# Supplemental Material: Active Universal Domain Adaptation

Xinhong Ma[1,2], Junyu Gao[1,2] and Changsheng Xu[1,2,3]
[1] National Lab of Pattern Recognition (NLPR),
Institute of Automation, Chinese Academy of Sciences (CASIA)
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS)
[3] Peng Cheng Laboratory, Shenzhen, China
{xinhong.ma, junyu.gao, csxu}@nlpr.ia.ac.cn

## Contents



## 1. Further Illustrations About Our Approach

### 1.1. The Proposed Active Learning Strategy (CNTGE)

In CNTGE, we first run k-means to select transferable and non-transferable target instances. Then, transferable target instances are used to construct pseudo labeled dataset while non-transferable target instances are leveraged for active learning. Some details need further illustrations.

**How to determine a suitable cluster number for k-means?** For each active learning round, we can only query $n_r$ samples' labels from an oracle. Therefore, we directly set the cluster number of k-means as $n_r$ for a fair comparison with other active learning strategies, such as `K-means`, `Coreset` and `BADGE` which also need run k-means algorithm and their clustering numbers are set as $n_r$ too.

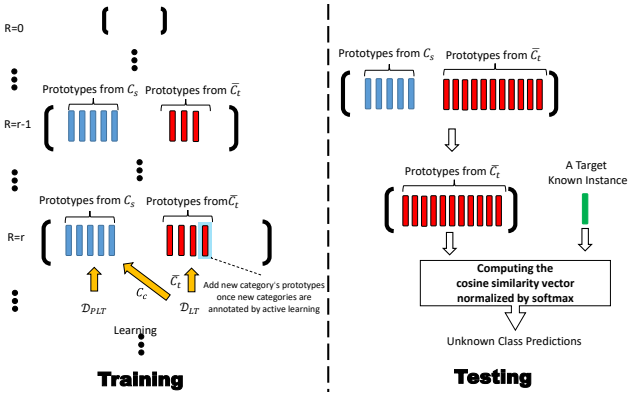



Figure 1. The training and testing of the prototype classifiers $G_p$.

**How to compute gradient embeddings for non-transferabel instances $\mathcal{D}_{NT}$ and select $n_r$ informative instances for active learning?** Details refer to [1]. Here, we simply summarize the whole process as the following steps:

- Computing the pseudo label vector of $\mathbf{x}_i$, *i.e.*, $\bar{\mathbf{y}}_i = G_c(G_f(\mathbf{x}_i))$.
- Computing the gradient embedding

$$g_{\mathbf{x}_i} = \frac{\partial L_c(\bar{\mathbf{y}}_i, G_c(G_f(\mathbf{x}_i); \theta_c))}{\partial \theta_{out}}|_{\theta_c=\theta_c^t}$$

  where $\theta_{out}$ refers to parameters of the final (output) layer of $G_c$ and $\theta_c^t$ denotes the classifier's parameters at time $t$.
- Running K-Means++ seeding algorithm on $\{g_{\mathbf{x}_i}|\mathbf{x}_i \in \mathcal{D}_{NT}\}$ where the cluster number is set as $n_r$.

**About K-Means.** For a fair comparison, K-means in our CNTGE strategy follows [1], which utilizes cosine distance to measure similarities without dimension reduction.

### 1.2. The Prototype Classifiers $G_p$

The prototype classifiers $G_p$ is designed to classify target "unknown" instances as target private classes, which maintains class representations (prototypes) in the target domain.

Table 1. Average class accuracy (%) on Office-Home, Office-31, VisDA and DomainNet for comparing different DA methods equipped with different AL strategies. The best results are bolded.

| | AL | Random | | | Margin | | | Coreset | | | BADGE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Task | Resnet | UAN | ADCL | Resnet | UAN | ADCL | Resnet | UAN | ADCL | Resnet | UAN | ADCL |
| **Office-Home** | Ar→Cl | 18.31 | 23.61 | **29.33** | 20.13 | 25.75 | **29.49** | 21.17 | 24.96 | **26.94** | 20.66 | 26.56 | **30.53** |
| | Ar→Pr | 32.34 | 38.09 | **60.05** | 31.32 | 36.38 | **58.49** | 37.50 | 38.04 | **56.77** | 34.80 | 39.62 | **58.44** |
| | Ar→Rw | 31.40 | 33.04 | **48.32** | 31.55 | 30.93 | **51.36** | 35.36 | 30.90 | **48.89** | 29.33 | 32.92 | **49.79** |
| | Cl→Ar | 26.27 | 27.73 | **47.27** | 30.67 | 29.84 | **44.79** | 29.99 | 29.29 | **44.29** | 28.46 | 29.00 | **44.59** |
| | Cl→Pr | 26.27 | 32.85 | **56.53** | 32.85 | 32.85 | **56.35** | 36.19 | 32.12 | **52.94** | 40.31 | 32.81 | **58.63** |
| | Cl→Rw | 31.71 | 31.80 | **47.20** | 31.79 | 30.48 | **47.73** | 36.73 | 30.02 | **48.50** | 31.58 | 30.67 | **48.00** |
| | Pr→Ar | 25.70 | 35.25 | **52.78** | 26.02 | 35.22 | **50.11** | 29.03 | 34.95 | **51.47** | 28.46 | 35.57 | **54.68** |
| | Pr→Cl | 17.25 | 26.94 | **29.25** | 17.80 | 26.96 | **30.02** | 19.63 | 27.67 | **28.00** | 18.34 | 29.28 | **29.91** |
| | Pr→Rw | 31.03 | 37.41 | **58.68** | 32.20 | 37.14 | **60.89** | 34.60 | 37.97 | **59.10** | 29.54 | 38.95 | **61.01** |
| | Rw→Ar | 26.24 | 32.85 | **50.11** | 27.89 | 34.81 | **49.26** | 29.79 | 35.32 | **50.17** | 24.47 | 36.60 | **51.79** |
| | Rw→Cl | 17.25 | **27.54** | 25.01 | 19.56 | **28.33** | 22.73 | 19.59 | **29.45** | 22.68 | 18.58 | **30.34** | 25.64 |
| | Rw→Pr | 32.09 | 43.84 | **61.77** | 34.96 | 42.24 | **61.06** | 35.44 | 42.55 | **58.99** | 35.71 | 43.99 | **62.28** |
| | AVG | 26.32 | 32.58 | **47.19** | 28.06 | 32.58 | **46.86** | 30.42 | 32.77 | **45.73** | 28.35 | 33.86 | **47.94** |
| | Average Accuracy | ResNet: 28.29 | | UAN: 32.95 | | ADCL (Ours): **47.33** | | | | | | | |
| **Office-31** | A→D | 84.68 | 67.33 | **85.23** | 80.81 | 57.59 | **83.75** | 84.85 | 68.14 | **86.18** | 83.73 | 61.82 | **84.45** |
| | A→W | 83.72 | 65.01 | **84.22** | **85.95** | 62.48 | 82.11 | 84.77 | 65.71 | **86.10** | **86.77** | 64.82 | 86.16 |
| | D→A | 59.37 | 55.69 | **64.90** | 64.74 | 58.00 | **66.31** | 63.58 | 57.71 | **66.33** | 64.59 | 58.09 | **68.77** |
| | D→W | 84.00 | 77.31 | **91.35** | 85.12 | 71.02 | **91.27** | 86.45 | 74.36 | **92.76** | 85.78 | 70.88 | **91.34** |
| | W→A | 59.53 | 53.63 | **62.84** | 60.24 | 54.97 | **61.55** | 64.08 | 55.91 | **65.21** | 61.78 | 53.38 | **65.74** |
| | W→D | 82.76 | 68.39 | **86.33** | 81.35 | 59.28 | **86.32** | 84.07 | 70.74 | **88.17** | 81.11 | 62.74 | **86.59** |
| | AVG | 75.67 | 64.56 | **79.15** | 76.37 | 60.56 | **78.55** | 77.97 | 65.43 | **80.79** | 77.29 | 61.96 | **80.51** |
| | Average Accuracy | ResNet: 76.83 | | UAN: 63.13 | | ADCL (Ours): **79.75** | | | | | | | |
| | **VisDA** | 59.27 | 59.11 | **63.15** | 61.98 | **72.45** | 63.49 | 61.43 | 57.28 | **62.58** | 61.91 | 62.83 | **64.00** |
| | Average Accuracy | ResNet: 61.15 | | UAN: 62.92 | | ADCL (Ours): **63.31** | | | | | | | |
| **DomainNet** | P→R | 27.46 | 43.08 | **48.56** | 36.37 | 40.65 | **46.69** | 35.91 | **40.98** | 31.74 | 37.68 | **40.41** | 40.31 |
| | R→P | 27.46 | 35.00 | **36.24** | 27.75 | 33.58 | **36.57** | 27.73 | 33.74 | **34.57** | 27.33 | 36.27 | **37.31** |
| | P→S | 24.00 | 25.57 | **29.53** | 24.42 | 29.52 | **31.11** | 23.73 | **29.61** | 29.45 | 24.76 | 29.44 | **30.69** |
| | S→P | 25.92 | 27.00 | **33.98** | 26.08 | 30.92 | **34.05** | 26.75 | 31.81 | **33.77** | 26.88 | 32.35 | **35.70** |
| | R→S | 24.19 | **31.37** | 29.49 | 24.33 | **31.66** | 29.93 | 24.00 | **31.93** | 29.09 | 25.85 | **31.83** | 31.13 |
| | S→R | 35.56 | 42.70 | **47.46** | 35.49 | 43.17 | **45.90** | 35.35 | 44.74 | **47.42** | 41.57 | 45.10 | **47.38** |
| | AVG | 27.43 | 34.12 | **37.54** | 29.07 | 34.91 | **37.37** | 28.91 | **35.47** | 34.34 | 30.68 | 35.90 | **37.09** |
| | Average Accuracy | ResNet: 29.02 | | UAN: 35.10 | | ADCL (Ours): **36.59** | | | | | | | |

The training and testing mechanisms of $G_p$ is shown in Figure 1. Before training, $G_p$ contains no prototypes. During training, new prototypes will be dynamically added into $G_p$, once an instance with target private classes is annotated by active learning and its prototype is not stored in $G_p$. The prototypes of the shared label set are learned on $\mathcal{D}_{PLT}$ and $\mathcal{D}_{LT}$ while the prototypes of target private label set are learned on $\mathcal{D}_{LT}$. All prototypes in $G_p$ is optimized by the classification loss $L_p$. Once the $G_p$ is trained, only prototypes beyond the source label set are used for inferring target unknown instances.

## 2. Additional Experiments

### 2.1. comparing different DA methods equipped with different AL strategies

In the main paper, we have already compared different DA methods equipped with different AL strategies. Here, we provide the complete results (Table 1) on all transfer tasks in Office-Home, Office-31, VisDA and DomainNet datasets.

### 2.2. Comparing the Proposed Adversarial and Diverse Curriculum Leaning (ADCL) with Universal Domain Adaptation Methods

**Compared Methods.** To further testify whether the proposed adversarial and diverse curriculum leaning can effectively deal with the domain gap and semantic shift, we conduct extensive experiments with representative universal domain adaptation (UniDA) methods, including two adversarial alignments methods (UAN [4], CMU [2]) and a neighborhood clustering method (DANCE [3]). The ADCL model can be obtained by directly removing the active leaning and joint training stages in AUAN. Under universal domain adaptation setting, a robust UniDA model requires to classify target samples as any class in the source labels set or mark them as "unknown" without any prior knowledge on the target label set.

**Evaluation Protocols.** To evaluate the classification performance of UniDA models, we follow [2] to report H-score for comparisons. H-score is a harmonic mean of the instance accuracy on common classes $a_c$ and accuracy on the "unknown" class $a_{\bar{c}}$ as: $h = 2 \cdot \frac{a_c \cdot a_{\bar{c}}}{a_c + a_{\bar{c}}}$. The evaluation metric is high only when both the $a_c$ and $a_{\bar{c}}$ high, which emphasizes the importance of both abilities of UniDA methods.

**Comparison against state-of-the-art universal domain adaptation methods.** As shown in Table 2, ADCL achieves state-of-the-art results on all domain adaptation tasks of Office-Home dataset. The average improvement of H-scores is 8.62 %. It indicates that the proposed ADCL can effectively deal with domain gap and semantic shift problems, helping active learning to select more informative instances for annotations.

**Comparison between UniDA models with different metrics trained with ADCL.** In our ADCL algorithm, we utilize the transfer score $w_t(\mathbf{x}_i^t)$ to measure the transferability of a target sample $\mathbf{x}_i^t$. Other metrics in some adversarial alignment UniDA methods [4, 2] can also be used to replace the transfer score and perform adversarial and diverse curriculum learning. To prove the generalization of the proposed adversarial and diverse curriculum learning, we insert adversarial and diverse curriculums into UAN [4]. Specifically, based on the adversarial training loss in UAN [4], we modify the adversarial loss by inserting the indicator function $\mathbb{1}_{w_t'(\mathbf{x}_i^t) \geqslant w_\alpha(t)}$ to select target samples for adversarial alignment. Same as UAN [4], source domain are also scored, by the score $w_s'(\mathbf{x}_i^s) = -w_t'(\mathbf{x}_i^s)$. Meanwhile, the loss $L_{div}$ is applied to UAN. Note that we utilize the metric $w_t'$ in UAN, not the transfer score $w_t$, to measure samples' transferability. $w_t'$ is defined as Eq (1):

$$w_t'(\mathbf{x}_i^t) = d(x) - H(G_c(G_f(\mathbf{x}_i^t)))/log(|\mathcal{C}_s|) \quad (1)$$

For comparison, we also adopt a variant of our ADCL model named w/o two curriculums, which can be obtained by removing $w_s(\mathbf{x}_i^s)$, indicator function $\mathbb{1}_{w_t(\mathbf{x}_i^t) \geqslant w_\alpha(t)}$ and loss $L_{div}$. The model w/o two curriculums still utilizes the transfer score for final predictions. Results are shown in Table 3. We can observe that the performance of two base UniDA models (UAN and w/o two curriculums) is improved by the proposed adversarial and diverse curriculum learning, though they utilize different metrics to measure the transferability of samples. The results indicate that the proposed adversarial and diverse curriculum learning algorithm can be applied to other adversarial alignment UniDA methods, and it can help to further improve classification performance. Besides, UAN outperforms w/o two curriculums because w/o two curriculums utilizes the naive adversarial loss for cross-domain alignment which cannot constrain the cross-modal alignment into the common label set. Differently, UAN assigns weights to samples in the source and target domain via $w_t'$, which can alleviate the negative transfer. With the help of ADCL, UAN and w/o two curriculums can effectively constrain the cross-modal alignment into the common label set and reduce the classifier's over-reliance so that the performance is improved by 4.77% and 20.15% respectively. It is interesting that the performance gain of w/o two curriculums is much larger than that of UAN when w/o two curriculums and UAN are both equipped with ADCL, due to the following two reasons. (1) In UAN ($w_t'$) + ADCL, the adversarial training loss is defined as Eq (2), which is a little different from the proposed $L_{adv}$.

$$\begin{aligned} L_{adv}^{Universal} = & \mathbb{E}_{\mathbf{x}_i^s \in \mathcal{D}_S} \left[ -w_t'\left(\mathbf{x}_i^s\right) \cdot \log(1 - G_d(G_f(\mathbf{x}_i^s))) \right] \\ & + \mathbb{E}_{\mathbf{x}_i^t \in \mathcal{D}_{UT}} \left[ w_t'(\mathbf{x}_i^t) \cdot \mathbb{1}_{w_t'(\mathbf{x}_i^t) \geqslant w_\alpha(t)} \cdot \log(G_d(G_f(\mathbf{x}_i^t))) \right], \end{aligned} \quad (2)$$

where $w_t'$ is not only used for adversarial curriculum but also used for weighting each selected instance's contribution to the adversarial alignment, which may weaken the cross-domain alignment among high confident samples from the common label set. (2) The values of samples' $w_t'$ are distributed widely and are not sensitive to different samples, which is harmful for sample selection in curriculum learning.

## 2.3. Prototype Classifiers or KNN Classifier for Inferring Target Unknown Instances

To infer target unknown instances with limited data annotated by active learning, two kinds of classifiers can be learned to achieve this goal, *i.e.* prototype classifiers and KNN classifier. Therefore, we apply the prototype classifiers or KNN classifier to the AUAN model, and compare their classification performance in the target domain. Results are shown in Table 4. We can observe that the prototype classifier performs similar to KNN classifier across four datasets. KNN classifier has to store all labeled target features which have to be updated once the model parameters change. Differently, the prototype classifiers only store prototypes for all target categories, which saves storage cost and is robust to model parameters. Therefore, in our paper, we apply prototype classifiers to AUAN for inferring labels in $\tilde{\mathcal{C}}_t$.

## 2.4. Further Remarks

**Performance on $\tilde{\mathcal{C}}_t$.** To show the performance on the target private classes, we also report the average class accuracy among target private classes and the ratio of the identified target private classes in $\tilde{\mathcal{C}}_t$ in two tasks, *i.e.*, Rw $\rightarrow$ Pr and D $\rightarrow$ A. Results are shown in Figure 2. Our approach can recognize more classes in $\tilde{\mathcal{C}}_t$ and annotate more informative instances for prototype classifiers learning than other AL strategies.

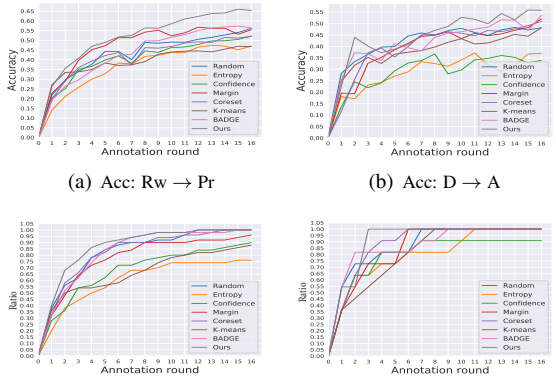
**Performance of varying target private classes $\tilde{\mathcal{C}}_t$.** Following [4], with the fixed $|\mathcal{C}_s \cup \mathcal{C}_t|$ and $|\mathcal{C}_s \cap \mathcal{C}_t|$, we explore

Table 2. Comparison Against State-of-the-Art Universal Domain Adaptation Methods. We report H-score (%) on the Office-Home dataset. The Best results are bolded.

| Method | Office-Home | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A2C | A2P | A2R | C2A | C2P | C2R | P2A | P2C | P2R | R2A | R2C | R2P | AVG |
| UAN | 51.64 | 51.70 | 54.30 | 61.74 | 57.63 | 61.86 | 50.38 | 47.62 | 61.46 | 62.87 | 52.61 | 65.19 | 56.58 |
| CMU | 56.02 | 56.93 | 59.15 | 66.95 | 64.27 | 67.82 | 54.72 | 51.09 | 66.39 | 68.24 | 57.89 | 69.73 | 61.60 |
| DANCE | 59.74 | 68.17 | 75.61 | 61.74 | 62.17 | 71.17 | 66.23 | 58.41 | 70.16 | 67.14 | 58.74 | 72.01 | 65.94 |
| ADCL | **61.97** | **78.10** | **82.31** | **71.48** | **75.71** | **79.46** | **78.07** | **60.20** | **87.90** | **78.03** | **59.44** | **82.09** | **74.56** |

Table 3. Comparison between UniDA models (UAN and our proposed model) with different metrics ($w_t'$ and $w_t$) trained with Adversarial and Diverse Curriculum Learning (ADCL). We report H-score (%) on the Office-Home dataset.

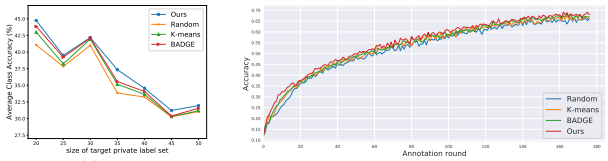
| Different Metrics | A2C | A2P | A2R | C2A | C2P | C2R | P2A | P2C | P2R | R2A | R2C | R2P | AVG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UAN ($w_t'$) | 51.64 | 51.70 | 54.30 | 61.74 | 57.63 | 61.86 | 50.38 | 47.62 | 61.46 | 62.87 | 52.61 | 65.19 | 56.58 |
| UAN ($w_t'$) + ADCL | 53.82 | 63.39 | 67.75 | 64.60 | 59.25 | 63.13 | 60.11 | 50.91 | 65.02 | 66.37 | 52.75 | 69.08 | $\mathbf{61.35}^{\uparrow \mathbf{4.77}}$ |
| w/o two curriculums ($w_t$) | 47.16 | 51.35 | 50.79 | 56.90 | 57.74 | 59.53 | 52.84 | 45.81 | 59.48 | 59.29 | 51.00 | 60.99 | 54.41 |
| ADCL ($w_t$) | 61.97 | 78.10 | 82.31 | 71.48 | 75.71 | 79.46 | 78.07 | 60.20 | 87.90 | 78.03 | 59.44 | 82.09 | $\mathbf{74.56}^{\uparrow \mathbf{20.15}}$ |




(a) Acc: Rw → Pr (b) Acc: D → A

(c) Ratio: Rw → Pr (d) Ratio: D → A

Figure 2. The performance on $\tilde{\mathcal{C}}_t$. (a)(b) are the curves of the average class accuracy among $\tilde{\mathcal{C}}_t$ (Acc). (c)(d) are ratios of the identified target private classes in $\tilde{\mathcal{C}}_t$.

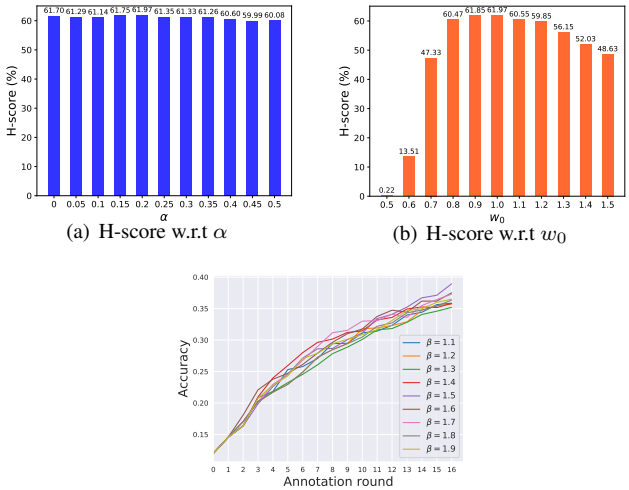



(a) (b)

Figure 3. (a) Average class accuracy varying $\tilde{\mathcal{C}}_t$; (b) Curves of average class accuracy.

the average class accuracy with the various sizes of target private classes $\tilde{\mathcal{C}}_t$ on task A2C of Office-Home dataset. As shown in Figure 3(a), the proposed model outperforms all competitors consistently with different $\tilde{\mathcal{C}}_t$.

**Training speed and convergence.** We explore the curve of average class accuracy over 170 rounds of active learning on task A2C of Office-Home dataset. The result is shown in Figure 3(b), indicating that the training speed and convergence of our model is the fastest.




(a) H-score w.r.t $\alpha$ (b) H-score w.r.t $w_0$

(c) The curves of average class accuracy w.r.t $\beta$

Figure 4. (a) (b): H-score with respect to $\alpha$ and $w_0$; (c): The curves of average class accuracy respect to $\beta$ on the task Ar→Cl of Office-Home dataset.

**Selecting $\alpha$ and $w_0$.** $\alpha$ and $w_0$ are two important hyper-parameters which mainly take effects in the adversarial and diverse curriculum learning. For simplicity, we utilize the ADCL model, not the full AUAN model, to select suitable $\alpha$ and $w_0$. The classification performance of ADCL is evaluated on task Ar → Cl of Office-Home dataset under the UniDA setting. We fix the $\alpha$ and $w_0$ for other transfer tasks across datasets. As shown in Figure 4(a), with $\alpha$ varying in a reasonable range [0, 0.5], the H-score changes little, which proves that the performance is not very sensitive to the $\alpha$. Since the H-score when $\alpha = 0.2$ is the best, we set $\alpha = 0.2$ for all transfer tasks in our experiments. As shown in Figure 4(b), we visualize the H-score with $w_0$

Table 4. Comparing the classification performance of prototype and KNN classifiers equipped with different active learning strategies.

| | | Random | | Badge | | Ours | |
|---|---|---|---|---|---|---|---|
| | | Prototype | KNN | Prototype | KNN | Prototype | KNN |
| Office-Home | Ar→Cl | 29.33 | 28.61 | 30.53 | 29.29 | 36.51 | 35.84 |
| | Ar→Pr | 60.05 | 61.26 | 58.44 | 58.49 | 67.96 | 67.82 |
| | Ar→Rw | 48.32 | 47.94 | 49.79 | 49.28 | 61.02 | 59.18 |
| | Cl→Ar | 47.27 | 46.75 | 44.59 | 44.90 | 53.75 | 52.25 |
| | Cl→Pr | 56.53 | 57.71 | 58.63 | 58.29 | 65.82 | 65.82 |
| | Cl→Rw | 47.20 | 47.22 | 48.00 | 47.34 | 61.88 | 60.13 |
| | Pr→Ar | 52.78 | 51.92 | 54.68 | 53.42 | 54.26 | 52.74 |
| | Pr→Cl | 29.25 | 28.85 | 29.91 | 29.67 | 39.11 | 37.54 |
| | Pr→Rw | 58.68 | 57.59 | 61.01 | 61.45 | 69.76 | 67.11 |
| | Rw→Ar | 50.11 | 50.99 | 51.79 | 51.47 | 55.03 | 52.89 |
| | Rw→Cl | 25.01 | 24.65 | 25.64 | 25.98 | 36.55 | 34.75 |
| | Rw→Pr | 61.77 | 62.29 | 62.28 | 61.64 | 69.24 | 69.15 |
| | AVG | 47.19 | 47.15 | 47.94 | 47.60 | 55.91 | 54.60 |
| | Average Accuracy: | Prototype: 50.35 | | KNN: 49.78 | | | |
| Office-31 | A→D | 85.23 | 85.49 | 84.45 | 84.45 | 87.21 | 86.73 |
| | A→W | 84.22 | 83.03 | 86.16 | 85.95 | 86.19 | 86.19 |
| | D→A | 64.90 | 64.87 | 68.77 | 69.01 | 68.13 | 67.98 |
| | D→W | 91.35 | 90.56 | 91.34 | 91.34 | 92.26 | 92.26 |
| | W→A | 62.84 | 62.73 | 65.74 | 66.09 | 65.67 | 66.26 |
| | W→D | 86.33 | 89.38 | 86.59 | 87.01 | 87.60 | 87.90 |
| | AVG | 79.15 | 79.34 | 80.51 | 80.64 | 81.18 | 81.22 |
| | Average Accuracy: | Prototype: 80.28 | | KNN: 80.40 | | | |
| VisDA | S→R | 63.15 | 63.16 | 64.00 | 63.76 | 73.91 | 73.85 |
| | Average Accuracy: | Prototype: 67.02 | | KNN: 66.92 | | | |
| DomainNet | P→R | 48.56 | 48.99 | 40.31 | 40.31 | 54.00 | 54.59 |
| | R→P | 36.24 | 37.26 | 37.31 | 37.30 | 38.12 | 38.44 |
| | P→S | 29.53 | 30.99 | 30.69 | 31.94 | 31.55 | 33.76 |
| | S→P | 33.98 | 35.35 | 35.70 | 35.83 | 36.71 | 37.55 |
| | R→S | 29.49 | 30.76 | 31.13 | 31.79 | 32.35 | 33.36 |
| | S→R | 47.46 | 50.27 | 47.38 | 47.41 | 48.77 | 49.18 |
| | AVG | 37.54 | 38.94 | 37.09 | 37.43 | 40.25 | 41.15 |
| | Average Accuracy: | Prototype: 38.29 | | KNN: 39.17 | | | |

varying in [0.5, 1.5]. Since $w_0$ can decide a target sample is predicted as a class in the source label set or a class in the target private label set, it is very important to determine a suitable value for $w_0$. We can obtain that the H-score when $w_0 = 1.0$ is the best. Besides, due to the transfer score $w_t(\mathbf{x}_i^t) \in [0, 2]$, setting $w_0 = 1.0$ can equally divide the value range of $w_t$, which removes the classification bias. That is, a target sample is equally predicted as known classes or unknown classes. Therefore, we set $w_0 = 1.0$ in our experiments.

**Selecting $\beta$.** $\beta$ is the margin to decide whether a cluster is from known classes or not during active learning. If a cluster satisfies $w_t(\mathbf{u}_i) > \beta$, the cluster's category is from the shared common label set $\mathcal{C}_c$ and their samples are leveraged to construct pseudo labeled target dataset $\mathcal{D}_{PLT}$. We investigate how to select suitable $\beta$ on task Ar → Cl of Office-Home dataset under the AUDA setting. As shown in Figure 4(c), with $\beta$ varying in a reasonable range [1.1, 1.9], the performance curves change little, which proves that the performance is not very sensitive to the $\beta$. However, the average performance of $\beta$ with high and low values seems a little poor. When $\beta = 1.5$, the average performance seems better. Therefore, we set $\beta = 1.5$ in our experiments.

**Why the bottleneck layer in $G_p$ does not share parameters with that in $G_d$ or $G_c$.** In AUAN, $G_d$ and $G_c$ share a bottleneck layer which embeds visual features extracted by $G_f$ into 256-dimensional features. $G_p$ apply a new a bottleneck layer to embed visual features extracted by $G_f$ for prototypes learning. Since classifier $G_c$ overfit source domain, $G_c$ easily classify target unknown instances as source classes with high confidence, termed as over-reliance. The $G_c$'s over-reliance is attributed to mode collapse of the bottleneck layer which prefer learning patterns related to source classes. Therefore, sharing the bottleneck layer with $G_p$ or $G_c$ cannot help learn discriminative representations for target unknown instances. We conduct experiments that whether $G_p$ shares parameters of bottleneck layer with $G_d$ or $G_c$. The results on Office-Home dataset are shown in Figure 5. The single branch refers to the model where the bottleneck layer in $G_p$ shares parameters with that in $G_d$ or $G_c$. Two branch model refers to the proposed model where the bottleneck layer in $G_p$ does not share parameters with that in $G_d$ or $G_c$. We can observe that sharing parameters (single branch) lead to significant performance drop, which proves that the bottleneck layer in $G_d$ or $G_c$ cannot learn well discriminative representations for target unknown instances.

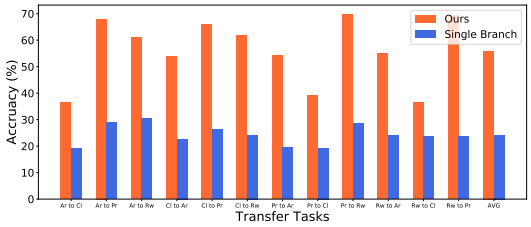



Figure 5. Two branch (Ours) vs Single branch: Average Class Accuracy on transfer tasks in Office-Home dataset.

## 2.5. More Performance Curves

In Figure 6, 7 and 8, we show full performance curves on Office-Home, Office-31, VisDA and DomainNet datasets. We fix the domain adaptation method as ADCL and vary the active learning methods (seven prior work) for comparison. The proposed CNTGE strategy is robust to domain gap and semantic shift, and could select informative instances for AUDA.

## 3. Implementation Details

In this section, we will introduce more details about network architecture and optimization in our experiments.

### 3.1. Network Architecture

For simplicity, we define some notations about various layers in deep networks. Let `fc-k` denotes a fully-connected with `k`-dimensional output. `drop` denotes dropout layer. `relu` and `sigmoid` represent two kinds of activation functions. `softmax` represent the softmax operation. `GRL` denotes gradient reverse layer. `bottleneck-256` denotes a fully-connected with 256-dimensional output. `Cosine Similarity` denotes the operation of calculating the cosine similarity between the inputs and prototypes in the $G_p$.

**Feature Extractor** $G_f$. In our experiments, we apply ResNet-50 as feature extractor. All source and target samples are fed into feature extractor to obtain ***visual features***.

**Domain Discriminator** $G_d$. The Domain discriminator inputs visual features and outputs ***domain predictions***, *i.e.*, ***visual features*** $\rightarrow$ `bottleneck-256` $\rightarrow$ `GRL` $\rightarrow$ `fc-1024` $\rightarrow$ `relu` $\rightarrow$ `drop` $\rightarrow$ `fc-1024` $\rightarrow$ `relu` $\rightarrow$ `drop` $\rightarrow$ `fc-1` $\rightarrow$ `sigmoid` $\rightarrow$***domain predictions***.

**Classifier** $G_c$. Classifier inputs visual features and generates ***label predictions*** in source label set, *i.e.*, ***visual features*** $\rightarrow$ `bottleneck-256` $\rightarrow$ `fc-k` $\rightarrow$ `softmax` $\rightarrow$***label predictions*** where $k$ is the size of source label set. Note that the `bottleneck-256` in $G_d$ and $G_c$ share parameters.

**Prototype Classifiers** $G_p$. The prototype classifiers input visual features and generates ***label predictions*** in target private label set, *i.e.*, ***visual features*** $\rightarrow$ `bottleneck-256` $\rightarrow$ `Cosine Similarity` $\rightarrow$ `softmax` $\rightarrow$***Unknown Category Inference***.

### 3.2. Optimization

- Batch Size: 36 for each domain
- Loss Weight of $L_{adv}$ and $L_{div}$: 1.0
- Loss Weight of $L_{nc}$: 0.01
- Threshold $w_0$: 1.0
- $\alpha = 0.2$
- Learning Rate: 0.01
- Optimizer: The stochastic gradient decent with 0.9 momentum is used, and the learning rate is annealed by $\mu_p = \frac{\mu_0}{(1+\alpha \cdot p)^\beta}$, where $\mu_0 = 0.01$, $\alpha = 10$, and $\beta = 0.75$. We set the learning rate for fine-tuned layers to be 0.1 times of that from scratch.

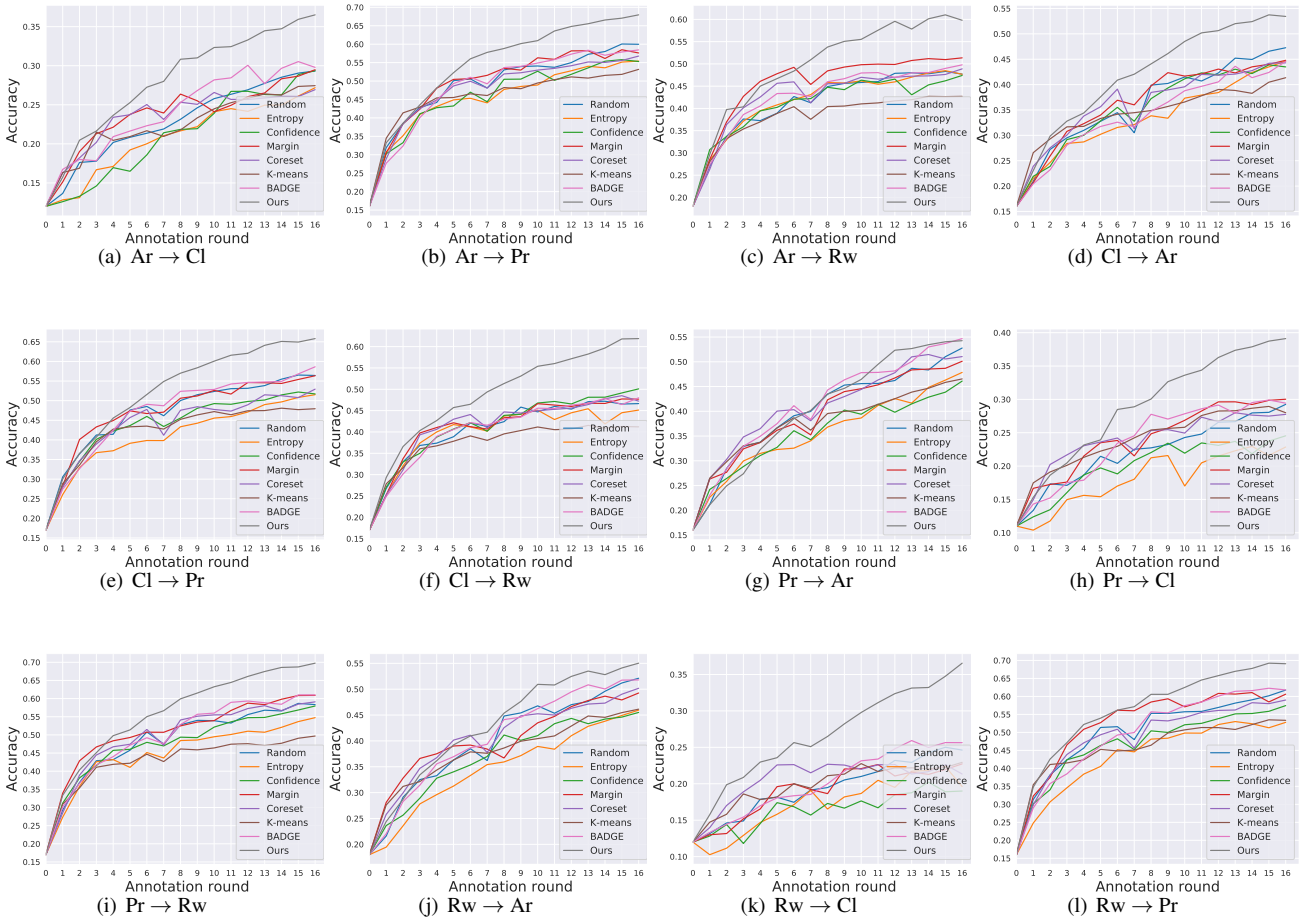
## References


[1] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *ICLR*, 2019.

[2] Bo Fu, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Learning to detect open classes for universal domain adaptation. In *ECCV*, pages 567–583, 2020.

[3] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, and Kate Saenko. Universal domain adaptation through self-supervision. In *NeurIPS*, pages 16282–16292, 2020.

[4] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *CVPR*, pages 2720–2729, 2019.

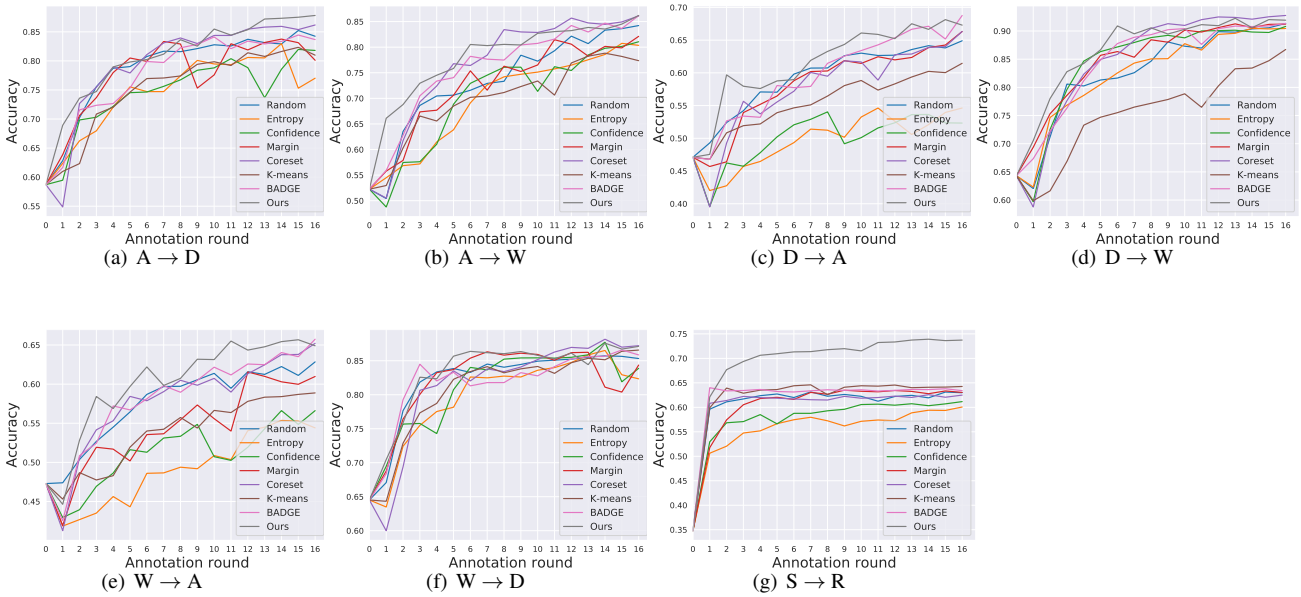



(a) Ar → Cl (b) Ar → Pr (c) Ar → Rw (d) Cl → Ar

(e) Cl → Pr (f) Cl → Rw (g) Pr → Ar (h) Pr → Cl

(i) Pr → Rw (j) Rw → Ar (k) Rw → Cl (l) Rw → Pr

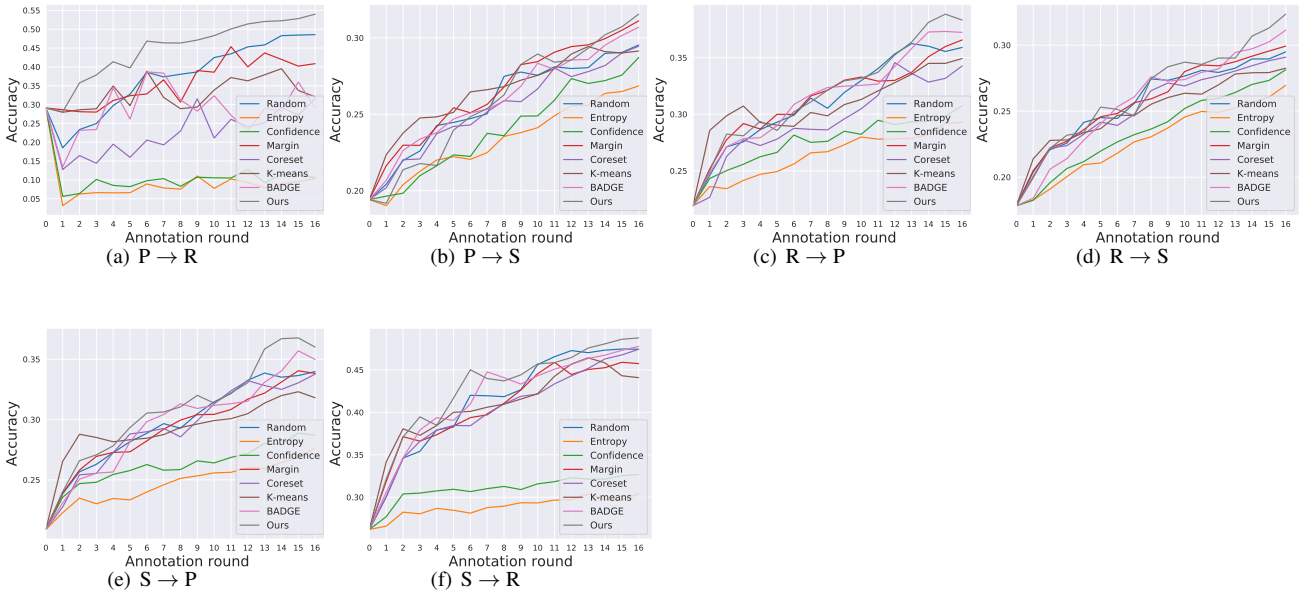
Figure 6. The curves of average class accuracy with different annotation round on Office-Home dataset. *Best viewed in high resolution.*




(a) A → D (b) A → W (c) D → A (d) D → W

(e) W → A (f) W → D (g) S → R

Figure 7. The curves of average class accuracy with different annotation round on Office-31 and VisDA datasets. *Best viewed in high resolution.*




(a) P → R (b) P → S (c) R → P (d) R → S

(e) S → P (f) S → R

Figure 8. The curves of average class accuracy with different annotation round on DomainNet datasets. *Best viewed in high resolution.*