MemSAC: Memory Augmented Sample Consistency for Large Scale Domain Adaptation
Supplementary Material

Tarun Kalluri, Astuti Sharma, and Manmohan Chandraker

University of California San Diego, La Jolla CA 92093, USA
`{sskallur,asharma,mkchandraker}@eng.ucsd.edu`

1 Results on DomainNet-126

In the main paper, we choose the complete 345-class split of DomainNet to report the results. In Tab. 1, we show that the benefits using MemSAC persist even on the split with 126 classes which has much lesser label noise, as proposed in prior works like [6, 9, 16]. For this experiment, we choose the recommended train-test splits in the official DomainNet website for the domains real (**R**), clipart (**C**), sketch (**S**) and painting (**P**). MemSAC achieves an accuracy of 64.76% classes, which is 3% larger than the next best approach, PAN (61.75%).

Source	**Real→**			**Clipart→**			**Painting→**			**Sketches→**			
Target	C	P	S	R	P	S	R	C	S	R	C	P	Avg.
Resnet-50	54.60	57.92	43.71	50.87	38.37	43.92	66.65	50.33	39.87	48.28	52.46	43.47	49.20
MCD [10]	52.94	57.29	40.38	55.71	43.69	47.57	67.80	51.88	44.95	56.83	56.32	50.83	52.18
RSDA [3]	54.60	61.54	50.94	56.56	45.50	48.63	60.41	45.74	48.64	58.62	56.09	54.00	53.44
DANN [2]	61.67	60.27	53.86	58.23	46.46	51.63	64.17	52.70	52.88	61.55	62.73	56.70	56.90
BSP [1]	55.16	60.80	48.60	58.73	45.66	<u>55.47</u>	65.18	48.59	48.58	61.40	56.78	55.79	55.06
SAFN [14]	55.81	64.82	48.50	58.68	49.96	52.42	**73.71**	56.25	53.54	64.32	60.65	59.53	58.18
CDAN [7]	<u>70.41</u>	<u>66.87</u>	<u>57.73</u>	61.61	50.90	54.72	68.47	<u>59.43</u>	<u>55.49</u>	64.27	64.22	59.14	61.11
PAN [12][†]	67.56	66.73	55.86	<u>65.16</u>	**58.87**	54.55	70.46	57.54	53.14	<u>66.55</u>	<u>64.40</u>	<u>60.22</u>	<u>61.75</u>
MemSAC	**73.23**$^{\pm 0.09}$	**70.46**$^{\pm 0.13}$	**61.51**$^{\pm 0.08}$	**66.51**$^{\pm 0.21}$	<u>53.61</u>$^{\pm 0.39}$	**58.79**$^{\pm 0.68}$	<u>71.23</u>$^{\pm 0.20}$	**63.17**$^{\pm 0.75}$	**58.11**$^{\pm 0.63}$	**67.60**$^{\pm 0.16}$	**68.77**$^{\pm 0.52}$	**64.09**$^{\pm 0.51}$	**64.76**

Table 1: Accuracy scores on 126 classes on DomainNet. **Bold** and <u>underline</u> indicate the best and next best methods respectively. [†]Uses hierarchical label annotation.

2 Using a different adaptation backbone

The benefits obtained by MemSAC are complementary to the nature of adaptation method used. In Tab. 2a on DomainNet dataset with 345 classes, we show gains starting from a DANN [2] and CAN [5] objective as well, besides gains using the CDAN objective showcased in the main paper using a Resnet-50 backbone. These results indicate that the benefits using our objective are available to a wide variety of alignment methods.

3 MemSAC with deeper ResNets

In Tab. 2b, we show the results of the baselines as well as MemSAC with a deeper Resnet-101 backbone. **Due to memory constraints, we use a batch size**

Source	Real→			Clipart→			Painting→			Sketches→			
Target	C	P	S	R	P	S	R	C	S	R	C	P	Avg.
DANN [2]	45.93	44.51	35.47	46.85	30.52	36.77	48.02	34.76	32.15	47.1	46.45	38.47	40.58
DANN + MemSAC	49.67	48.61	39.14	49.81	35.1	40.59	50.04	38.51	36.61	50.31	50.8	42.73	**44.32**
CAN [5]	40.71	37.77	33.7	54.93	31.41	37.37	51.05	33.64	30.95	52.13	42.19	32.04	39.82
CAN + MemSAC	43.79	38.99	36.71	55.36	32.41	39.46	52.48	35.21	32.89	54.15	44.60	33.02	**41.59**

(a) Accuracy values of MemSAC using DANN and CAN adaptation backbones on DomainNet-345 classes. Note improved accuracy using MemSAC on top of both the backbones.

Source	Real→			Clipart→			Painting→			Sketches→			
Target	C	P	S	R	P	S	R	C	S	R	C	P	Avg.
Resnet-101	45.62	44.24	33.12	41.96	27.07	33.07	48.54	34.92	29.84	35.87	42.64	28.01	37.07
DANN [2]	47.71	44.1	35.99	48.33	32.00	38.54	48.13	34.57	34.23	48.19	48.56	39.67	41.67
MCD [10]	41.11	39.01	26.1	40.77	28.26	33.02	45.49	33.03	29.1	38.29	42.3	29.51	35.49
CDAN [7]	52.47	48.0	40.42	46.63	32.42	39.18	48.81	37.92	35.39	45.69	48.92	37.31	42.76
SAFN [14]	44.93	46.52	28.2	37.2	31.11	36.3	53.32	36.95	32.48	44.12	53.46	40.05	40.38
ToAlign [13]	50.10	48.27	35.98	50.24	31.41	41.10	54.60	43.67	36.82	50.15	54.32	42.06	44.89
MemSAC	**56.25**	**52.96**	**42.22**	**53.52**	**37.46**	**43.46**	**53.38**	**42.69**	**39.65**	**53.17**	**55.29**	**44.29**	**47.86**

(b) Results on DomainNet-345 dataset with Resnet-101 backbone and batch size of 24.

Table 2: **Ablations on DomainNet-345 dataset**.

of 24 for all the methods (unlike the experiments with Resnet-50 in the main paper where we use a batch size of 32). It is clearly evident that relative improvements by MemSAC over other baselines are still significant, indicating that our benefits persist even with a more powerful CNN backbone.

4 Ablations on KNN-based pseudo-labeling

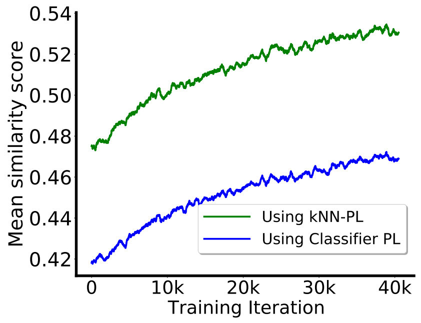


Fig. 1: Similarity Score

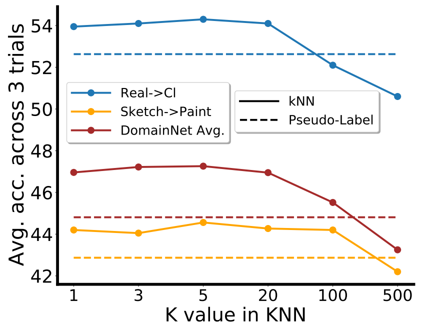


Fig. 2: Effect of K

A crucial choice made in the design of MemSAC is the use of kNN-based pseudo-labeling instead of directly using the classifier predictions on unlabeled target samples as pseudo-labels for all the target samples. This follows from the observation that the kNN based pseudo-labeling is generally robust to noisy classifier boundaries, especially amidst domain shifts. Moreover, with the help of the memory bank, the neighborhood from which the nearest neighbors are computed is much larger than the size of the mini-batch. We verify this intuition in Fig. 1, where the mean similarity score between the samples from the same class is much higher when trained using the proposed kNN based pseudo-labeling technique as compared to the classifier based pseudo-labeling technique. Furthermore, we analyze the effect of the choice of the parameter K in Fig. 2. Our accuracy is robust to most values of K in the range of 1-20. At large values of K, however, the accuracy falls steeply due to large amounts of noise in the pseudo-labels. Additionally, in Tab. 4b, we show that both the memory bank and kNN based pseudo-labeling are crucial to achieve performance gains using the consistency loss, as removing one of them (or both of them) results in significant drop in performance.

5 Results on Office-Home dataset

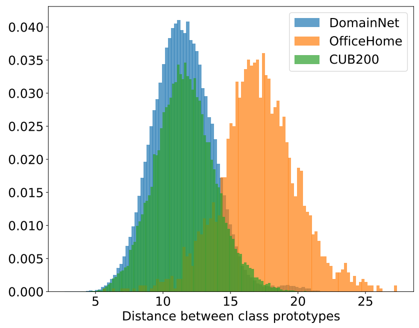
In the main paper, we show results using largest available datasets for domain adaptation, namely DomainNet-345 as well as CUB-200 with 345 and 200 categories, respectively. In Tab. 3, we show result using a medium-sized dataset, Office-Home [11]. Office-home contains 65 categories across 4 domains, and around 4k images in each domain. We observe that MemSAC outperforms competitive baselines even on Office-Home. Specifically, we use CDAN as the adaptation backbone, and report an improvement of 1.12% over this baseline, indicating the effectiveness of MemSAC even for unsupervised adaptation even on medium-sized datasets.

However, the results of MemSAC on OfficeHome dataset are not SOTA, which might be attributed to two reasons. Firstly, the categories in OfficeHome dataset are clearly distinct from each other leading to little avenues for negative alignment, which MemSAC tries to alleviate. To illustrate this, we use a ImageNet pre-trained Resnet-50 model and compute feature embeddings for all images from DomainNet-clipart domain. We then use the ground truth labels to find the class prototypes (or per-class average feature) for all the 345 classes and compute pairwise Euclidean distance between class prototypes. Lower euclidean distances between class prototypes indicates more class confusion and more likely negative transfer. We compute inter-class distances for OfficeHome and CUB200 datasets as well in similar fashion and show results in Fig. 3a. Evidently, the inter-class distances between classes from DomainNet and CUB200 are much smaller compared to OfficeHome, and hence greater benefit in using an approach like MemSAC.

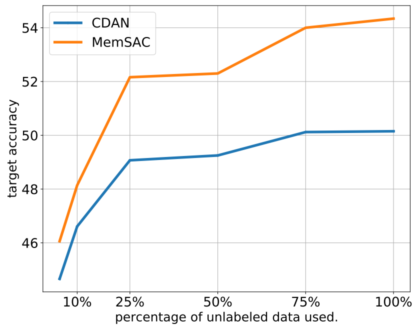
A second reason could be that the amount of unlabeled data in OfficeHome is much lesser compared to DomainNet on any transfer setting. On average, around 30k unlabeled samples are available in any target domains from DomainNet while only 4k samples are available from OfficeHome. To verify this argument, we

Table 3: Accuracy scores on 65 categories on OfficeHome [11] dataset.

Method	**A→C**	**A→P**	**A→R**	**C→A**	**C→P**	**C→R**	**P→A**	**P→C**	**P→R**	**R→A**	**R→C**	**R→P**	AVG
Resnet-50	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DANN [2]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN [8]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
CDAN [7]	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
BSP [1]	52.0	68.6	76.1	58.0	70.3	70.2	58.6	50.2	77.6	72.2	59.3	81.9	66.3
SAFN [14]	52.0	71.7	76.3	64.2	69.9	71.9	63.7	51.4	77.1	70.9	57.1	81.5	67.3
RSDA [3]	53.2	77.7	81.3	66.4	74.0	76.5	67.9	53.0	82.0	75.8	57.8	85.4	**70.9**
MemSAC	53.10	73.7	77.8	62.9	71.22	72.32	61.22	51.93	79.22	75.0	59.39	83.35	68.42



(a) Histogram of distances between class prototypes from each dataset. Lesser distance indicates more class confusion and more difficulty.



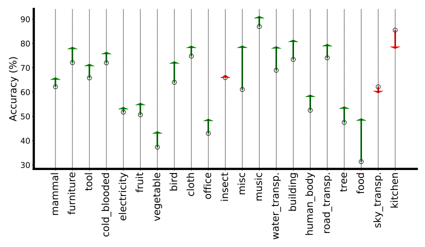
(b) Accuracy using varying degrees of unlabeled samples from DomainNet-345 dataset on **R** → **C**.

Fig. 3: (a) shows the distribution of distances between class prototypes from each category from DomainNet, CUB-200 and OfficeHome datasets. DomainNet and CUB-200 have lesser inter-class distances than Office-Home. (b) shows the effect of using varying degree of target unlabeled samples on the adaptation performance. The performance of MemSAC consistently improves as more unlabeled data becomes available.

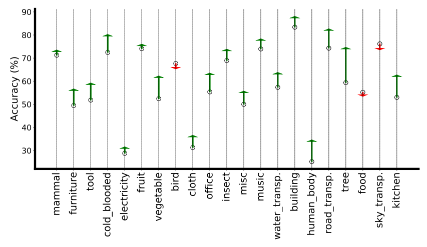
subsample the *clipart* domain from DomainNet to only use {5,10,25,50,75}% of unlabeled data during adaptation on the transfer task **R** → **C**. As indicated in Fig. 3b, the benefits from MemSAC grows significantly when larger unlabeled data is available. Note that OfficeHome contains only 10% of unlabeled data compared to DomainNet, and from Fig. 3b, the gains using MemSAC is minimal.

6 Category wise accuracies on DomainNet

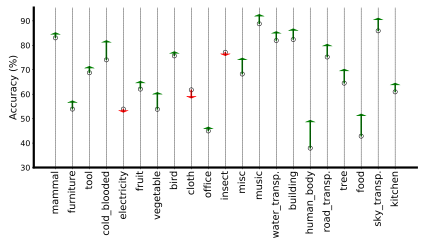
We show the accuracy for each *coarse* category and the gain/fall in accuracy between the baseline CDAN and MemSAC in Fig. 4 for few more tasks from DomainNet, in addition to the R→C task shown in the main paper. Evidently, MemSAC has non-trivial benefits over the baseline over most of the categories



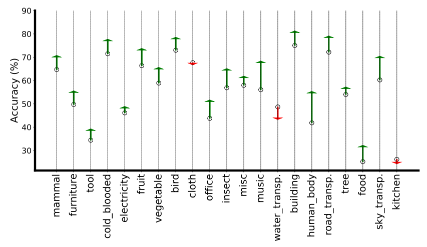
(a) **C** → **R**



(b) **P** → **C**



(c) **P** → **R**



(d) **S** → **P**

Fig. 4: Category wise accuracy increase and decrease on DomainNet dataset compared with CDAN baseline.

(marked by ↑), and any drops in accuracy (marked by ↓) are negligible. For example, on the task of P→C, we observe improvements of 14.6% on `trees` and 4.2% on category `insects`, thus indicating that our benefits sustain over most categories, and are more pronounced for categories containing finer grained classes.

7 Results on Birds-123 and CompCars datasets

In addition to the results on CUB200 in the main paper for fine-grained adaptation, we also show the results using MemSAC on other fine-grained datasets such as Birds-123 and CompCars [15]. Birds-123 contains images of different bird species from 123 common categories across CUB, NaBirds and iNaturalist datasets. CompCars, on the other hand, contains images from web and surveillance domains of 181 car models, and involves domain shift in the form of curated web images vs in-the-wild surveillance footage. Our method efficiently handles the domain shift across these challenging settings as shown in Tab. 4a. MemSAC attains an accuracy of 78.42% on the Birds-123 dataset and 52.75% on CompCars dataset which is much higher than all prior methods including PAN, even though PAN is specifically designed for fine-grained adaptation. These results verify the effectiveness of MemSAC on challenging fine-grained dataset settings.

	Birds-123	CompCars
Source	72.02	15.64
DANN	63.36	48.90
PAN	74.04	48.62
CDAN	72.95	50.40
MemSAC	78.42	52.75

(a)

	W/ kNN	Classifier PL
w/ Mem.	**47.26**	44.81
w/o Mem.	43.32	43.24

(b)

	DomainNet
Source Only	35.98
+ MemSAC	38.11(+2.13%)
CDAN	43.24
+ MemSAC	47.26(+4.02%)

(c)

Table 4: In a, we show the comparison of MemSAC with prior methods on fine-grained datasets Birds-123 and CompaCars. In b, we show the role of memory module and kNN pseudo labeling. In c, we show the role of adversarial losses to improve MemSAC training.

8 Role of adaptation in MemSAC

We next verify the role of adaptation losses in our proposed framework. While MemSAC can efficiently improve alignment using similarity consistency losses, we still need to bootstrap the training using adaptation losses for few iterations, to avoid noisy pseudo-labels in the later stages of training. As shown in Tab. 4c, while MemSAC can still boost performance of *Source Only* model, the gains observed using MemSAC alongside adaptation losses like CDAN are much higher.

9 Queue updates using momentum encoder

We now discuss possible alternative strategies to update the memory bank. For this purpose, we generalize the update rule using a *momentum encoder*, proposed in [4]. After the initial bootstrapping phase where we train the encoder on source data for few iterations, we initialize the momentum encoder $\mathcal{F}$ using the state of the encoder $\mathcal{E}$. After that, at every iteration, the parameters of the momentum encoder $\theta_{\mathcal{F}}$ are updated as follows.

$$\theta_{\mathcal{F}} = (1-\mu) * \theta_{\mathcal{E}} + (\mu) * \theta_{\mathcal{F}} \tag{1}$$

Here, μ is called the momentum parameter, and controls the speed of updates. The source features encoded in the memory bank $\mathcal{M}$ are obtained by a forward pass on $\mathcal{F}$, while the source features used to compute the supervised loss as well as all the target features are computed using a forward pass on $\mathcal{E}$. We note that the original update rule discussed in the main paper is just a special case of Eq. (1), which is obtained by putting $\mu = 0$.

The intuition behind using such a momentum based encoder is that it gives features with a slow drift through the training, and hence can support larger queues. We use such a momentum update on MemSAC and show results for CUB-Drawings dataset in Tab. 5a We found no benefit using such a momentum encoder in our method. This might be because we already bootstrap the encoder until the features stabilize and achieve a slow-drift phenomenon, and using momentum based updates on top of that might not improve accuracy. In light of these results, designing better memory bank update schedules is left as a potential direction for future work.

μ	C→D	D→C	Avg. Acc.
0	**73.97**	**61.94**	**67.95**
0.5	68.61	55.24	61.92
0.9	68.89	55.24	62.06
0.999	71.43	58.81	65.12

(a) MemSAC with different values of momentum parameter μ.

λ_{sc}	C→D	D→C	Avg. Acc.
0.001	65.84	51.29	58.56
0.01	69.38	55.91	62.64
0.1	**73.97**	**61.94**	**67.95**
1	19.02	50.56	36.29

(b) Effect of loss coefficient λ_{sc} on the accuracy for CUB-Drawings dataset on 200 classes.

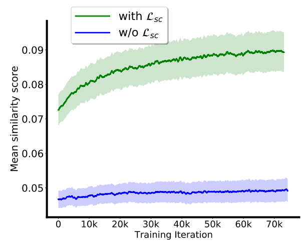
Table 5: **Ablation on CUB-Drawing dataset** using Resnet-50 backbone

10 Effect of loss coefficient

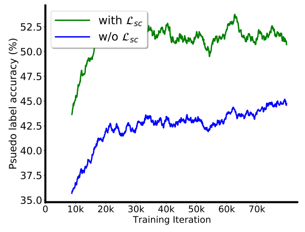
We show the ablation using the loss coefficient of our sample consistency loss in Tab. 5b on CUB-Drawing dataset. We find that using a value of λ_{sc} as 0.1 gave the best result, while using any larger value gives much inferior results, as noisy negative and positive pairs have a high influence on the training.

11 Training Curves

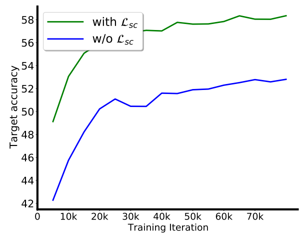
In Fig. 5, we show the trends for the mean similarity score, psuedo label accuracy as well as the final target accuracy during training. We compare between MemSAC which uses a consistency based loss, with an approach which does not contain such a consistency constraint. We observe that using our sample consistency loss gives a higher value of mean similarity score, psuedo-label accuracy as well as final target accuracy during training, and each of them improve with training indicating the effectiveness of our proposed loss.



(a)



(b)



(c)

Fig. 5: **Training Curves for $D \to C$** (a) Mean similarity score of within class samples vs. Training iterations. (b) Pseudo-label accuracy vs. Training iterations. (c) Final target accuracy vs. Training iterations

12 Training details

In Tab. 6, we give complete details regarding all the hyperparameters used for the experiments. While all the hyperparameters are same across both DomainNet and CUB-Drawings, we use a memory bank $\mathcal{M}$ of size 24k for CUB-Drawings and 48k for DomainNet. This is because datasets with larger number of images can give benefit with larger memory banks.

All the models were implemented using PyTorch 1.4.0 using 2080Ti GPUs. Following [7], learning rate is 0.003 for the feature encoder which is pretrained on ImageNet and 0.03 for the classifier.

Hyperparameter	Value
BatchSize	32
QueueSize	48000
$(\lambda_{adv}, \lambda_{sc})$	(1,0.1)
Temperature τ	0.07
Bootstrap Iter.	4000
Total Iterations	90k
k in kNN	5
Learning Rate for $\mathcal{E}$	0.003
Learning rate for $\mathcal{G}$ and $\mathcal{C}$	0.03
No. of GPUs	1

Table 6: Values of hyperparameters used in training MemSAC on all the experiments.

13 Limitations

Domain adaptation aims to efficiently address the problem of labeling overhead in low-resource domains enabling equitable performance of machine learning models across geographic, social or economic factors. However, MemSAC shares with other deep domain adaptation approaches the limitation of lack of explainability and uncalibrated model uncertainty, which may have a negative impact on applications where decisions based on domain adaptation have a bearing on safety or equity. Moreover, we also note the significant room for improvement to achieve accuracy levels of fully supervised models, as noted in Table 1 in the main paper (MemSAC vs. Tgt. Supervised).

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) 1, 4
2. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: International conference on machine learning. pp. 1180–1189. PMLR (2015) 1, 2, 4

3. Gu, X., Sun, J., Xu, Z.: Spherical space domain adaptation with robust pseudo-label loss. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9101–9110 (2020) 1, 4
4. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9729–9738 (2020) 6
5. Kang, G., Jiang, L., Yang, Y., Hauptmann, A.G.: Contrastive adaptation network for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4893–4902 (2019) 1, 2
6. Liang, J., Hu, D., Feng, J.: Combating domain shift with self-taught labeling. arXiv preprint arXiv:2007.04171 (2020) 1
7. Long, M., Cao, Z., Wang, J., Jordan, M.I.: Conditional adversarial domain adaptation. In: Advances in Neural Information Processing Systems. pp. 1640–1650 (2018) 1, 2, 4, 8
8. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Deep transfer learning with joint adaptation networks. In: International conference on machine learning. pp. 2208–2217. PMLR (2017) 4
9. Saito, K., Kim, D., Sclaroff, S., Darrell, T., Saenko, K.: Semi-supervised domain adaptation via minimax entropy. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8050–8058 (2019) 1
10. 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) 1, 2
11. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5018–5027 (2017) 3, 4
12. Wang, S., Chen, X., Wang, Y., Long, M., Wang, J.: Progressive adversarial networks for fine-grained domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9213–9222 (2020) 1
13. Wei, G., Lan, C., Zeng, W., Zhang, Z., Chen, Z.: Toalign: Task-oriented alignment for unsupervised domain adaptation. In: NeurIPS (2021) 2
14. 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) 1, 2, 4
15. Yang, L., Luo, P., Change Loy, C., Tang, X.: A large-scale car dataset for fine-grained categorization and verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3973–3981 (2015) 5
16. Yang, L., Wang, Y., Gao, M., Shrivastava, A., Weinberger, K.Q., Chao, W.L., Lim, S.N.: Mico: Mixup co-training for semi-supervised domain adaptation. arXiv preprint arXiv:2007.12684 (2020) 1