2.6
	SAFN	58.20±1.7	42.83±6.3	58.62±1.3	44.82±8.8	56.89±2.1	59.09±2.8	62.64±1.5
	BA3US	55.10±3.7	65.58±1.4	58.40±1.4	51.07±4.3	64.77±1.4	67.44±1.2	67.67±1.3
	AR	66.68±1.0	64.27±3.6	**67.20±1.5**	55.69±0.9	70.29±1.7	72.60±0.8	73.85±0.9
	JUMBOT	60.63±0.7	62.42±2.4	59.86±0.6	77.69±4.2	**78.34±1.9**	83.49±1.9	84.01±1.9
	MPOT	**70.02±2.0**	**74.64±4.4**	61.62±1.3	**78.40±3.9**	70.96±3.7	**86.69±5.1**	**86.95±5.0**

Table 1: Task accuracy average over seeds 2020, 2021, 2022 on Partial OFFICE-HOME and Partial VISDA for the different PDA methods and model selection strategy pairs. Best results in **bold**.

3 Partial domain adaptation experiments

We start the results section by comparing the performance of the different model selection strategies. Finally, we discuss the sensitivity of methods to the random seed. In total, *2547* models were trained to make this study.

3.1 Results for Model Selection Strategies

Model Selection Strategies (w/ vs w/o target labels) All average accuracies on the OFFICE-HOME and VISDA datasets can be found in Table 1. For all methods on OFFICE-HOME, we can see that the results for model selections strategies which do not use target labels are below the results given by ORACLE. For some pairs, the drop of performance can be significant, leading some methods to perform on par or worse than the S. ONLY method. That is the case on OFFICE-HOME when DEV is paired with either BA3US, JUMBOT and MPOT. Even worse is MPOT with SND as the average accuracy is more than 10 percentage points below that of S. ONLY with any model selection strategy. Overall on OFFICE-HOME, except for MPOT, all methods when paired with either ENT or SND give results that are at most 2 percentage points below compared to when paired with ORACLE.

A similar situation can be seen over the VISDA dataset where the accuracy without target labels can be down to 25 percentage points. Yet again, some model selection strategies can lead to scores even worse than S. ONLY. That is the case for PADA, SAFN and BA3US. Contrary to OFFICE-HOME, all model selection strategies without target labels lead to at least one method with results on par or worse in comparison to the S. ONLY method. Overall, no model selection strategy without target labels can lead to score on par to the ORACLE model selection strategy. Finally, PADA performs worse than S. ONLY for most model selection strategies, including the ones which use target labels. However, when combined with SND it performs better than with ORACLE on average, although still within the standard deviation. This is a consequence of the random seed dependence, discussed in detail below: as the hyper-parameters were chosen by performing just one run, we were "unlucky". In general, it confirms the assumption in the literature about the difficulty of the VISDA dataset.

Model Selection Strategies (w/ target labels) We recall that the ORACLE model selection strategy uses all the target samples to compute the accuracy. 1-SHOT and 100-RND use only subsets of target samples: 1-SHOT has only one sample per class for a total of 25 and 6 on OFFICE-HOME and VISDA, respetively, while 100-RND has 100 random target samples. Our results show that using only 100 random target labeled samples is enough to reasonably approximate the target accuracy leading to only a small accuracy drop (one percentage point in almost all cases) for both datasets. Not surprisingly, the gap between the 1-SHOT and ORACLE model selection strategies is even bigger, leading in some instances to worse results than with a model selection strategy that uses no target labels. This poor performance of the 1-SHOT model selection strategy also highlights that semi-supervised domain adaptation (SSDA) methods are not a straightforward alternative to the 100-RND model selection strategy. While one could argue that the target labels could be leveraged during training like in SSDA methods, one still needs labeled target data to perform model selection. However our results suggest

that we would need at least 3 samples per class for SSDA methods. In addition, knowing that we have a certain number of labeled samples per class provides information regarding which classes are target only, one of the main assumptions in PDA. In that case, PDA methods could be tweaked. This warrants further study that we leave as future work. Finally, we have also investigated a smaller labeled target set of 50 random samples (50-RND) instead of 100 random samples. The accuracies of methods using 50-RND were not as good as when using 100-RND. All results of pairs of methods and 50-RND can be found in Appendix D. The smaller performance show that the size of the labeled target set is an important element and we suggest to use at least 100 random samples.

Model Selection Strategies (w/o target labels) Overall, only the JUMBOT and SND pair performed reasonably well with respect to the JUMBOT and ORACLE pair on both datasets. All other pairs failed in either one of the datasets. Our experiments show that there is no model selection strategy which performs well for all methods. That is why to deploy models in a real-world scenario, we advise to test selected models on a small labeled target set (our 100-RND model selection strategy) to assess the performance of the models as current model selection without target labels can perform poorly.

Our conclusion is that the model selection for PDA methods is still an open problem. We conjecture that is also likely the case for UDA.

3.2 Random Seed Dependence

Ideally, PDA methods should be robust to the choice of random seed. This is of particular importance when performing hyper-parameter tuning since typically only one run per set of hyper-parameters is done (that was the case in our work as well). We investigate this robustness by averaging all the results presented over three different seeds (2020, 2021 and 2022) and reporting the standard deviations. This is in contrast with previous work where only single run is reported (Fatras et al., 2021; Gu et al., 2021). Other works (Cao et al., 2018; Xu et al., 2019; Jian et al., 2020) that report standard deviations do not specify if the random seed is different across runs. Results for all tasks on VISDA dataset and OFFICE-HOME are in Appendix D due to space constraints.

Our experiments show that some methods express a non-negligible instabilities over randomness with respect to any model selection methods. This is particularly true for BA3US when paired with DEV and 1-SHOT as model selection strategies: there are several tasks where the standard deviation is above 10%. While in this case this instability may stem from the poor performance of the model selection strategies, it is also visible when ORACLE is the model selection strategy used. For instance, the M-POT has a standard deviation of 3.3% on the AP task of OFFICE-HOME which corresponds to a variance of 11%. On VISDA this instability and seed dependence is even larger.

4 Conclusion

In this paper, we investigated how model selection strategies affect the performance of PDA methods. We performed a quantitative study with seven PDA methods and seven model selection strategies on two real-word datasets. Based on our findings, we provide the following recommendations:

i) Target label samples should be used to test models before using them in real-world scenario. While this breaks the main PDA assumption, it is impossible to confidently deploy PDA models selected without the use of target labels. Indeed, model selection strategies without target labels lead to a significant drop in performance in most cases in comparison to using a small validation set. We argue that the cost of labeling it outweighs the uncertainty in current model selection strategies.

ii) The robustness of new PDA method to randomness should be tested over at least three different seeds. We suggest to use the seeds (2020, 2021, 2022) to allow for a fair comparison with our results.

iii) An ablation study should be considered when a novel architecture is proposed to quantify the associated increase of performance.

As our work focus on a quantitative study of model selection methods and reproducibility of state-of-the-art partial domain adaptation methods, we do not see any potential ethical concern. Future work will investigate new model selection strategies which can achieve similar results as model selection strategies which use label target samples.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/arjovsky17a.html.

Bharath Bhushan Damodaran, Benjamin Kellenberger, Remi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 135–150, 2018.

Laetitia Chapel, Mokhtar Z. Alaya, and Gilles Gasso. Partial optimal transport with applications on positive-unlabeled learning. In *Advances in Neural Information Processing Systems*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Kilian Fatras, Younes Zine, Szymon Majewski, Rémi Flamary, Rémi Gribonval, and Nicolas Courty. Minibatch optimal transport distances; analysis and applications, 2021.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1180–1189, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ganin15.html.

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.

Xiang Gu, Jian Sun, and Zongben Xu. Spherical space domain adaptation with robust pseudo-label loss. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Xiang Gu, Xi Yu, Yan Yang, Jian Sun, and Zongben Xu. Adversarial reweighting for partial domain adaptation. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=f5liPryFRoA.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=lQdXeXDoWtI.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Liang Jian, Wang Yunbo, Hu Dapeng, He Ran, and Feng Jiashi. A balanced and uncertainty-aware approach for partial domain adaptation. In *European Conference on Computer Vision (ECCV)*, August 2020.

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 (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/ab88b15733f543179858600245108dd8-Paper.pdf.

Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJWechg0Z.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. Unsupervised domain adaptation: A reality check. *arXiv preprint arXiv: Arxiv-2111.15672*, 2021.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. Benchmarking validation methods for unsupervised domain adaptation. *arXiv preprint arXiv: Arxiv-2208.07360*, 2022.

Khai Nguyen, Dang Nguyen, Tung Pham, and Nhat Ho. Improving mini-batch optimal transport via partial transportation. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *CoRR*, abs/1710.06924, 2017.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

Kuniaki Saito, Donghyun Kim, Piotr Teterwak, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density. *arXiv preprint arXiv:2108.10860*, 2021.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(35):985–1005, 2007. URL http://jmlr.org/papers/v8/sugiyama07a.html.

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.

Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

Kaichao You, Ximei Wang, Mingsheng Long, and Michael Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7124–7133. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/you19a.html.

A Reproducible and Realistic Evaluation of Partial Domain Adaptation Methods

Supplementary material

Outline. The supplementary material of this paper is organized as follows:

- In Section A, we give an overview of the different model selection strategies.
- In Section B, we give an overview of the different PDA methods considered in this work.
- In Section C, we give more details on our experimental protocol.
- In Section D, we provide additional results from our experiments.

A Model Selection Strategies: An Overview

Model selection (choosing hyper-parameters, training checkpoints, neural network architectures) is a crucial part of training neural networks. In the supervised learning setting, a validation set is used to estimate the model's accuracy. However, in UDA such approach is not possible as we have unlabeled target samples. Several strategies have been designed to address this issue. Below, we discuss the ones used in this work.

Source Accuracy (S-ACC). Ganin & Lempitsky (2015) used the accuracy estimated on a small validation set from the source domain to perform the model selection. While the source and target accuracies are related, there are no theoretical guarantees. You et al. (2019) showed that when the domain gap is large this approach fails to select competitive models.

Deep Embedded Validation (DEV). Sugiyama et al. (2007) and Long et al. (2018) perform model selection through Importance-Weighted Cross-Validation (IWCV). Under the assumption that the source and target domain follow a covariate shift, the target risk can be estimated from the source risk through importance weights that give increased importance to source samples that are closer to target samples. These importance weights correspond to the ratio of the target and source densities and are estimated using Gaussian kernels. Recently, You et al. (2019) proposed an improved variant, Deep Embedded Validation (DEV), that controls the variance of the estimator and estimates the importance weights with a discriminative model that distinguish source samples from target samples leading to a more stable and effective method.

Entropy (ENT). While minimizing the entropy of the target samples has been used in domain adaptation to improve accuracy by promoting tighter clusters, Morerio et al. (2018) showed that it can also be used for model selection. The intuition is that a lower entropy model corresponds to a high confident model with discriminative target features and therefore reliable predictions.

Soft Neighborhood Density (SND). Saito et al. (2021) argue that a good UDA model will have a neighborhood structure where nearby target samples are in the same class. They point out that entropy is not able to capture this property and propose the Soft Neighborhood Density (SND) score to address it.

Target Accuracy (ORACLE). We consider as well the target accuracy on all target samples. While we emphasize once again its use is not realistic in unsupervised domain adaptation (hence why we will refer to it as ORACLE), it has nonetheless been used to report the best accuracy achieved by the model along training in several previous works (Cao et al., 2018; Xu et al., 2019; Jian et al., 2020; Gu et al., 2021; Nguyen et al., 2022). Here, we use it as an upper bound for all the other model selection strategies and to check the reproducibility of previous works.

Small Labeled Target Set (1-SHOT and 100-RND). For real-world applications in an industry setting, it is unlikely that a model will be deployed without the very least an estimate of its performance for which target labels are required. Therefore, one can imagine a situation where a PDA method is used and a small set of target samples is available. Thus, we will compute the target accuracy with 1 labeled sample per class (1-SHOT) and 100 random labeled target samples (100-RND) as model selection strategies. One could argue that the 100 random samples could have been used in the training with semi-supervised domain adaptation methods. However, note that we do not know how

Method	Architecture (bottleneck)	Runs per task	Model Selection	
			Hyper-Parameters	Along Training
PADA	Linear	1	IWCV (lacks details)	ORACLE
SAFN	Non-Linear	3	Unknown	ORACLE
BA3US	Linear	3	Unknown	ORACLE
AR	Non-Linear	1	IWCV (lacks details)	ORACLE
JUMBOT	Linear	1	ORACLE	FINAL
MPOT	Linear	3	Unknown	ORACLE

Table 2: Summary of the experimental protocol used for SOTA partial domain adaptation methods.

many classes we have on the target domain so it is hard to form a split when we have uncertainty of classes. For instance, 100 random represents possibly less than 2 samples per class for one of our real-world dataset, as we do not know the number of classes, making a potential split between a train and validation target sets not possible.

Since S-ACC, DEV and SND require a source validation set, we divide the source samples into a training subset (80%) and validation subset (20%). Regardless of the model selection strategy used, all methods are trained using the source training subset. This is in contrast with previous work that uses all source samples, but necessary to ensure a fair comparison of the model selection strategies.

B Partial Domain Adaptation Methods

In this section, we give a brief description of the PDA methods considered in our study. They can be grouped into two families: adversarial training and divergence minimization.

Adversarial training. To solve the UDA problem, Ganin et al. (2016) aligned the source and target domains with the help of a domain discriminator trained adversarially to be able to distinguish the samples from the two domains. However, when applied to the PDA problem this strategy leads to negative transfer and the model performs worse than a model trained only on source data. Cao et al. (2018) proposed PADA that introduces a PDA specific solution to adversarial domain adaptation: the contribution of the source-only class samples to the training of both the source classifier and the domain adversarial network is decreased. This is achieved through class weights that are calculated by simply averaging the classifier prediction on all target samples. As the source-only classes should not be predicted in the target domain, they should have lower weights. More recently, Jian et al. (2020) proposed BA3US which augments the target mini-batch with source samples to transform the PDA problem in a vanilla DA problem. In addition, an adaptive weighted complement entropy objective is used to encourage incorrect classes to have uniform and low prediction scores.

Divergence minimization. Another standard direction to align the source and target distributions in the feature space of a neural network is to minimize a given divergence between distributions of domains. Xu et al. (2019) empirically found than target samples have low feature norm compared to source samples. Based on this insight, they proposed SAFN which progressively adapts the feature norms of the two domains by minimizing the Maximum Mean Feature Norm Discrepancy. Other approaches are based on optimal transport (OT) (Bhushan Damodaran et al., 2018). For the PDA problem in specific, (Fatras et al., 2021) developed JUMBOT, a mini-batch unbalanced optimal transport that learns a joint distribution of the embedded samples and labels. The use of unbalanced OT is critical for the PDA problem as it allows to transport only a portion of the mass limiting the negative transfer between distributions. Based on this work, (Nguyen et al., 2022) investigated the partial OT variant (Chapel et al., 2020), a particular case of unbalanced OT, proposing M-POT. Finally, another line of work is to use the Kantorovich-Rubenstein duality of optimal transport to perform the alignment similarly to WGAN(Arjovsky et al., 2017). This is precisely the work of Gu et al. (2021) that proposed, AR. In addition, source samples are reweighted in order to reduce the negative transfer from the source-only class samples. The Kantorovich-Rubenstein duality relies on a one Lipschitz function which is approximated using adversarial training like the PDA methods described above.

We sum-up all considered methods and their experimental protocol in Table 2.

C Additional details on Experimental Protocol

C.1 Implementations in BenchmarkPDA

In order to reimplement the different PDA methods, we adapted the code from the official repository associated with each of the paper. We list them in Table 3.

Method	Code Repository
PADA	https://github.com/thuml/PADA/blob/master/pytorch/src/
SAFN	https://github.com/jihanyang/AFN/blob/master/partial/OfficeHome/SAFN/code/
BA3US	https://github.com/tim-learn/BA3US/blob/master/
AR	https://github.com/XJTU-XGU/Adversarial-Reweighting-for-Partial-Domain-Adaptation
JUMBOT	https://github.com/kilianFatras/JUMBOT
M-POT	https://github.com/UT-Austin-Data-Science-Group/Mini-batch-OT/tree/master/PartialDA

Table 3: Office Github code repositories for the PDA methods considered in this work.

One of our main claims regarding previous work is the use of target labels to choose the best model along training. This can be easily verified by inspecting the code. For PADA it can be seen on line 240 of the script "train_pada.py", for BA3US in line 116 for the script "run_partial.py", for M-POT it can be seen line 164 of the file "run_mOT.py", for SAFN it can be seen in the "eval.py" file and finally for AR in line 149 of the script "train.py".

C.2 Model Selection

DEV requires learning a discriminative model to distinguish source samples from target samples. Its neural network architecture must be specified as well the training details. You et al. (2019) (DEV) use a multilayer perceptron, while Saito et al. (2021) (SND) use a Support Vector Machine in their reimplementation of DEV. We empirically observed the latter to yield more stable weights and so that was the one we used. In order to train the SVM discriminator, following (Saito et al., 2021), we take 3000 feature embeddings from source samples used in training and 3000 random feature embeddings from target samples, both chosen randomly. We do a 80/20 split into training and test data. The SVM is trained with a linear kernel for a maximum of 4000 iterations. Of 5 different SVM models trained with decay values spaced evenly on log space between 10^{-2} and 10^{4} the one that leads to the highest accuracy (in distinguishing source from target features) on the test data split is the chosen one.

As for SND, it also requires specifying a temperature for the temperature scaling component of the strategy. We used the default value of 0.05 that is suggested in (Saito et al., 2021).

Finally, we mention that the samples used for 100-RND were randomly selected and their list is made available together with the code. As for the samples used for 1-SHOT, they are the same as the ones used in semi-supervised domain adaptation.

C.3 Optimizer

We use the SGD (Robbins & Monro, 1951) algorithm with momentum of 0.9, a weight decay of $5e^{-4}$ and Nesterov acceleration. As the bottleneck and classifer layers are randomly initialized, we set their learning rates to be 10 times that of the pre-trained ResNet50 backbone. We schedule the learning rate with a strategy similar to the one in (Ganin et al., 2016): $\chi_p = \frac{\chi_0}{(1+\mu i)^{-\nu}}$, where i is the current iteration, $\chi_0 = 0.001$, $\gamma = 0.001$, $\nu = 0.75$. While this schedule is slightly different than the one reported in previous work, it is the one implemented in the different official code implementations. Finally, as for the mini-batch size, JUMBOT and M-POT were designed with a stratified sampling, *i.e.,* a balanced source mini-batch with the same number of samples per class. This allows to reduce the negative transfer between domains and is crucial to their success. On the other hand, it was shown that for some methods (e.g. BA3US) using a larger mini-batch, than what was reported, leads to a decreased performance (Fatras et al., 2021). As a result, we used the default mini-batch strategies for each method. JUMBOT and M-POT use stratefied mini-batches of size 65 for OFFICE-HOME and 36 for VISDA. All other methods use a standard random uniform sampling strategy with a mini-batch size of 36. For OFFICE-HOME, all methods are trained for 5000 iterations, while for VISDA are

Dataset	Variant	BA3US			JUMBOT			MPOT			SAFN		
		ENT	DEV	SND	ENT	DEV	SND	ENT	DEV	SND	ENT	DEV	SND
OFFICE-HOME	Naive	52.60	**63.10**	44.48	52.30	26.75	17.67	**49.01**	16.72	**30.63**	32.12	**49.67**	5.01
	Heuristic	**58.45**	**63.10**	**60.96**	**56.24**	**45.79**	**55.16**	**49.01**	**45.61**	**30.63**	**46.27**	**49.67**	**49.67**
VISDA	Naive	39.06	**36.99**	1.14	35.89	**54.53**	11.99	**75.04**	**55.33**	36.11	**52.82**	**53.26**	0.83
	Heuristic	**67.50**	34.94	**38.76**	**47.23**	**54.53**	**66.42**	**75.04**	**55.33**	**85.36**	**52.82**	**53.26**	**52.82**

Table 4: Comparison between the naive model selection strategy and our heuristic approach. Accuracy on AC task for OFFICE-HOME and SR task for VISDA. Best results in **bold**.

Method	HP	Values
PADA	λ	$[0.1, 0.5, 1.0, 5.0, 10.0]$
BA3US	λ_{wce}	$[0.1, 0.5, 1, 5, 10]$
	λ_{ent}	$[0.01, 0.05, 0.1, 0.5, 1]$
SAFN	λ	$[0.005, 0.01, 0.05, 0.1, 0.5]$
	Δ_r	$[0.01, 0.1, 1.0]$
AR	ρ_0	$[2.5, 5.0, 7.5, 10.0]$
	A_{up}	$[5.0, 10.0]$
	A_{low}	$-A_{up}$
	λ_{ent}	$[0.01, 0.1, 1.0]$
JUMBOT	τ	$[0.001, 0.01, 0.1]$
	η_1	$[0.00001, 0.0001, 0.001, 0.01, 0.1]$
	η_2	$[0.1, 0.5, 1.]$
	η_3	$[5, 10, 20]$
MPOT	ϵ	$[0.5, 1.0, 1.5]$
	η_1	$[0.0001, 0.001, 0.01, 0.1, 1.0]$
	η_2	$[0.1, 1.0, 5.0, 10.0]$
	m	$[0.1, 0.2, 0.3, 0.4]$

Table 5: Hyper-Parameter values for each PDA method considered in the grid search.

trained for 10000 iterations, with the exception of the S. ONLY which is trained for 1000 iterations on OFFICE-HOME and 5000 iterations on VISDA.

C.4 Hyper-Parameters

Previous works (Gulrajani & Lopez-Paz, 2021; Musgrave et al., 2021, 2022) perform random searches with the same number of runs for each method. In contrast, we perform hyper-parameter grid searches for each method. As a result, the hyper-parameter tuning budgets differs across the methods depending on the number of hyper-parameters and the chosen grid. While one can argue this leads to an unfair comparison of the methods, in practice in most real-world applications one will be interested in using the best method and our approach will capture precisely that.

The hyper-parameter tuning needs to be performed for each task of each dataset, but that would require a significant computational resources without a clear added benefit. Instead for each dataset, we perform the hyper-parameter tuning on a single task: AC for OFFICE-HOME and SR for VISDA. This same strategy was adopted in (Fatras et al., 2021) and the hyper-parameters were found to generalize to the remaining tasks in the dataset. We conjecture that this may be due to the fact that information regarding the number of target only classes is implicitly hidden in the hyper-parameters.

Several runs in our hyper-parameter search for JUMBOT, M-POT and BA3US were unsuccessful with the optimization reaching its end without the model being trained at all. This poses a challenge to DEV, SND and ENT and its one of the failures modes accounted for in (Saito et al., 2021). Following their recommendations, for JUMBOT, M-POT and BA3US we discard runs with low source accuracy. Our threshold was 69.01% and 89.83% for the AC task on OFFICE-HOME and TV task on VISDA, respectively. It corresponds to 90% of the source accuracy attained by the Source-Only model on

Method	Dataset	HP	ORACLE	1-SHOT	50-RND	100-RND	S-ACC	ENT	DEV	SND
PADA	OFFICE-HOME	λ	0.5	0.1	0.1	0.5	0.1	1.0	5.0	0.5
	VISDA	λ	0.5	1.0	10.0	0.5	1.0	0.5	5.0	0.1
SAFN	OFFICE-HOME	λ	0.005	0.1	0.005	0.01	0.005	0.01	0.005	0.005
		Δr	0.1	0.01	0.01	0.01	0.01	0.1	0.1	0.1
	VISDA	λ	0.005	0.005	0.05	0.05	0.005	0.05	0.005	0.05
		Δr	0.1	0.01	0.01	0.01	0.01	0.01	0.01	0.01
BA3US	OFFICE-HOME	λ_{wce}	5.0	10.0	5.0	5.0	5.0	0.1	10.0	1.0
		λ_{ent}	0.05	0.05	0.01	0.05	0.01	0.1	0.05	0.01
	VISDA	λ_{wce}	1.0	1.0	0.1	1.0	5.0	1.0	5.0	5.0
		λ_{ent}	0.5	0.5	0.5	0.5	0.05	0.5	0.05	1.0
AR	OFFICE-HOME	ρ_0	2.5	2.5	5.0	5.0	2.5	5.0	7.5	10.0
		A_{up}	5.0	5.0	10.0	5.0	5.0	10.0	10.0	10.0
		A_{low}	-5.0	-5.0	-10.0	-5.0	-5.0	-10.0	-10.0	-10.0
		λ_{ent}	0.1	0.1	1.0	1.0	0.01	1.0	0.01	1.0
	VISDA	ρ_0	2.5	2.5	2.5	2.5	2.5	7.5	2.5	10.0
		A_{up}	10.0	10.0	10.0	10.0	5.0	10.0	10.0	10.0
		A_{low}	-10.0	-10.0	-10.0	-10.0	-5.0	-10.0	-10.0	-10.0
		λ_{ent}	0.1	0.1	0.1	0.1	0.01	0.1	0.01	0.01
JUMBOT	OFFICE-HOME	τ	0.01	0.01	0.01	0.001	0.1	0.01	0.01	0.001
		η_1	0.0001	0.0001	0.001	0.0001	0.01	1e-05	0.01	1e-05
		η_2	0.5	1.0	0.5	0.1	0.1	0.5	1.0	1.0
		η_3	10.0	5.0	5.0	5.0	5.0	20.0	10.0	5.0
	VISDA	τ	0.01	0.01	0.01	0.01	0.001	0.01	0.001	0.01
		η_1	0.001	0.001	0.001	0.001	0.01	1e-05	0.01	0.0001
		η_2	1.0	1.0	0.5	1.0	0.1	0.5	1.0	1.0
		η_3	5.0	5.0	5.0	5.0	10.0	5.0	20.0	5.0
MPOT	OFFICE-HOME	ϵ	0.5	0.5	1.0	0.5	1.0	1.5	1.0	1.5
		η_1	0.01	0.01	0.01	0.01	0.001	0.0001	1.0	0.01
		η_2	10.0	1.0	1.0	1.0	1.0	10.0	0.1	1.0
		m	0.3	0.1	0.1	0.2	0.3	0.4	0.2	0.4
	VISDA	ϵ	0.5	0.5	0.5	0.5	1.0	1.0	1.0	0.5
		η_1	0.01	0.001	0.01	0.01	0.001	0.0001	0.0001	0.01
		η_2	1.0	1.0	1.0	1.0	1.0	10.0	1.0	10.0
		m	0.3	0.1	0.3	0.3	0.2	0.4	0.2	0.3

Table 6: Hyper-parameters selected for the different methods for each model selection strategy on both OFFICE-HOME and VISDA.

each task. We consider 90% because the ablation study of some methods showed that doing the adaptation decreased slightly the performance on the source domain (Bhushan Damodaran et al., 2018). See Table 4 that shows that this heuristic leads to improved results.

Lastly, when choosing the hyper-parameters, we only consider the model at the end of training, discarding the intermediate checkpoint models in order to select hyper-parameters which do not lead to overfitting at the end of training and better generalize to the other tasks. Following the above protocol, for each dataset we trained *468* models in total in order to find the best hyper-parameters. Then, to obtain the results with our neural network architecture on all tasks of each dataset, we trained an additional *1224* models for OFFICE-HOME and *156* models for VISDA. We additionally trained 231 models with the different neural network architectures for AR and SAFN. In total, *2547* models were trained to make this study.

In Table 5, we report the values used for each hyper-parameter in our grid search. We report in Table 6 the hyper-parameters chosen by each model selection strategy for each method on both datasets. In addition, for the reproducibility of AR with the proposed architecture in Gu et al. (2021), a feature

METHOD	AC	AP	AR	CA	CP	CR	PA	PC	PR	RA	RC	RP	Avg
S. ONLY†	46.33	67.51	75.87	59.14	59.94	62.73	58.22	41.79	74.88	67.40	48.18	74.17	61.35
S. ONLY (Ours)	45.43	68.91	79.53	55.59	57.42	65.23	59.32	40.80	75.80	69.88	47.20	77.31	61.87
PADA†	51.95	67.00	78.74	52.16	53.78	59.03	52.61	43.22	78.79	73.73	56.60	77.09	62.06
PADA (Ours)	50.53	67.45	80.14	57.30	54.47	64.55	61.07	40.94	79.55	73.09	54.63	80.93	63.72
SAFN†*	58.93	76.25	81.42	70.43	72.97	77.78	72.36	55.34	80.40	75.81	60.42	79.92	71.84
SAFN* (Ours)	59.98	79.85	85.18	72.02	73.73	78.54	76.09	59.32	83.25	80.04	64.20	84.44	74.72
SAFN (Ours)	49.57	68.55	78.26	57.91	59.29	66.81	59.87	45.29	75.98	69.08	51.68	77.29	63.30
BA3US†	60.62	83.16	88.39	71.75	72.79	83.40	75.45	61.59	86.53	79.25	62.80	86.05	75.98
BA3US (Ours)	63.26	82.75	89.16	69.91	71.93	77.58	75.73	59.94	86.89	80.93	66.77	86.93	75.98
AR†*	62.13	79.22	89.12	73.92	75.57	84.37	78.42	61.91	87.85	82.19	65.37	85.27	77.11
AR* (Ours)	62.75	81.55	89.07	71.63	73.41	82.94	75.88	61.03	85.70	79.86	62.93	85.30	76.00
AR (Ours)	57.33	79.61	86.31	69.45	71.88	79.94	70.28	53.57	83.78	77.26	59.68	83.72	72.73
JUMBOT†	62.70	77.50	84.40	76.00	73.30	80.50	74.70	60.80	85.10	80.20	66.50	83.90	75.47
JUMBOT (Ours)	61.87	78.19	88.11	77.69	76.75	84.15	76.83	63.72	84.80	81.79	64.70	87.17	77.15
MPOT†	64.60	80.62	87.17	76.43	77.61	83.58	77.07	63.74	87.63	81.42	68.50	87.38	77.98
MPOT (Ours)	64.48	80.88	86.78	76.22	77.95	82.59	75.18	64.60	84.87	80.59	67.04	86.52	77.31

Table 7: Comparison between reported (†) accuracies on partial OFFICE-HOME from published methods with our implementation using the ORACLE model selection strategy. * denotes different bottleneck architectures.

ALGORITHM	SR	RS	Avg
S. ONLY†	45.26	64.28	54.77
S. ONLY (Ours)	51.86	67.11	59.48
PADA†	53.53	76.50	65.02
PADA (Ours)	49.34	59.81	54.57
SAFN†	67.65	-	-
SAFN (Ours)	56.88	68.40	62.64
BA3US†	69.86	67.56	68.71
BA3US (Ours)	71.77	63.56	67.67
AR†*	85.30	74.82	80.06
AR (Ours)	76.33	71.36	73.85
JUMBOT†	-	-	-
JUMBOT (Ours)	90.55	77.46	84.01
MPOT†	-	-	-
MPOT (Ours)	87.23	86.67	86.95

Table 8: Comparison between reported (†) accuracies on partial VISDA from published methods with our implementation using the ORACLE model selection strategy. * denotes different bottleneck architectures.

normalization layer is added in the bottleneck which requires specifying r, the value to which the 2-norm is set. This hyper-parameter is therefore included in the hyper-parameter grid search with the possible values of $[5, 10, 20]$ which are the different values used in the experiments in (Gu et al., 2021).

D Additional Discussion of Results

D.1 Reproducibility Of Previous Results

In order to trust our results, we need to unsure we that we correctly reimplementated the different PDA methods. We do so by comparing our reproduced results and the previously reported results in

TASK	METHOD	S-ACC	ENT	DEV	SND	1-SHOT	100-RND	ORACLE
SR	S. ONLY	46.96 ± 1.5	48.17 ± 3.9	49.00 ± 0.9	48.17 ± 3.9	49.43 ± 0.8	50.01 ± 1.6	51.86 ± 1.4
	PADA	44.56 ± 5.9	40.83 ± 11.3	41.04 ± 4.3	56.14 ± 9.7	52.94 ± 4.3	49.34 ± 8.4	49.34 ± 8.4
	SAFN	52.04 ± 3.5	29.86 ± 16.7	52.42 ± 2.9	28.46 ± 16.5	49.97 ± 3.3	47.83 ± 0.6	56.88 ± 2.1
	BA3US	44.21 ± 3.0	71.17 ± 1.9	48.78 ± 1.9	46.12 ± 7.8	66.79 ± 1.5	71.45 ± 0.8	71.77 ± 1.1
	AR	$\mathbf{68.39 \pm 1.3}$	75.28 ± 2.9	$\mathbf{68.54 \pm 1.3}$	57.61 ± 0.4	70.11 ± 1.4	75.09 ± 5.2	76.33 ± 4.5
	JUMBOT	55.23 ± 2.3	56.25 ± 2.1	54.35 ± 2.0	75.23 ± 8.4	$\mathbf{81.27 \pm 6.9}$	$\mathbf{89.94 \pm 1.1}$	$\mathbf{90.55 \pm 0.5}$
	MPOT	64.57 ± 2.9	$\mathbf{82.10 \pm 2.0}$	57.02 ± 1.5	$\mathbf{84.45 \pm 0.4}$	71.33 ± 4.4	87.20 ± 2.3	87.23 ± 2.3
RS	S. ONLY	63.34 ± 3.4	62.32 ± 2.7	61.13 ± 3.3	61.88 ± 2.3	62.00 ± 3.9	66.30 ± 2.0	67.11 ± 2.1
	PADA	50.39 ± 3.8	23.80 ± 1.6	45.82 ± 9.2	57.53 ± 10.3	53.36 ± 1.7	59.43 ± 5.8	59.81 ± 6.2
	SAFN	64.37 ± 0.7	55.80 ± 5.2	64.82 ± 0.5	61.19 ± 3.3	63.82 ± 1.0	70.34 ± 5.8	68.40 ± 1.2
	BA3US	65.99 ± 4.6	59.99 ± 1.3	$\mathbf{68.01 \pm 1.9}$	56.01 ± 2.9	62.75 ± 2.6	63.44 ± 1.9	63.56 ± 1.8
	AR	64.97 ± 0.8	53.26 ± 9.7	65.86 ± 3.5	53.78 ± 2.1	70.46 ± 4.7	70.11 ± 5.0	71.36 ± 5.5
	JUMBOT	66.04 ± 1.0	$\mathbf{68.59 \pm 4.6}$	65.36 ± 0.8	$\mathbf{80.16 \pm 1.1}$	$\mathbf{75.42 \pm 4.8}$	77.03 ± 2.7	77.46 ± 3.3
	MPOT	$\mathbf{75.47 \pm 3.8}$	67.18 ± 9.1	66.21 ± 1.2	72.36 ± 7.4	70.58 ± 3.1	$\mathbf{86.18 \pm 8.1}$	$\mathbf{86.67 \pm 7.8}$
Avg	S. ONLY	55.15 ± 2.4	55.24 ± 3.2	55.07 ± 1.2	55.02 ± 2.9	55.72 ± 2.2	58.16 ± 0.6	59.48 ± 0.4
	PADA	47.48 ± 4.8	32.32 ± 4.9	43.43 ± 5.3	56.83 ± 1.0	53.15 ± 2.9	54.38 ± 2.7	54.57 ± 2.6
	SAFN	58.20 ± 1.7	42.83 ± 6.3	58.62 ± 1.3	44.82 ± 8.8	56.89 ± 2.1	59.09 ± 2.8	62.64 ± 1.5
	BA3US	55.10 ± 3.7	65.58 ± 1.4	58.40 ± 1.4	51.07 ± 4.3	64.77 ± 1.4	67.44 ± 1.2	67.67 ± 1.3
	AR	66.68 ± 1.0	64.27 ± 3.6	$\mathbf{67.20 \pm 1.5}$	55.69 ± 0.9	70.29 ± 1.7	72.60 ± 0.8	73.85 ± 0.9
	JUMBOT	60.63 ± 0.7	62.42 ± 2.4	59.86 ± 0.6	77.69 ± 4.2	$\mathbf{78.34 \pm 1.9}$	83.49 ± 1.9	84.01 ± 1.9
	MPOT	$\mathbf{70.02 \pm 2.0}$	$\mathbf{74.64 \pm 4.4}$	61.62 ± 1.3	$\mathbf{78.40 \pm 3.9}$	70.96 ± 3.7	$\mathbf{86.69 \pm 5.1}$	$\mathbf{86.95 \pm 5.0}$

Table 9: Accuracy of different PDA methods based on different model selection strategies on the 2 Partial VISDA tasks. Average is done over three seeds (2020, 2021, 2022). Best results in **bold**.

METRIC	METHOD	AC	AP	AR	CA	CP	CR	PA	PC	PR	RA	RC	RP	Avg
S-ACC	S. ONLY	44.50 ± 1.7	67.71 ± 2.4	78.37 ± 0.3	52.56 ± 0.9	54.81 ± 0.1	62.88 ± 0.9	58.77 ± 0.5	39.28 ± 0.8	75.08 ± 0.5	68.90 ± 0.6	45.33 ± 1.0	76.34 ± 0.7	60.38 ± 0.5
	PADA	50.15 ± 2.8	66.93 ± 1.2	76.73 ± 1.7	58.00 ± 1.4	56.13 ± 1.4	66.45 ± 0.8	60.33 ± 2.1	43.50 ± 1.2	76.70 ± 0.4	69.27 ± 3.5	53.93 ± 1.3	78.88 ± 0.8	63.08 ± 0.3
	SAFN	47.36 ± 0.1	66.82 ± 1.9	77.62 ± 0.2	57.85 ± 0.6	57.89 ± 0.7	66.92 ± 0.9	58.80 ± 0.7	42.49 ± 0.6	75.46 ± 0.4	67.92 ± 0.0	49.73 ± 0.1	76.23 ± 0.8	62.09 ± 0.2
	BA3US	$\mathbf{54.89 \pm 4.7}$	71.34 ± 0.8	$\mathbf{81.91 \pm 3.9}$	61.68 ± 5.2	$\mathbf{67.13 \pm 3.9}$	$\mathbf{72.96 \pm 1.0}$	$\mathbf{68.90 \pm 5.0}$	$\mathbf{55.92 \pm 1.3}$	$\mathbf{79.13 \pm 4.7}$	$\mathbf{72.27 \pm 3.5}$	51.84 ± 0.5	$\mathbf{81.85 \pm 4.1}$	$\mathbf{68.32 \pm 1.1}$
	AR	51.12 ± 1.2	$\mathbf{72.79 \pm 0.7}$	77.91 ± 0.2	$\mathbf{63.21 \pm 1.5}$	60.54 ± 4.0	72.76 ± 0.9	63.39 ± 3.1	48.36 ± 1.7	78.02 ± 1.7	70.00 ± 1.1	52.52 ± 1.0	77.55 ± 2.6	65.68 ± 0.3
	JUMBOT	49.07 ± 0.2	65.45 ± 0.4	77.14 ± 0.3	60.09 ± 0.1	59.59 ± 1.3	66.67 ± 1.3	60.24 ± 1.0	43.60 ± 0.0	74.43 ± 0.9	70.19 ± 0.5	51.12 ± 1.1	77.12 ± 1.3	62.89 ± 0.2
	MPOT	53.07 ± 0.3	72.61 ± 1.2	78.50 ± 0.7	61.92 ± 0.5	64.16 ± 1.8	70.22 ± 0.2	64.13 ± 0.9	50.87 ± 1.1	77.40 ± 0.1	70.40 ± 0.6	$\mathbf{53.99 \pm 1.5}$	77.61 ± 0.3	66.24 ± 0.1
ENT	S. ONLY	45.27 ± 1.1	68.91 ± 1.4	79.26 ± 0.7	54.21 ± 2.1	55.52 ± 0.6	63.19 ± 0.3	56.96 ± 1.5	38.75 ± 0.6	75.65 ± 1.3	69.24 ± 1.0	45.31 ± 1.0	76.47 ± 0.8	60.73 ± 0.2
	PADA	46.03 ± 2.9	62.09 ± 2.8	76.05 ± 1.4	55.07 ± 2.7	47.28 ± 0.1	60.92 ± 2.4	56.69 ± 2.8	38.43 ± 3.0	77.08 ± 0.2	69.48 ± 1.3	49.73 ± 3.5	78.00 ± 1.7	59.74 ± 0.5
	SAFN	47.08 ± 2.0	66.83 ± 0.5	77.73 ± 0.2	56.54 ± 2.2	59.07 ± 0.7	66.22 ± 0.5	56.75 ± 2.1	39.58 ± 2.0	73.90 ± 0.9	67.80 ± 0.2	48.76 ± 0.1	76.23 ± 0.7	61.37 ± 0.3
	BA3US	$\mathbf{59.26 \pm 0.9}$	76.38 ± 1.5	$\mathbf{86.03 \pm 0.6}$	68.96 ± 1.8	71.07 ± 0.8	76.22 ± 1.2	$\mathbf{73.16 \pm 0.6}$	57.91 ± 2.5	$\mathbf{85.59 \pm 1.2}$	78.11 ± 1.4	$\mathbf{62.85 \pm 2.7}$	84.84 ± 0.6	73.36 ± 0.6
	AR	54.91 ± 1.8	$\mathbf{78.45 \pm 1.8}$	84.23 ± 0.9	64.86 ± 2.3	68.16 ± 3.5	$\mathbf{80.45 \pm 0.8}$	67.58 ± 0.4	52.34 ± 1.0	82.48 ± 1.9	74.75 ± 2.1	55.64 ± 1.2	83.06 ± 1.2	70.58 ± 0.4
	JUMBOT	57.69 ± 5.6	75.44 ± 1.4	85.24 ± 2.7	$\mathbf{75.97 \pm 1.4}$	$\mathbf{74.85 \pm 3.3}$	79.75 ± 1.2	72.85 ± 2.4	$\mathbf{60.18 \pm 0.9}$	83.21 ± 1.1	$\mathbf{81.97 \pm 1.0}$	61.81 ± 4.6	$\mathbf{86.33 \pm 1.6}$	$\mathbf{74.61 \pm 0.8}$
	MPOT	52.94 ± 2.0	68.94 ± 1.2	75.98 ± 0.6	60.58 ± 0.8	65.99 ± 2.2	71.51 ± 0.8	58.28 ± 0.9	49.87 ± 2.6	73.77 ± 1.3	64.98 ± 0.4	57.53 ± 0.6	73.17 ± 2.7	64.46 ± 0.1
DEV	S. ONLY	43.74 ± 1.8	67.81 ± 1.2	78.28 ± 0.7	51.42 ± 2.7	54.55 ± 1.2	63.94 ± 1.7	57.94 ± 0.9	39.40 ± 0.9	74.91 ± 0.6	69.27 ± 1.0	45.33 ± 1.0	75.99 ± 1.3	60.22 ± 0.3
	PADA	44.70 ± 1.3	61.61 ± 5.4	68.99 ± 11.3	35.08 ± 13.1	24.24 ± 20.5	61.66 ± 2.4	57.91 ± 1.7	38.03 ± 0.6	73.11 ± 3.4	66.33 ± 0.7	29.97 ± 21.0	71.07 ± 11.3	52.72 ± 2.8
	SAFN	48.12 ± 0.4	67.30 ± 0.5	77.43 ± 0.5	56.75 ± 0.3	58.17 ± 1.2	65.64 ± 1.3	59.08 ± 0.5	43.00 ± 1.1	74.64 ± 0.4	68.11 ± 0.9	50.53 ± 0.6	75.65 ± 0.5	62.03 ± 0.4
	BA3US	41.67 ± 18.9	50.05 ± 28.7	63.74 ± 26.1	60.70 ± 2.2	59.08 ± 10.9	67.88 ± 0.9	$\mathbf{64.62 \pm 1.6}$	$\mathbf{56.74 \pm 1.3}$	75.21 ± 0.6	$\mathbf{70.92 \pm 2.0}$	$\mathbf{58.39 \pm 2.3}$	$\mathbf{78.06 \pm 1.3}$	62.25 ± 7.1
	AR	$\mathbf{49.25 \pm 2.8}$	$\mathbf{70.20 \pm 1.7}$	$\mathbf{79.73 \pm 2.5}$	$\mathbf{62.72 \pm 1.0}$	$\mathbf{61.85 \pm 4.6}$	$\mathbf{70.86 \pm 5.6}$	61.65 ± 1.0	43.72 ± 0.7	$\mathbf{76.29 \pm 0.7}$	70.31 ± 1.7	49.61 ± 0.8	75.61 ± 0.4	$\mathbf{64.32 \pm 0.9}$
	JUMBOT	46.11 ± 0.1	66.33 ± 0.6	76.42 ± 0.3	56.81 ± 0.1	56.36 ± 0.5	66.70 ± 0.8	58.03 ± 1.1	41.99 ± 0.8	74.97 ± 0.5	67.43 ± 0.3	48.12 ± 0.5	76.04 ± 0.1	61.28 ± 0.1
	MPOT	46.07 ± 0.7	65.43 ± 0.8	76.46 ± 0.4	56.44 ± 1.0	57.95 ± 1.0	66.35 ± 1.0	57.64 ± 0.8	43.60 ± 0.6	74.86 ± 1.3	67.68 ± 0.5	48.12 ± 0.8	75.89 ± 0.4	61.37 ± 0.2
SND	S. ONLY	42.23 ± 1.3	68.91 ± 1.4	79.35 ± 0.6	51.76 ± 3.7	53.48 ± 2.1	63.94 ± 1.7	55.37 ± 0.6	37.35 ± 1.0	74.10 ± 2.8	68.53 ± 1.4	43.78 ± 0.6	75.84 ± 1.6	59.55 ± 0.3
	PADA	50.43 ± 0.8	66.72 ± 1.5	79.72 ± 1.8	57.30 ± 1.9	52.10 ± 1.7	63.11 ± 1.9	60.82 ± 3.0	39.26 ± 2.0	79.33 ± 1.3	73.09 ± 1.5	45.77 ± 1.6	80.62 ± 0.4	62.36 ± 0.4
	SAFN	49.57 ± 0.3	68.18 ± 1.3	77.86 ± 0.5	57.91 ± 0.3	58.17 ± 1.2	66.13 ± 1.0	59.14 ± 0.8	43.90 ± 0.5	75.81 ± 0.7	68.17 ± 1.6	49.59 ± 1.6	76.64 ± 0.5	62.59 ± 0.1
	BA3US	$\mathbf{62.21 \pm 0.9}$	$\mathbf{83.29 \pm 0.4}$	$\mathbf{88.50 \pm 0.6}$	68.50 ± 0.9	71.45 ± 3.6	76.96 ± 0.6	$\mathbf{76.19 \pm 1.2}$	$\mathbf{59.94 \pm 1.7}$	$\mathbf{86.31 \pm 1.4}$	79.46 ± 1.4	$\mathbf{65.35 \pm 1.9}$	86.35 ± 0.9	$\mathbf{75.37 \pm 0.8}$
	AR	54.37 ± 1.6	79.01 ± 2.2	84.54 ± 0.8	64.52 ± 1.6	68.05 ± 3.2	79.16 ± 2.8	65.60 ± 1.7	51.28 ± 1.6	83.05 ± 1.1	75.02 ± 1.6	55.02 ± 1.8	83.40 ± 0.9	70.25 ± 0.2
	JUMBOT	56.60 ± 2.8	68.48 ± 1.5	84.70 ± 2.1	$\mathbf{71.81 \pm 1.8}$	$\mathbf{71.84 \pm 1.6}$	$\mathbf{80.91 \pm 0.9}$	70.28 ± 0.8	50.69 ± 4.9	83.89 ± 1.5	$\mathbf{81.21 \pm 0.6}$	58.85 ± 1.7	$\mathbf{88.18 \pm 0.4}$	72.29 ± 0.2
	MPOT	32.96 ± 0.4	49.73 ± 1.1	57.39 ± 1.4	44.11 ± 2.4	38.66 ± 1.2	50.06 ± 1.0	43.74 ± 4.3	28.66 ± 2.6	58.40 ± 1.9	56.90 ± 1.9	39.34 ± 1.2	63.14 ± 0.6	46.92 ± 0.4
1-SHOT	S. ONLY	43.84 ± 1.7	66.52 ± 3.1	77.38 ± 0.9	50.47 ± 2.4	53.24 ± 2.0	61.77 ± 1.1	56.11 ± 1.7	37.35 ± 1.0	71.97 ± 1.8	68.96 ± 0.5	46.13 ± 2.0	73.33 ± 2.2	58.92 ± 0.4
	PADA	52.98 ± 0.2	63.03 ± 1.6	78.06 ± 2.6	51.67 ± 5.0	56.28 ± 0.4	64.00 ± 1.4	58.92 ± 3.3	43.62 ± 1.0	74.27 ± 4.1	68.26 ± 3.1	54.25 ± 1.6	78.62 ± 0.4	62.00 ± 0.5
	SAFN	31.40 ± 3.7	49.73 ± 4.3	62.82 ± 2.0	48.88 ± 2.4	45.27 ± 0.7	57.26 ± 2.2	42.33 ± 1.6	29.77 ± 2.6	63.52 ± 3.2	56.11 ± 3.2	37.55 ± 0.8	67.00 ± 1.4	49.30 ± 0.7
	BA3US	44.60 ± 21.0	51.39 ± 29.8	65.47 ± 27.2	65.63 ± 1.4	59.78 ± 15.3	68.49 ± 1.3	68.38 ± 1.7	57.83 ± 1.3	$\mathbf{82.05 \pm 1.0}$	$\mathbf{80.78 \pm 1.1}$	$\mathbf{63.10 \pm 0.8}$	79.20 ± 1.1	65.56 ± 7.6
	AR	56.00 ± 2.3	$\mathbf{78.58 \pm 1.9}$	82.77 ± 2.0	68.99 ± 0.2	68.35 ± 1.9	77.25 ± 1.4	69.67 ± 1.5	51.98 ± 1.8	78.72 ± 1.0	76.19 ± 0.7	55.48 ± 2.1	82.73 ± 1.0	70.56 ± 0.7
	JUMBOT	$\mathbf{61.59 \pm 1.7}$	76.86 ± 3.4	$\mathbf{86.45 \pm 2.1}$	$\mathbf{74.20 \pm 0.9}$	$\mathbf{73.43 \pm 3.3}$	$\mathbf{79.85 \pm 0.3}$	$\mathbf{74.96 \pm 3.4}$	$\mathbf{62.87 \pm 0.6}$	81.83 ± 0.9	78.48 ± 2.0	61.59 ± 2.2	$\mathbf{87.34 \pm 0.2}$	$\mathbf{74.95 \pm 0.1}$
	MPOT	53.97 ± 1.3	68.78 ± 1.7	78.04 ± 2.1	69.24 ± 0.4	65.88 ± 0.5	71.42 ± 0.7	70.31 ± 1.0	53.03 ± 0.7	76.88 ± 1.3	76.52 ± 0.4	57.39 ± 1.7	77.95 ± 1.4	68.28 ± 0.2
100-RND	S. ONLY	43.28 ± 1.6	68.76 ± 1.6	77.97 ± 1.2	53.75 ± 1.1	55.57 ± 2.2	63.94 ± 0.4	58.37 ± 0.4	39.12 ± 0.4	75.56 ± 1.3	69.02 ± 0.5	43.46 ± 0.2	75.28 ± 2.3	60.34 ± 0.4
	PADA	50.41 ± 0.8	67.21 ± 1.8	79.97 ± 1.5	56.69 ± 1.5	53.86 ± 1.6	63.94 ± 1.3	60.27 ± 2.7	40.56 ± 1.8	78.91 ± 1.8	72.70 ± 1.4	53.39 ± 2.2	80.73 ± 0.9	63.22 ± 0.1
	SAFN	47.58 ± 0.8	67.53 ± 0.8	77.91 ± 0.4	56.47 ± 1.0	58.19 ± 0.4	65.88 ± 0.2	59.69 ± 0.1	43.14 ± 1.7	75.00 ± 0.7	69.64 ± 1.0	50.85 ± 0.3	76.41 ± 0.8	62.36 ± 0.2
	BA3US	$\mathbf{62.53 \pm 2.0}$	$\mathbf{82.09 \pm 0.8}$	$\mathbf{88.28 \pm 0.4}$	69.15 ± 1.2	71.65 ± 1.5	77.21 ± 0.6	$\mathbf{75.15 \pm 1.3}$	58.17 ± 1.0	$\mathbf{85.92 \pm 1.3}$	79.86 ± 2.1	$\mathbf{66.57 \pm 1.5}$	85.66 ± 1.0	75.19 ± 0.4
	AR	54.89 ± 2.0	78.54 ± 1.4	84.34 ± 0.6	64.95 ± 2.4	69.00 ± 3.7	79.57 ± 0.2	66.73 ± 0.3	50.85 ± 1.4	82.39 ± 1.9	74.66 ± 2.3	55.42 ± 1.6	82.80 ± 0.4	70.34 ± 0.2
	JUMBOT	61.07 ± 0.9	77.87 ± 1.4	86.01 ± 1.3	$\mathbf{74.56 \pm 0.4}$	$\mathbf{76.40 \pm 1.4}$	$\mathbf{81.54 \pm 1.7}$	72.60 ± 1.2	$\mathbf{59.92 \pm 0.4}$	84.63 ± 2.3	$\mathbf{81.85 \pm 1.7}$	64.84 ± 1.0	$\mathbf{87.64 \pm 0.7}$	$\mathbf{75.74 \pm 0.3}$
	MPOT	61.59 ± 1.2	75.56 ± 1.7	82.59 ± 0.6	72.48 ± 1.0	69.77 ± 0.9	75.41 ± 0.5	72.64 ± 0.9	57.67 ± 1.6	82.02 ± 0.6	79.80 ± 0.5	64.64 ± 0.1	82.60 ± 0.5	73.06 ± 0.3
ORACLE	S. ONLY	45.43 ± 0.9	68.91 ± 1.4	79.53 ± 0.3	55.59 ± 0.7	57.42 ± 1.2	65.23 ± 0.8	59.32 ± 0.7	40.80 ± 0.9	75.80 ± 1.2	69.88 ± 0.9	47.20 ± 0.9	77.31 ± 0.1	61.87 ± 0.3
	PADA	50.53 ± 0.7	67.45 ± 1.6	80.14 ± 1.4	57.30 ± 1.9	54.47 ± 1.7	64.55 ± 1.1	61.07 ± 3.0	40.94 ± 1.6	79.55 ± 1.4	73.09 ± 1.5	54.63 ± 0.9	80.93 ± 0.6	63.72 ± 0.3
	SAFN	49.57 ± 0.3	68.55 ± 1.0	78.26 ± 0.2	57.91 ± 0.3	59.29 ± 0.5	66.81 ± 0.5	59.87 ± 0.7	45.29 ± 0.7	75.98 ± 0.6	69.08 ± 0.6	51.68 ± 0.8	77.29 ± 0.5	63.30 ± 0.2
	BA3US	63.26 ± 1.0	$\mathbf{82.75 \pm 0.9}$	$\mathbf{89.16 \pm 0.2}$	69.91 ± 0.2	71.93 ± 1.6	77.58 ± 0.9	75.73 ± 1.3	59.94 ± 0.7	$\mathbf{86.89 \pm 0.5}$	80.93 ± 0.8	66.77 ± 1.5	86.93 ± 0.2	75.98 ± 0.3
	AR	57.33 ± 1.7	79.61 ± 1.6	86.31 ± 0.4	69.45 ± 0.5	71.88 ± 0.9	79.94 ± 0.8	70.28 ± 1.0	53.57 ± 0.2	83.78 ± 1.0	77.26 ± 0.6	59.68 ± 1.1	83.72 ± 0.6	72.73 ± 0.3
	JUMBOT	61.87 ± 1.4	78.19 ± 2.4	88.11 ± 1.5	$\mathbf{77.69 \pm 0.1}$	76.75 ± 0.8	$\mathbf{84.15 \pm 1.3}$	$\mathbf{76.83 \pm 1.9}$	63.72 ± 0.5	84.80 ± 1.3	$\mathbf{81.79 \pm 0.8}$	64.70 ± 1.1	$\mathbf{87.17 \pm 1.7}$	77.15 ± 0.4
	MPOT	$\mathbf{64.48 \pm 1.2}$	80.88 ± 3.3	86.78 ± 0.5	76.22 ± 0.1	$\mathbf{77.95 \pm 1.3}$	82.59 ± 0.7	75.18 ± 1.3	$\mathbf{64.60 \pm 0.0}$	84.87 ± 1.4	80.59 ± 0.6	$\mathbf{67.04 \pm 0.6}$	86.52 ± 1.2	$\mathbf{77.31 \pm 0.5}$

Table 10: Average accuracy of different PDA methods based on different model selection strategies on the 12 tasks of Partial OFFICE-HOME. Average is done over three seeds (2020, 2021, 2022). Best results in **bold**.

Table 7. On OFFICE-HOME, both PADA and JUMBOT achieved higher average task accuracy (1.6 and 1.7 percentage points, respectively) in our reimplementation, while for BA3US and MPOT we recover the reported accuracy in their respective papers. However, we saw a decrease in performance for both SAFN and AR of roughly 8 and 5 percentage points respectively. This is to be expected due to the

DATASET	METHOD	S-ACC	ENT	DEV	SND	1-SHOT	50-RND	100-RND	ORACLE
OFFICE-HOME	S. ONLY	60.38±0.5	60.73±0.2	60.22±0.3	59.55±0.3	58.92±0.4	60.28±0.4	60.34±0.4	61.87±0.3
	PADA	63.08±0.3	59.74±0.5	52.72±2.8	62.36±0.4	62.00±0.5	63.82±0.4	63.22±0.1	63.72±0.3
	SAFN	62.09±0.2	61.37±0.3	62.03±0.4	62.59±0.1	49.30±0.7	62.00±0.2	62.36±0.2	63.30±0.2
	BA3US	**68.32±1.1**	73.36±0.6	62.25±7.1	**75.37±0.8**	65.56±7.6	**73.22±0.3**	75.19±0.4	75.98±0.3
	AR	65.68±0.3	70.58±0.4	**64.32±0.9**	70.25±0.2	70.56±0.7	70.26±0.2	70.34±0.2	72.73±0.3
	JUMBOT	62.89±0.2	**74.61±0.8**	61.28±0.1	72.29±0.2	**74.95±0.1**	64.95±0.3	**75.74±0.3**	77.15±0.4
	MPOT	66.24±0.1	64.46±0.1	61.37±0.2	46.92±0.4	68.28±0.2	69.90±0.5	73.06±0.3	**77.31±0.5**
VISDA	S. ONLY	55.15±2.4	55.24±3.2	55.07±1.2	55.02±2.9	55.72±2.2	57.90±1.1	58.16±0.6	59.48±0.4
	PADA	47.48±4.8	32.32±4.9	43.43±5.3	56.83±1.0	53.15±2.9	55.67±2.5	54.38±2.7	54.57±2.6
	SAFN	58.20±1.7	42.83±6.3	58.62±1.3	44.82±8.8	56.89±2.1	57.90±3.3	59.09±2.8	62.64±1.5
	BA3US	55.10±3.7	65.58±1.4	58.40±1.4	51.07±4.3	64.77±1.4	66.66±2.4	67.44±1.2	67.67±1.3
	AR	66.68±1.0	64.27±3.6	**67.20±1.5**	55.69±0.9	70.29±1.7	71.91±0.3	72.60±0.8	73.85±0.9
	JUMBOT	60.63±0.7	62.42±2.4	59.86±0.6	77.69±4.2	**78.34±1.9**	82.85±2.9	83.49±1.9	84.01±1.9
	MPOT	**70.02±2.0**	**74.64±4.4**	61.62±1.3	**78.40±3.9**	70.96±3.7	**86.65±5.1**	**86.69±5.1**	**86.95±5.0**

Table 11: Task accuracy average for the different PDA methods and model selection strategy pairs on Partial Office-Home and Partial VisDA. The average is computed over three difference seeds (2020, 2021, 2022).

differences in the neural network architectures. While we use a linear bottleneck layer, SAFN use a nonlinear bottleneck layer. As for AR, they make two significant changes: the linear classification head is replaced by a spherical logistic regression (SLR) layer (Gu et al., 2020) and the features are normalized (the 2-norm is set to a dataset dependent value, another hyper-parameter that requires tuning) before feeding them to the classification head. While we account for the first change by comparing to the AR (w/ linear) results reported in (Gu et al., 2021), in our neural network architecture we do not normalize the features. These changes, nonlinear bottleneck layer for SAFN and feature normalization for AR, significantly boost the performance of both methods. When now comparing our reimplementation with the same neural network architectures, our SAFN reimplementation achieves a higher average task accuracy by 3 percentage points, while our AR reimplementation is now only 1 percentage points below. The fact that AR reported results are from only one run, while ours are averaged across 3 distinct seeds, justifies the small remaining gap. Moreover, we report higher accuracy or on par on 4 tasks of the 12 tasks.

In Table 8, we compare previously reported results with ours on VISDA. While proposed methods reported results on OFFICE-HOME, only PADA and AR results are reported in the original papers for VISDA. Gu et al. (2021) AR) also report results for BA3US. Analysing the results, we see a 9 percentage point decrease in average task accuracy for PADA, but our experiments show that there is a significant seed dependence which we discuss in detail below. This is particularly important since Cao et al. (2018) (PADA) report results from a single run. Comparing our best seeds for PADA on the SR and RS tasks, we achieve 58.01% and 67.9% accuracy versus a reported 53.53% and 76.5%. Moreover, we point out that the official code repository for PADA does not include the details to reproduce the VISDA experiments, so it is possible that minor tweaks (e.g learning rate) are necessary. As for BA3US, our results are within the standard deviation being better on the SR task and worse on the RS task. Finally as for AR we see a decrease in performance which, as the results on OFFICE-HOME show, can be explained by the differences in the neural network architecture.

Given all the above results, our reimplementations are trustworthy and give validity to our results.

D.2 Additional Results

In this section, we provide additional results that we could not add to the main paper due to the space constraints. In Table 9 and Table 10, we show the accuracy per task on OFFICE-HOME and VISDA respectively, averaged over three different seeds (2020, 2021, 2022) for all pairs of methods and model selection strategies. Finally in Table 11, we show all the average task accuracies from all pairs of methods and model selection strategies on the OFFICE-HOME and VISDA datasets including the 50-RND model selection strategy.