Domain Consensus Clustering for Universal Domain Adaptation
Supplementary Materials

Guangrui Li[1], Guoliang Kang[2], Yi Zhu[3], Yunchao Wei[1], Yi Yang[1]
[1] ReLER Lab, AAII, Univeristy of Technology Sydney
[2] Carnegie Mellon University, [3] Amazon Web Services

In appendix A, we introduce the formulation of contrastive domain discrepancy. Then, we complement more ablation analysis and results in appendix B and appendix C, respectively. In appendix D, we elaborate the implementation details. Finally, in appendix E, we show the feature visualization with t-SNE [12].

A. Contrastive Domain Discrepancy

In this section, we detail the formulation of Contrastive Domain Discrepancy (CDD) [7]. Here, given input $\boldsymbol{x}_i$, we define the output of l th layer as $\phi_l(\boldsymbol{x}_i)$, where the model is parameterized by ϕ.

Formally, Maximum Mean Discrepancy(MMD) [5] defines the difference between two distributions with their mean embeddings in the reproducing kernel Hilbert space (RKHS), i.e., $\mathcal{D}_{\mathcal{H}}(P,Q) \triangleq \sup_{f\sim\mathcal{H}} \left(\mathbb{E}_{\boldsymbol{X}^s}[f(\boldsymbol{X}^s)] - \mathbb{E}_{\boldsymbol{X}^t}[f(\boldsymbol{X}^t)]\right)_{\mathcal{H}}$, where H is class of functions. Then, for a layer l, the squared value of MMD is estimated with the empirical kernel mean embeddings:

$$\hat{\mathcal{D}}_l^{mmd} = \frac{1}{n_s^2}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s} k_l(\phi_l(\boldsymbol{x}_i^s),\phi_l(\boldsymbol{x}_j^s)) + \frac{1}{n_t^2}\sum_{i=1}^{n_t}\sum_{j=1}^{n_t} k_l(\phi_l(\boldsymbol{x}_i^t),\phi_l(\boldsymbol{x}_j^t)) - \frac{2}{n_s n_t}\sum_{i=1}^{n_s}\sum_{j=1}^{n_t} k_l(\phi_l(\boldsymbol{x}_i^s),\phi_l(\boldsymbol{x}_j^t)), \tag{1}$$

where $\boldsymbol{x}^s \in \mathcal{S}' \subset \mathcal{S}$, $\boldsymbol{x}^t \in \mathcal{T}' \subset \mathcal{T}$, $n_s = |\mathcal{S}'|$, $n_t = |\mathcal{T}'|$. The $\mathcal{S}'$ and $\mathcal{T}'$ represent the mini-batch source and target data sampled from $\mathcal{S}$ (*i.e.*, set of source data) and $\mathcal{T}$ (*i.e.*, set of target data) respectively. And k_l denotes the kernel selected for the l-th layer of deep neural network.

Built upon Maximum Mean Discrepancy, Contrastive Domain Discrepancy(CDD) takes the intra- and inter- class discrepancy into the consideration simultaneously.

Concretely, supposing $\rho_{cc'}(y,y') = \begin{cases} 1 & \text{if } y = c, y' = c'; \\ 0 & \text{otherwise.} \end{cases}$, for two classes c_1, c_2 (which can be same or different), the kernel mean embedding estimation for squared $\mathcal{D}_{\mathcal{H}}(P,Q)$ is

$$\hat{\mathcal{D}}^{c_1c_2}(\hat{y}_1^t,\hat{y}_2^t,\cdots,\hat{y}_{n_t}^t,\phi) = e_1 + e_2 - 2e_3, \tag{2}$$

where

$$\begin{aligned} e_1 &= \sum_{i=1}^{n_s}\sum_{j=1}^{n_s} \frac{\rho_{c_1c_1}(y_i^s,y_j^s)k(\phi(\boldsymbol{x}_i^s),\phi(\boldsymbol{x}_j^s))}{\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}\rho_{c_1c_1}(y_i^s,y_j^s)} \\ e_2 &= \sum_{i=1}^{n_t}\sum_{j=1}^{n_t} \frac{\rho_{c_2c_2}(\hat{y}_i^t,\hat{y}_j^t)k(\phi(\boldsymbol{x}_i^t),\phi(\boldsymbol{x}_j^t))}{\sum_{i=1}^{n_t}\sum_{j=1}^{n_t}\rho_{c_2c_2}(\hat{y}_i^t,\hat{y}_j^t)} \\ e_3 &= \sum_{i=1}^{n_s}\sum_{j=1}^{n_t} \frac{\rho_{c_1c_2}(y_i^s,\hat{y}_j^t)k(\phi(\boldsymbol{x}_i^s),\phi(\boldsymbol{x}_j^t))}{\sum_{i=1}^{n_s}\sum_{j=1}^{n_t}\rho_{c_1c_2}(y_i^s,\hat{y}_j^t)}. \end{aligned} \tag{3}$$

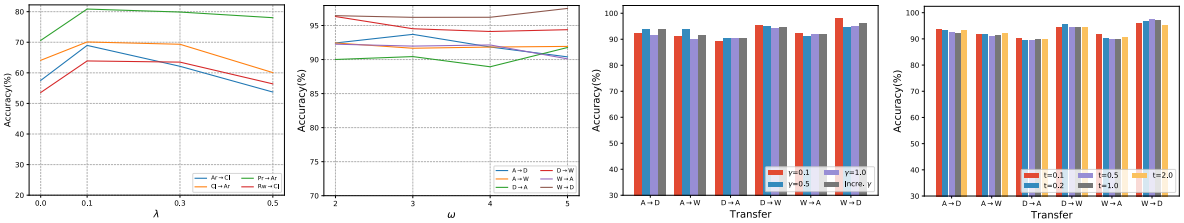
Notably, Eq. 2 delivers two kinds of class-aware domain discrepancy, 1) when $c_1 = c_2$, it measures *intra-class* domain discrepancy; 2) when $c_1 \neq c_2$, it becomes the *inter-class* domain discrepancy.

Based on the above definitions, the CDD is calculated as (The $\hat{y}^t_1, \hat{y}^t_2, \cdots, \hat{y}^t_{n_t}$ is abbreviated as $\hat{y}^t_{1:n_t}$)

$$\mathcal{L}_{cdd} = \underbrace{\frac{1}{|C^s|}\sum_{c=1}^{|C^s|}\hat{\mathcal{D}}^{cc}(\hat{y}^t_{1:n_t}, \phi)}_{intra} - \underbrace{\frac{1}{|C^s|(|C^s|-1)}\sum_{c=1}^{|C^s|}\sum_{\substack{c'=1 \\ c'\neq c}}^{|C^s|}\hat{\mathcal{D}}^{cc'}(\hat{y}^t_{1:n_t}, \phi)}_{inter}, \tag{4}$$

where the intra- and inter-class domain discrepancies will be optimized in the opposite direction.

B. More Ablation Analysis



(a) (b) (c) (d)

Figure A. (a) Sensitivity analysis to λ on OfficeHome. (b) Sensitivity analysis to ω (Office-31). (c) Comparison between constant weight and incremental weight of γ (Office-31). (d) Sensitivity analysis to τ (Office-31). All experiments are conducted under the UniDA setting.

Sensitivity to hyper-parameters. In the main paper, we have discussed the sensitivity to λ and γ in Sec. 5.3. Here, we complement more experimental results and further analysis. All these experiments are conduct under the UniDA setting.

In Fig A (a), we present the sensitivity analysis to λ on OfficeHome, which further justifies the robustness of the proposed approach. Then in Fig A (b), we conducts experiments on Office-31 to testify the robustness to ω, which controls the increase rate of the weight. We could observe that the performance only fluctuates in a narrow range under the change of ω, which proves the the insensitivity to ω. Finally, we present the comparison between constant weight and incremental weight of γ on Office-31 in Fig. A (c). Obviously, incremental weight of γ does not shows apparent improvement than the constant ones, which is inconsistent with the phenomenon on OfficeHome (Sec. 5.3). This is because the samples of Office-31 converge to the optimal clustering more quickly so that the boundary between clusters are distinct enough, which results in the insensitivity to the setting of γ. Moreover, we also testify the robustness to τ (temperature on the regularizer) with a wide range, and present the comparisons on Fig. A (d). As shown in the figure, the regularizer is insensitive to τ as the design of incremental weight allows the regularizer to work in a progressive way.

Time Complexity of Clustering. The K-means implementation we adopt is based on scikit-learn, which employs Elkan algorithm. For n samples, k clusters, and e iterations, the time complexity is roughly $\mathcal{O}(nke)$. For one round of clusterings, we only need a very few times of searches, *e.g.*, 3 for Office31 and OfficeHome. Moreover, after a few rounds of training, we empirically observe the optimal K will converge to a certain value. And we will stop searching for K in the subsequent training to improve the efficiency, as explained in Sec. 4.3.

C. Supplemental Results

Table A. Results (%) on **Office-31** under Closed Set Scenario. All results except ours are cited from [14].

	A→W	A→D	D→W	W→D	D→A	W→A	Avg
DANN [4]	86.7	97.2	99.8	86.1	72.5	72.8	85.9
ETN [2]	87.9	**99.2**	**100**	88.4	68.7	66.8	85.2
STA [10]	77.1	90.7	98.1	75.5	51.4	48.9	73.6
UAN [16]	86.5	97.0	**100**	84.5	69.6	68.7	84.4
DANCE [14]	88.6	89.4	97.5	100	69.5	68.2	85.5
Ours	**89.09**	87.24	96.81	100	**74.39**	**76.77**	**87.38**

Table B. Results (%) on **Office-31** for OSDA (ResNet-50).

OSDA	A→W		A→D		D→W		W→D		D→A		W→A		Avg	
	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*	OS	OS*
RTN [11]	85.6±1.2	88.1±1.0	89.5±1.4	90.1±1.6	94.8±0.3	96.2±0.7	97.1±0.2	98.7±0.9	72.3±0.9	72.8±1.5	73.5±0.6	73.9±1.4	85.4	86.8
DANN [4]	85.3±0.7	87.7±1.1	86.5±0.6	87.7±0.6	97.5±0.2	98.3±0.5	99.5±0.1	**100.0**±.0	75.7±1.6	76.2±0.9	74.9±1.2	75.6±0.8	86.6	87.6
ATI-λ [13]	87.4±1.5	88.9±1.4	84.3±1.2	86.6±1.1	93.6±1.0	95.3±1.0	96.5±0.9	98.7±0.8	78.0±1.8	79.6±1.5	80.4±1.4	81.4±1.2	86.7	88.4
OSBP [15]	86.5±2.0	87.6±2.1	88.6±1.4	89.2±1.3	97.0±1.0	96.5±0.4	97.9±0.9	98.7±0.6	88.9±2.5	90.6±2.3	85.8±2.5	84.9±1.3	90.8	91.3
STA [10]	89.5±0.6	92.1±0.5	93.7±1.5	96.1± 0.4	**97.5**±0.2	96.5±0.5	99.5±0.2	99.6±0.1	89.1±0.5	93.5±0.8	87.9±0.9	87.4±0.6	92.9	94.1
Inheritune [8]	91.3±0.7	93.2±1.2	**94.2**±1.1	**97.1**±0.8	96.5±0.5	97.4±0.7	**99.5**±0.2	99.4±0.3	90.1±0.2	91.5± 0.2	88.7±1.3	88.1±0.9	93.4	94.5
Ours	**93.8**±1.0	**99.4**±1.1	90.72±1.1	95.6±0.9	96.9±0.5	**98.4**±0.7	95.7±0.2	98.4±0.1	**92.5**±0.5	**96.6**± 0.4	**94.5**±2.1	**96.3**±1.8	**94.0**	**97.5**

Table C. Results (%) on **Office-Home** for OSDA (ResNet-50).

OSDA	Ar→Cl	Pr→Cl	Rw→Cl	Ar→Pr	Cl→Pr	Rw→Pr	Cl→Ar	Pr→Ar	Rw→Ar	Ar→Rw	Cl→Rw	Pr→Rw	Avg
ATI-λ [13]	55.2	52.6	53.5	69.1	63.5	74.1	61.7	64.5	70.7	79.2	72.9	75.8	66.1
DANN [4]	54.6	49.7	51.9	69.5	63.5	72.9	61.9	63.3	71.3	80.2	71.7	74.2	65.4
OSBP [15]	56.7	51.5	49.2	67.5	65.5	74.0	62.5	64.8	69.3	80.6	74.7	71.5	65.7
STA [10]	58.1	53.1	54.4	71.6	69.3	81.9	63.4	65.2	74.9	85.0	75.8	80.8	69.5
Inheritune [8]	60.1	**54.2**	56.2	70.9	70.0	78.6	64.0	**66.1**	**74.9**	83.2	75.7	81.3	69.6
Ours	**60.89**	52.68	**60.25**	**82.94**	**70.20**	**83.28**	**64.54**	65.74	73.26	**86.11**	**77.99**	**83.53**	**71.78**

Results under Open Set Setting. In Table B and Table C, we provide more results Office-31 and OfficeHome under OSDA setting. We presents the **OS** and **OS*** results in Office-31 and **OS** for OfficeHome, where **OS** is the class-wise mean accuracy on common classes and the *unknown* class and **OS*** is the average over common classes. Notably, these results follow the universal principle, *i.e.*, both domains may hold private classes.

Results under Closed Set Setting. In Table C, we present the classification results under the closed set domain adaptation setting (CSDA). Notably, all above results, except ours, are drawn from DANCE [14]. Inferred from this table, we could observe that our method maintains its superiority against other approaches, and achieves better results on Office, which again verify the stronger generalizability of the proposed method.

Table D. Average Accuracy (%) on **Office-Home** (ResNet-50)

Method	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg
ResNet [6]	59.37	76.58	87.48	68.86	71.11	81.66	73.72	56.30	86.07	78.68	59.22	78.59	73.22
DANN [4]	56.17	81.72	85.87	68.67	73.38	83.76	69.92	56.84	85.80	79.41	57.26	78.26	73.17
RTN [11]	50.46	77.80	86.90	65.12	73.40	85.07	67.86	45.23	85.50	79.20	55.55	78.79	70.91
IWAN [17]	52.55	81.40	86.51	70.58	70.99	85.29	74.88	57.33	85.07	77.48	59.65	79.91	73.39
PADA [1]	39.59	69.37	76.26	62.57	67.39	77.47	48.39	35.79	79.60	75.94	44.50	78.10	62.91
ATI [13]	52.90	80.37	85.91	71.08	72.41	84.39	74.28	57.84	85.61	76.06	60.17	78.42	73.29
OSBP [15]	47.75	60.90	76.78	59.23	61.58	74.33	61.67	44.50	79.31	70.59	54.95	75.18	63.90
UAN [16]	63.00	82.83	87.85	76.88	78.70	85.36	78.22	58.59	86.80	83.37	63.17	79.43	77.02
USFDA [9]	63.35	83.30	89.35	70.96	72.34	86.09	78.53	**60.15**	87.35	81.56	63.17	**88.23**	77.03
CMU [3]	**63.52**	**83.81**	88.94	**77.72**	**79.37**	86.85	78.61	59.27	88.25	**84.06**	**64.57**	81.36	**78.03**
Ours	63.10	80.95	**92.12**	69.27	75.82	**87.11**	**81.46**	55.78	**92.10**	82.42	62.05	87.29	77.46

Results under Universal Setting. In Table D, we show the averaged accuracy on OfficeHome. Our proposed DCC achieves competitive results compared to previous literature. Note that CMU [3] reports higher average accuracy, but it employs classifier ensemble. Despite of this, our method still attains better results in terms of multiple sub-tasks.

D. Implementation Details

Update strategy of the prototype bank. Through K-means clustering, we could obtain the initial prototypes of target samples, $\{\mu^t_{1(0)}, ..., \mu^t_{K(0)}\}$. Here, we detail the update of prototypes during training.

In each iteration, we calculate the local prototypes in a batch as:

$$\bar{\mu}^t_{k(I)} = \frac{1}{|\bar{\mathcal{D}}^t_{k(I)}|} \sum_{\boldsymbol{x}^t_i \in \bar{\mathcal{D}}^t_{k(I)}} \frac{f_\phi(\boldsymbol{x}^t_i)}{||f_\phi(\boldsymbol{x}^t_i)||}, \tag{5}$$

where I denotes the current iteration and $\bar{\mathcal{D}}^t_{k(I)}$ represents target samples with cluster label k in iteration I.

The global prototype is updated as :

$$\mu^t_{k(I)} = \sigma_I \mu^t_{k(I-1)} + (1 - \sigma_I)\bar{\mu}^t_{k(I)}, \tag{6}$$

where σ is the similarity between local prototypes and global prototypes, it is calculated as:

$$\sigma_I = Sim(\mu^t_{k(I-1)}, \bar{\mu}^t_{k(I)}) \tag{7}$$

where $Sim(\cdot)$ denotes the cosine similarity, *i.e.*, $Sim(\boldsymbol{a}, \boldsymbol{b}) = \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$.

In this way, the prototype bank updates adaptively according to the similarity between global prototypes and local prototypes, which enables the prototype bank update more efficiently.

Training strategy. For Office-31 and OfficeHome, the model is trained for 5K steps and the clustering is performed every 200 steps. Considering the larger scale of VisDA and DomainNet, the total step is set to 10K, and the cluster interval is set to 500 and 1K, respectively. Besides, to handle the larger distribution shift in VisDA and DomainNet, we set the initial 1K steps as warm-up stage where only the cross-entropy loss are applied on source samples. We use random cropping and random horizontal flips for data augmentation in training.

H-score. Here, we present the formulation of H-score [3]:

$$h = 2 \cdot \frac{a_{\mathcal{C}} \cdot a_{\bar{\mathcal{C}}^t}}{a_{\mathcal{C}} + a_{\bar{\mathcal{C}}^t}}, \tag{8}$$

where $a_{\mathcal{C}}$ and $a_{\bar{\mathcal{C}}^t}$ denote the instance accuracy on common class and private class, respectively. This evaluation metric is high only both $a_{\mathcal{C}}$ and $a_{\bar{\mathcal{C}}^t}$ are high, so that emphasizes the recognition of both common samples and private samples.

E. Visualization

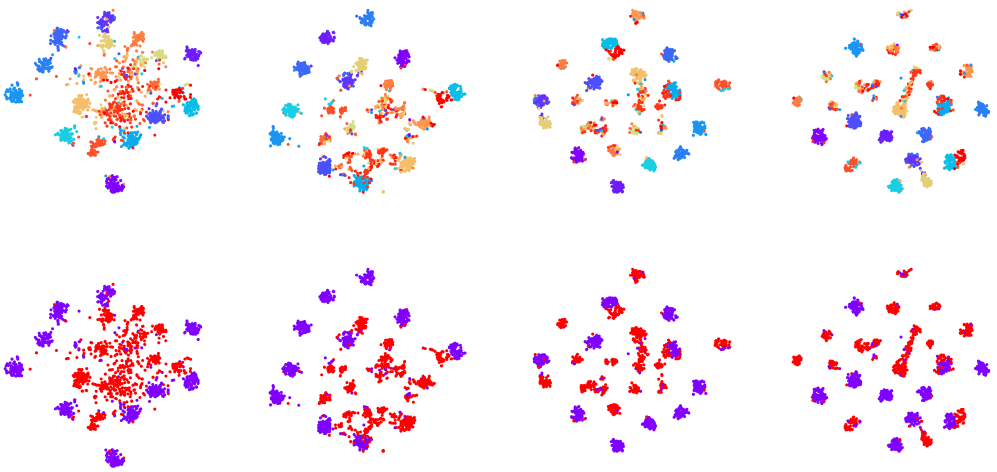


Figure B. t-SNE [12] plot of target samples at different training stages in **D** $\rightarrow$ **A** of Office-31 (Best viewed in color). In the first row, different colors represent different classes; in the second row, purple dots denote common samples while red ones denotes private samples.

In Fig. B, we present the target feature visualization in the transfer **D**$\rightarrow$**A** of Office-31. As shown in these figures, we could observe that both common samples and private samples are aggregated into discriminative clusters, which is consistent with the motivation of DCC. Moreover, the inter-cluster discrepancies are gradually enlarged as training progress, which verifies the effectiveness of the prototypical regularizer.

References

[1] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *ECCV*, 2018. 3

[2] Zhangjie Cao, Kaichao You, Mingsheng Long, Jianmin Wang, and Qiang Yang. Learning to transfer examples for partial domain adaptation. In *CVPR*, 2019. 2

[3] Bo Fu, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Learning to detect open classes for universal domain adaptation. In *ECCV*, 2020. 3, 4

[4] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 2016. 2, 3

[5] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *NeurIPS*, 2007. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[7] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *CVPR*, 2019. 1

[8] Jogendra Nath Kundu, Naveen Venkat, Rahul M V, and R. Venkatesh Babu. Universal source-free domain adaptation. In *CVPR*, 2020. 3

[9] Jogendra Nath Kundu, Naveen Venkat, Rahul M V, and R. Venkatesh Babu. Universal source-free domain adaptation. In *CVPR*, 2020. 3

[10] Hong Liu, Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Qiang Yang. Separate to adapt: Open set domain adaptation via progressive separation. In *CVPR*, 2019. 2, 3

[11] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, 2016. 3

[12] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008. 1, 4

[13] P. Panareda Busto, A. Iqbal, and J. Gall. Open set domain adaptation for image and action recognition. *IEEE TPAMI*, 2020. 3

[14] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, and Kate Saenko. Universal domain adaptation through self-supervision. In *NeurIPS*, 2020. 2, 3

[15] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *ECCV*, 2018. 3

[16] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Universal domain adaptation. In *CVPR*, 2019. 2, 3

[17] J. Zhang, Z. Ding, W. Li, and P. Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *CVPR*, 2018. 3