UDA-Bench: Revisiting Common Assumptions in Unsupervised Domain Adaptation Using a Standardized Framework

Supplementary Material

Tarun Kalluri, Sreyas Ravichandran, and Manmohan Chandraker

UC San Diego
https://github.com/ViLab-UCSD/UDABench_ECCV2024

1 UDABench: Code Overview

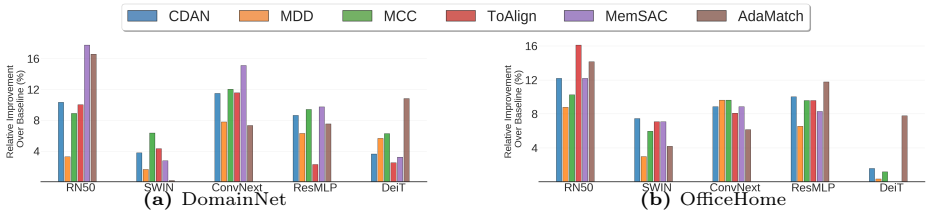
We build our codebase using PyTorch following several open-source deep-learning libraries like Detectron [8] and PyTorch3D [5]. The overarching motivation in designing UDA-Bench is to standardize evaluation and training of existing unsupervised adaptation methods to facilitate fair comparative studies like ours, while also enabling quick prototyping and design of new adaptation methods in the future. UDA-Bench is designed to be flexible to incorporate newer architecture backbones, classifier modules, optimizers, loss functions, dataloaders and training methods with minimal effort and design overhead, allowing researchers to build upon existing adaptation methods to develop new innovations in unsupervised adaptation.

We re-implement several classical as well as state-of-the-art UDA methods in UDA-Bench. We keep the adaptation independent hyper-parameters (such as architectures, batch sizes) same across the methods, and use the adaptation-specific hyper-parameters as recommended in the respective methods. We use the open-source repositories of prior UDA methods from the links given below.

- **CDAN:** https://github.com/thuml/CDAN/tree/master
- **MCC:** https://github.com/thuml/Versatile-Domain-Adaptation
- **MDD:** https://github.com/thuml/MDD
- **ToAlign:** https://github.com/microsoft/UDA
- **MemSAC:** https://github.com/ViLab-UCSD/MemSAC_ECCV2022
- **AdaMatch:** https://github.com/google-research/adamatch
- **DALN:** https://github.com/xiaoachen98/DALN
- **PMTrans:** https://github.com/JinjingZhu/PMTrans

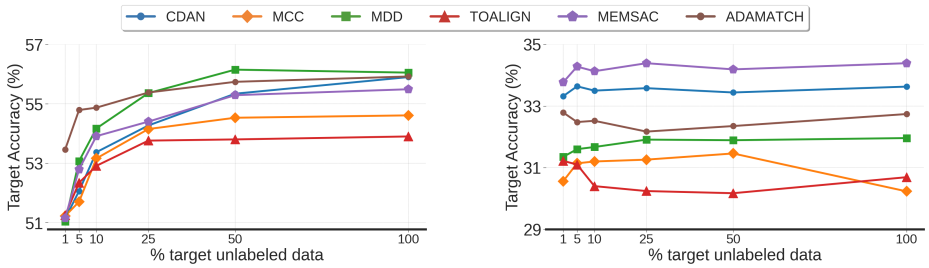
2 Additional Results on DomainNet and OfficeHome

Effect of Backbone Architecture We examine the effect caused due to backbone on further settings from the DomainNet dataset in Fig. 1a and OfficeHome in Fig. 1b, where we show the target accuracy on each dataset. We observe same trends as discussed in main paper with more focused transfer settings, with vision



(a) DomainNet **(b)** OfficeHome

Fig. 1: Effect of backbone. For each of the UDA methods, we show the gain in accuracy relative to a baseline trained only using source-data. The average accuracy across 12 tasks in (a) DomainNet and (b) OfficeHome.



(a) DomainNet **(b)** GeoPlaces (USA→Asia)

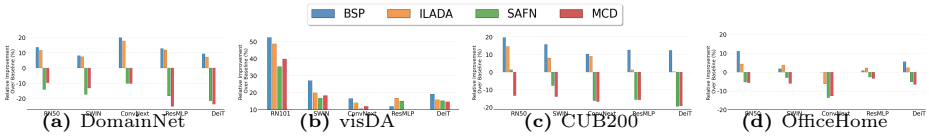
Fig. 2: Effect of unlabeled data We show the effect of target unlabeled data on the target accuracy - (a) average of 12 tasks on DomainNet and (b) GeoPlaces using a DeiT Backbone. The trends remain similar, where we observe that most UDA methods under-utilize unlabeled data.

transformer architecture like Swin and Deit diminishing the benefits of most UDA methods, that otherwise yield good gains with Resnet-50 as the backbone.

Amount of Unlabeled Data As demonstrated in main paper, current UDA methods under-utilize unlabeled data, and the performance saturates even when more unlabeled data is accessible to the algorithms. We examine this trend for other settings in DomainNet dataset as well, and show the average accuracy across 12 tasks from DomainNet in Fig. 2 using a DeiT backbone. We use a DeiT backbone for this comprehensive experiment since it converges faster and hence needs lesser GPU hours during training. We also show the scaling trends for adaptation another completely different kind of dataset GeoPlaces [4] in Fig. 2b, where we observe that unlabeled data rarely helps, even hurting the adaptation accuracy in some cases.

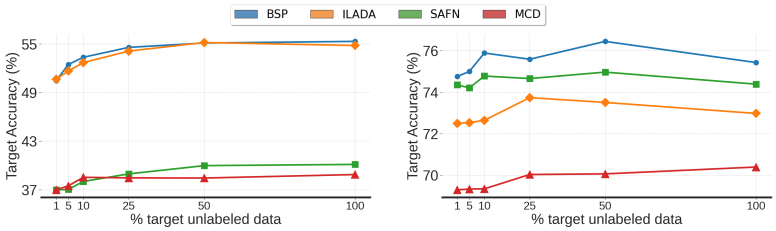
3 Results Using Additional UDA Methods

In addition to the wide variety of UDA methods studied in our main paper, we show results using four additional adaptation methods: BSP [1], ILADA [7],



(a) DomainNet (b) visDA (c) CUB200 (d) OfficeHome

Fig. 3: Newer backbones give limited returns or perform worse than baseline. For each of the UDA methods, we show the gain in accuracy relative to a baseline trained only using source-data. For methods like SAFN [9] and MCD [6], we observe that the relative improvement over a source-only baseline is negative in most cases. Further, the gains observed by other methods like BSP [1] and ILADA [7] are not same across architectures.



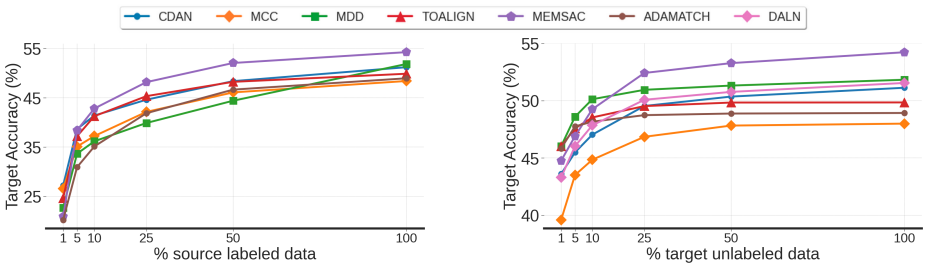
(a) DomainNet (b) VisDA

Fig. 4: Unlabeled Data-efficiency of UDA algorithms Across both DomainNet and visDA datasets, the performance of UDA methods exhibits diminishing returns with increasing amounts of unlabeled data. In most cases, utilizing only 25% of the available unlabeled data results in a performance drop of less than 1%, suggesting that collecting additional unlabeled data is unlikely to yield significant improvements for these methods.

SAFN [9] and MCD [6]. The observations for the effect of backbone architecture is presented in Fig. 3 and the study for the effect of unlabeled target domain data is presented in Fig. 4.

UDA methods are not always compatible with newer backbones. Resonating with the observations made in the main paper, we show in Fig. 3 that the gains obtained by UDA method are not independent of the backbone. For instance, on CUB200 dataset, BSP [1] and ILADA [7] gives 20% and 15% relative gain respectively, but using DeiT diminishes these gains to 12% and 3% respectively. Similarly, on visDA, the improvements using ResNet is much higher than improvements offered on other backbones like ConvNext and DeiT. Moreover, as demonstrated in previous research [3], other unsupervised domain adaptation (UDA) algorithms, such as SAFN [9] and MCD [6], under-perform compared to a source-only baseline, and the disparity worsens when employing these algorithms with newer architectures.

Adding More Unlabeled Data is Not Beneficial for UDA From Fig. 4, the performance of the additional adaptation methods studied also plateaus quickly, reaching near saturation after utilizing only 20% of the available unlabeled data.



(a) Source Labeled Data **(b)** Target Unlabeled Data

Fig. 5: Source labels vs. Target unsupervised data We show that collecting more labels from source dataset, even when it is from a different domain, has a more profound influence on the target accuracy (a) compared to collecting more unlabeled data from the target domain using current UDA methods (b). Results shown on Real→Clipart setting from DomainNet dataset.

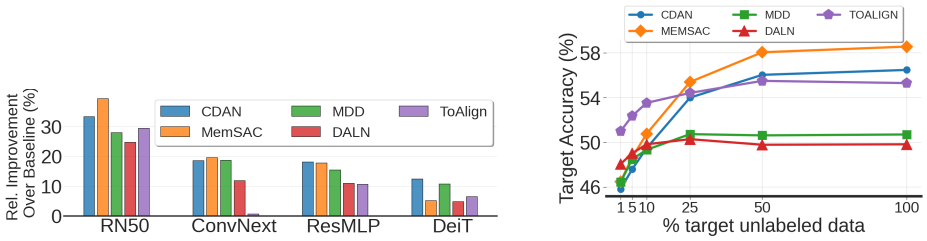
Further addition of unlabeled data yields negligible performance gains. This suggests that collecting additional unlabeled data is unlikely to yield significant improvements for these methods, corroborating the observations noted in the main paper for several other UDA methods.

4 Source Labeled vs. Target Unlabeled Data

In the main paper, we showed that volume of target data has minimal effect on the target accuracy after a certain point. To compare this with the importance held by source labels in determining the target accuracy, we conduct an experiment by using subsets of source labeled data, while using the full target unlabeled data each time. Specifically, we use $\{1, 5, 10, 25, 50, 100\}$% of source labels and train the UDA methods on each subset. We run three random seeds and plot the mean accuracy in Fig. 5. We observe that the scaling trends of target accuracy with respect to source labeled data are much more favorable towards improving performance. For example, doubling the number of source labels from 50% to 100% improves target accuracy by ~ 9% on average across UDA methods. In contrast, the improvement in doubling the target unlabeled data from 50% to 100% is less than 0.5% on average. This confirms the fact that labels have a more pronounced impact on target accuracy even when they arise from a different domain, compared to unlabeled data from the same domain.

5 Results using TinyImageNet

To further examine the presented trends on non-standard adaptation datasets, we show results using images from the TinyImageNet dataset as the source domain and *snow* perturbations from TinyImageNet-C [2] as the target domain. We train



(a) Varying backbones **(b)** Varying target unlabeled data

Fig. 6: Results on TinyImageNet vs. TinyImageNet-C We show the similar observations regarding backbone architectures and data volume hold also for a non-standard adaptation dataset. We use images from TinyImageNet as the source and *snow-3* perturbations from TinyImageNet-C as the target.

models using the 200 classes in each dataset, and use report accuracy on the target domain. In Fig. 6, we show that the broad trends observed for other adaptation datasets also hold for this novel setting. Specifically, from a, adaptation gains are much lesser with recent architectures (like *ConvNext* and *DeiT*) and from b, performance saturates in-spite of adding unlabeled data, further corroborating the main inferences from our study.

6 Training Details

Architecture-specific training details In our ablation on benchmarking UDA across architectures, we use all pre-trained checkpoints from the timm library, and all of them are pre-trained on ImageNet-1k. Across the architectures, we uniformly use a batch size of 32, SGD optimizer with an initial learning rate of 0.003 and cosine decay. It might be possible that ViT models benefit from other algorithms such as Adam [10], which we do not explore in this paper. For data augmentation, we first resize the images so that the shorter size is 256 and then choose a random 224×224 crop followed by random horizontal flip. However, we use a crop size of 256 instead of 224 for Swin transformer due to its input size. We train the networks for a total of 75k iterations on DomainNet and CUB200 with validation performed at every 5k steps, and for 30k iterations on the smaller OfficeHome dataset with validation at every 500 steps. We use early stopping on the test set to choose the best accuracy.

For the classifier, we use a 2-layer MLP with a hidden dimension of 256. The input dimension for the MLP, though, varies depending on the output dimension of the backbone architecture used. For Resnet-50, it is 2048, for Swin-t and ConvNext-t it is 768 and for Deit-s and ResMLP-s it is 384.

6.1 Unsupervised Pre-Training Network Details

We use the official repositories for SwAV, MoCo-v3, MAE to pre-train the models on our datasets. Note that we subsample an image set of 1M images from ImageNet, Places205 and iNat2021 to normalize the effects of data volume, using a per-class sampling strategy. We use the official repositories for Swav, MoCo-V3 and MAE, and use the code for supervised pre-training from PyTorch. We train Swav for 150 epochs, MoCo-v3 for 250 epochs, MAE for 400 epochs and supervised pre-training for 90 epochs. The training for all the methods is performed on 8 GPUs with a total batch size of 1024 in each case. For all other hyperparameters, we follow the ones recommended in the respective repositories.

References

1. Chen, X., Wang, S., Long, M., Wang, J.: Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In: International conference on machine learning. pp. 1081–1090. PMLR (2019)
2. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261 (2019)
3. Kalluri, T., Sharma, A., Chandraker, M.: Memsac: Memory augmented sample consistency for large scale domain adaptation. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXX. pp. 550–568. Springer (2022)
4. Kalluri, T., Xu, W., Chandraker, M.: Geonet: Benchmarking unsupervised adaptation across geographies. arXiv preprint arXiv:2303.15443 (2023)
5. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Accelerating 3d deep learning with pytorch3d. arXiv:2007.08501 (2020)
6. Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3723–3732 (2018)
7. Sharma, A., Kalluri, T., Chandraker, M.: Instance level affinity-based transfer for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5361–5371 (2021)
8. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. `https://github.com/facebookresearch/detectron2` (2019)
9. Xu, R., Li, G., Yang, J., Lin, L.: Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1426–1435 (2019)
10. Zhang, J., Karimireddy, S.P., Veit, A., Kim, S., Reddi, S., Kumar, S., Sra, S.: Why are adaptive methods good for attention models? Advances in Neural Information Processing Systems **33**, 15383–15393 (2020)