Supplementary Materiel
D^{3}GU: Multi-target Active Domain Adaptation via Enhancing Domain Alignment

Lin Zhang[†] Linghan Xu[†] Saman Motamed[†] Shayok Chakraborty[‡] Fernando De la Torre[†]

[†]Robotics Institute, Carnegie Mellon University

[‡]Department of Computer Science, Florida State University

1. Datasets

We provide a statistical summary of the three datasets (Office31 [9], OfficeHome [12], DomainNet [7]) in Table 1 and overviews of them in Figure 3,4,5.

2. Implementation

2.1. Architecture

backbone *F* The backbone consists of a ResNet-50 [5] pretrained on ImageNet-1K [3] provided by PyTorch model zoo [1] followed by a single fully connected layer downsizing the feature vector dimension from 2048 to e. e is set to 256 for Office31/OfficeHome and 512 for DomainNet, since DomainNet has larger capacity.

classifier *C* The classifier is a single fully connected layer with input dimension e and output dimension K, where K equals to number of classes.

domain discriminator *D* The domain discriminator is a three-layer fully connected network with ReLU activations and Dropouts of 0.5. Hidden dimensions are set to 512 and 1024 for Office31/OfficeHome and DomainNet, respectively.

gradient reversal layer When doing backpropagation, it weights and reverses the sign of the gradients transferred from D to F to achieve adversarial learning. During pretraining, the weight $\eta = \frac{2}{1+\exp(-10p)-1}$ as in original implementation [14], where $p \in [0, 1]$ is the training progress. During domain adapted training in active learning stages, η is constantly set to 1.0 since it is a continuation of pretraining.

2.2. Training recipes

We provide pseudocodes for unsupervised pretraining and domain adapated training in Algorithm 1 and 2 respectively. We use SGD as optimizer with momentum 0.9 and weight decay of 0.005 for stable convergence. The initial learning rate is 0.001 and 0.0003 for unsupervised pretraining and domain adapted training stages, respectively. The learning rate of backbone F except for the appended fully connected layer is decreased to $\frac{1}{10}$ as normal learning rate. As in [14], we adjust the learning rate as $\eta_p = \eta_0(1 + qp)^{-0.75}$, where p is the training progress. q is set to be 10 for unsupervised pretraining as in [6], and 1.0 for domain adapted training during active learning to ensure a smooth finetuning progress. We define the training epochs w.r.t. the source dataloader, and set them to values to ensure training convergence. On Office31, the numbers of pretrain epochs are 200, 672, 432 for source domains as *amazon*,*dslr*,*webcam* respectively. On OfficeHome, the number is 200 for *art* as source and 120 otherwise. On DomainNet, it is 24 for *quickdraw* as source and 64 otherwise, since we empirically found *quickdraw* as source easily leads to overfitting. The number of training epochs for each active domain adapted training stage is halved instead. During training, we first resize all images into size 256×256, then random resized crop a 224×224 patch from it with scale $[0.08, 1.0]$ and ratios $[\frac{3}{4}, \frac{4}{3}]$, implemented by PyTorch's *RandomResizedCrop* class. We then apply random horizontal flip and Imagenet normalization. During testing, we center crop a 224×224 patch from the resized 256×256 images. Results are averaged from 3 random trials on Office31/OfficeHome and 2 random trials on DomainNet. Batch sizes are set to values to ensure the balance between source and target data, as introduced below.

Unsupervised domain adapted pretraining: We use dataloader D_S to load source data. For the k-th target domain, we have a dataloader $D_{\mathcal{T}_k}$. In each iteration, classification loss is computed as the average loss on the source batch, and domain discrimination loss is computed as the average on all data. We set the batch sizes for the source dataloader and each target dataloader to be 64 and 32 on Office31, 64 and 24 on OfficeHome, and 48 and 8 on DomainNet.

Domain adapted training: Similar as pretraining, we use a source dataloader D_S to load all source data and K dat-

Dataset	class	Domain	Train	Test	Explanation
Office31	31	amazon	2817		Images downloaded from *amazon* website
		dslr	498		Images captured with a digital SLR camera in realistic environments with natural lighting conditions
		webcam	795		Images captured with a simple webcam towards the same objects in *dslr* domain
OfficeHome	65	art	2426		Artistic depictions of objects in the form of sketches, paintings, ornamentation, etc.
		clipart	4364		Collection of clipart images
		product	4438		Images of objects without a background, akin to the *amazon* category in Office31 dataset
		real	4356		Images of objects captured with a regular camera
DomainNet	345	clipart	33525	14814	Collection of clipart images, akin to *clipart* in OfficeHome
		infograph	36023	15582	Infographic images with specific object
		painting	50416	21850	Artistic depictions of objects in the form of paintings
		quickdraw	120750	51750	Drawings of the worldwide players of game “Quick Draw!”
		real	120906	52041	Photos and real world images
		sketch	48212	20916	Sketches of specific objects

Table 1. Details of datasets used in our experiments.

aloaders $D_{\mathcal{T}_k}{}_{k=1}^{K}$ to load all target data. Besides, we also use K dataloaders $D^l_{\mathcal{T}_k}{}_{k=1}^{K}$ to load labeled data in each target domain. Note that $D_{\mathcal{T}_k}$ includes data in $D^l_{\mathcal{T}_k}$. In each iteration, classification loss is averaged on source domain's classification loss and target domains classification loss, while domain discrimination loss is computed as average on source and target all-data batches. We set the batch sizes for the source dataloader, each labeled target dataloader, and each target dataloader to be 64/32/8 on Office31, 64/24/8 on OfficeHome, and 48/8/8 on DomainNet.

2.3. Baselines

We implemented several state-of-the-art active learning sampling algorithms based on their official code implementations. We include the comparison between some baselines and GU-KMeans, and their implementations below:

Coreset [11]: Coreset intends to select a subset of data that have the smallest distance with the original set, where the distance between two sets of data is determined by the smallest pairwise distance between any two samples from the two sets, respectively. However, it ignores contributions of the selected subset towards the classification tasks.

BADGE [2]: BADGE computes the classifier gradient of classification loss supervised by pseudo-labels for each sample. It then finds a subset of samples by applying KMeans++ on the gradient space. Compared with BADGE, GU-KMeans instead computes the gradients in the feature space, which is in much lower dimension thus much more efficient. We also consider domain alignment gradient, and cluster using the more effective KMeans algorithm.

AADA [4]: Since AADA did not make their implementations public, we directly followed their paper to use $\frac{1-p_{\mathcal{S}}}{p_f}$ to weight entropy.

LAMDA [10]: LAMDA proposed multiple techniques, including a cosine classifier, source data resampling during training time, pseudo-labeling to utilize unlabeled data, and a sampling algorithm to minimize the selected subset's MMD with the whole selection pool.

Algorithm 1 Unsupervised domain adapted pretraining

```
Input: source dataloader $D_{\mathcal{S}}$, target dataloaders $\{D_{\mathcal{T}_1}, ..., D_{\mathcal{T}_N}\}$, number of training iterations $te$, model $\theta$, optimizer $opt$
Initialize: training iteration $t \leftarrow 0$
Training:
  while $t < te$ do
      // get source images, classification and domain labels
      $x_{\mathcal{S}}, y_{\mathcal{S}}, m_{\mathcal{S}} \leftarrow D_{\mathcal{S}}.next$
      // get target images and domain labels
      $x_{\mathcal{T}_1}, m_{\mathcal{T}_1} \leftarrow D_{\mathcal{T}_1}.next$
      ...
      $x_{\mathcal{T}_N}, m_{\mathcal{T}_N} \leftarrow D_{\mathcal{T}_N}.next$

      // compute classification loss
      $l_{cls} \leftarrow \mathcal{L}_{cls}(x_{\mathcal{S}}, y_{\mathcal{S}})$
      // compute domain discrimination loss
      $x \leftarrow concat(x_{\mathcal{S}}, x_{\mathcal{T}_1}, ..., x_{\mathcal{T}_N})$
      $m \leftarrow concat(m_{\mathcal{S}}, m_{\mathcal{T}_1}, ..., m_{\mathcal{T}_N})$
      $l_{dom} \leftarrow \mathcal{L}_{dom}(x, m)$

      // update model
      $l \leftarrow l_{cls} + l_{dom}$
      $l.backpropagate$
      $\theta \leftarrow opt.update$
      $t \leftarrow t + 1$
  end while
```

CLUE [8]: CLUE is the most related work to our method. It uses the entropy scores as weights for KMeans sampling. We use the default temperature value of 1.0 as described in the paper. However, entropy score is less effective in measuring sample's contribution to classification task, as shown in Table 4 in our main paper. Besides, although being pro-

Algorithm 2 Domain adapted training in active learning

```
Input: source dataloader $D_{\mathcal{S}}$, labeled target dataloaders $\{D^l_{\mathcal{T}_1}, ..., D^l_{\mathcal{T}_N}\}$, target dataloaders $\{D_{\mathcal{T}_1}, ..., D_{\mathcal{T}_N}\}$, number of training iterations $te$, model $\theta$, optimizer $opt$
Initialize: training iteration $t \leftarrow 0$
Training:
  while $t < te$ do
    // get source images, classes, and domains
    $x_{\mathcal{S}}, y_{\mathcal{S}}, m_{\mathcal{S}} \leftarrow D_{\mathcal{S}}.next$
    // get labeled target images, classes, and domains
    $x^l_{\mathcal{T}_1}, y^l_{\mathcal{T}_1}, m^l_{\mathcal{T}_1} \leftarrow D^l_{\mathcal{T}_1}.next$
    ...
    $x^l_{\mathcal{T}_N}, y^l_{\mathcal{T}_N}, m^l_{\mathcal{T}_N} \leftarrow D^l_{\mathcal{T}_N}.next$
    // get target images and domains
    $x_{\mathcal{T}_1}, m_{\mathcal{T}_1} \leftarrow D_{\mathcal{T}_1}.next$
    ...
    $x_{\mathcal{T}_N}, m_{\mathcal{T}_N} \leftarrow D_{\mathcal{T}_N}.next$

    // compute classification loss
    $x^l_{\mathcal{T}} \leftarrow concat(x^l_{\mathcal{T}_1}, ..., x^l_{\mathcal{T}_N})$
    $y^l_{\mathcal{T}} \leftarrow concat(y^l_{\mathcal{T}_1}, ..., y^l_{\mathcal{T}_N})$
    $l_{cls} \leftarrow 0.5\mathcal{L}_{cls}(x_{\mathcal{S}}, y_{\mathcal{S}}) + 0.5\mathcal{L}_{cls}(x^l_{\mathcal{T}}, y^l_{\mathcal{T}})$
    // compute domain discrimination loss
    $x \leftarrow concat(x_{\mathcal{S}}, x_{\mathcal{T}_1}, ..., x_{\mathcal{T}_N})$
    $m \leftarrow concat(m_{\mathcal{S}}, m_{\mathcal{T}_1}, ..., m_{\mathcal{T}_N})$
    $l_{dom} \leftarrow \mathcal{L}_{dom}(x, m)$

    // update model
    $l \leftarrow l_{cls} + l_{dom}$
    $l.backpropagate$
    $\theta \leftarrow opt.update$
    $t \leftarrow t + 1$
  end while
```

posed as an active domain adaptation sampling algorithm, CLUE did not consider domain shifts in its algorithm. GU-KMeans instead uses the more effective gradient value to consider each sample's contributions to both classification and domain alignment.

SDM [13]: SDM is an improved version of margin sampling and was proposed to maximize the prediction margin for classification. It is composed of a dynamically adjusted margin loss and a query score to consider gradient direction consistency, which attempts to push a sample's feature representation $\boldsymbol{z}$ toward direction that minimizes the margin sampling function. SDM, however, does not consider domain shift either. In comparison, our GU-KMeans instead considers gradient correlation between classification task and domain alignment task, and explicitly measures sample's contribution in the feature space.

	α						all-way
	0(binary)	0.1	0.2	0.3	0.4	0.5	
D'	1.3579	1.3541	1.3516	1.3373	1.3280	1.3220	1.2974

Table 2. Average normalized domain distances with varying α values in domain discrimination.

3. Distance of aligned domains

To measure the degree of alignment between domains, we explicitly measure domain distances after unsupervised pretraining. We take $quickdraw \rightarrow rest$ as the multi-target domain adaptation setting and measure the domain distances on the test data. Given encoded features $\{\boldsymbol{z}_i^{\mathcal{D}_1}\}_{i=1}^{N_1}$ and $\{\boldsymbol{z}_i^{\mathcal{D}_2}\}_{i=1}^{N_2}$ from two domains $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, we define the domain distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ to be:

$$D(\mathcal{D}_1, \mathcal{D}_2) = \frac{1}{N_1 \times N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \|\boldsymbol{z}_i^{\mathcal{D}_1} - \boldsymbol{z}_j^{\mathcal{D}_2}\|_2 \tag{1}$$

Due to randomness in the training progress, the encoded feature space may not be in exactly the same scale. Consequently, Equation 1 is not scale-invariant. To avoid influence of feature space scales, we normalize the computed distance by the average distance of source samples:

$$D(\mathcal{S}, \mathcal{S}) = \frac{2}{N_{\mathcal{S}}(N_{\mathcal{S}} - 1)} \sum_{i=1}^{N_{\mathcal{S}}} \sum_{j=i+1}^{N_{\mathcal{S}}} \|\boldsymbol{z}_i^{\mathcal{S}} - \boldsymbol{z}_j^{\mathcal{S}}\|_2 \tag{2}$$

$$D'(\mathcal{D}_1, \mathcal{D}_2) = \frac{D(\mathcal{D}_1, \mathcal{D}_2)}{D(\mathcal{S}, \mathcal{S})} \tag{3}$$

Such a normalized distance measure will be invariant to the scale changes of features due to randomness of training, thus can be used to measure the extent of domain alignment between different algorithms.

We report the average pairwise domain distances of varying α values in Equation 4 in the main paper in Table 2. As shown in the table, all-way discrimination indeed leads to smaller average domain distances thus a more compact feature space. By increasing value of α, the domains are aligned better.

4. Visualization of selected sample distributions

In Figure 1, we visualize the distribution of selected samples in the feature space using PCA, taking the first active learning stage with *art* as source on OfficeHome. As shown in the figure, GU-KMeans not only selects many uncertain data, but also results in a much more diverse uncertain-data distribution compared to CLUE, LAMDA, and SDM.

5. Multi-target active domain adaptation results

We include more MT-ADA results in Table 3 and 4, where GU-KMeans demonstrates superior performance compared to other baselines with any domain adaptation methods. Decomposed domain discrimination can also improve upon binary and all-way alignment when combined with multiple active sampling baselines, as shown in Table 3.

6. Single-target active domain adaptation results

We conducted single-target active domain adaptation experiments on OfficeHome with average results from 3 random trials. We apply active learning for 4 stages with 100 annotation budget in each stage. As shown in Figure 2, performance of GU-KMeans sampling (red) still stays at the top in almost all $source \rightarrow target$ settings, except for adaptation between art and real, where score-based methods (AADA, margin, SDM) stand out to be more superior.

References

[1] Paszke Adam, Gross Sam, Massa Francisco, Lerer Adam, Bradbury James, Chanan Gregory, Killeen Trevor, Lin Zeming, Gimelshein Natalia, Antiga Luca, Desmaison Alban, Köpf Andreas, Yang Edward, DeVito Zach, Raison Martin, Tejani Alykhan, Chilamkurthy Sasank, Steiner Benoit, Fang Lu, Bai Junjie, and Chintala Soumith. Pytorch: An imperative style, high-performance deep learning library. In *NeuraIPS*, 2019. 1

[2] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. 2020. 2

[3] Deng Jia, Dong Wei, Socher Richard, Li Li-Jia, Li Kai, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[4] Su Jong-Chyi, Tsai Yi-Hsuan, Sohn Kihyuk, and Buyu Liu. Active adversarial domain adaptation. In *CVPRW*, 2019. 2

[5] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[6] Long Mingsheng, Cao Zhangjie, Wang Jianmin, and I. Jordan Michael. Conditional adversarial domain adaptation. In *NeurIPS*, 2018. 1

[7] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019. 1

[8] Viraj Prabhu, Arjun Chandrasekaran, Kate Saenko, and Judy Hoffman. Active domain adaptation via clustering uncertainty-weighted embeddings. In *ICCV*, 2021. 2

[9] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, 2010. 1

[10] Hwang Sehyun, Lee Sohyun, Kim Sungyeon, Ok Jungseul, and Kwak Suha. Combating label distribution shift for active domain adaptation. In *ECCV*, 2022. 2

[11] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018. 2

[12] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017. 1

[13] Ming Xie, Yuxi Li, Yabiao Wang, Zekun Luo, Zhenye Gan, Zhongyi Sun, Mingmin Chi, Chengjie Wang, and Pei Wang. Learning distinctive margin toward active domain adaptation. In *CVPR*, 2022. 3

[14] Ganin Yaroslav and Lempitsky Victor. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 1

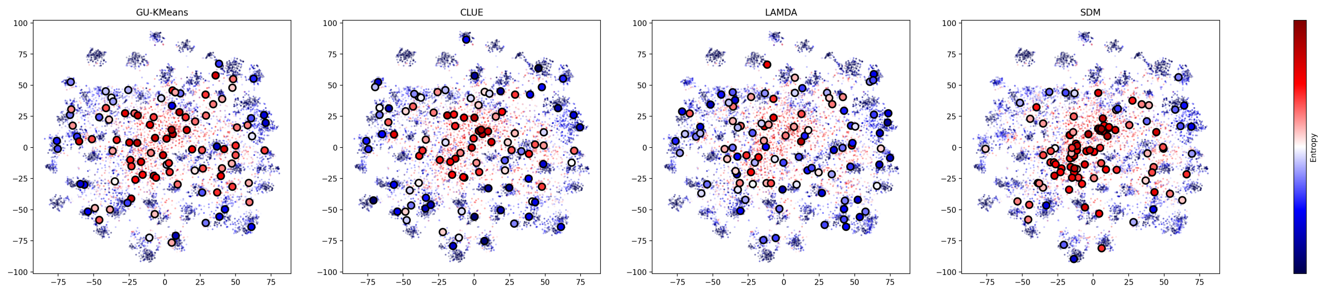

Figure 1. PCA visualization of selected subsets at the first active learning stage, with art as source on OfficeHome. Red stands for high entropy score and blue stands for low entropy score.

DA	Method	Office31								OfficeHome									
		budget=30/stage-1				budget=120/stage-4				budget=100/stage-1					budget=400/stage-4				
		amzn	dslr	web	AVG	amzn	dslr	web	AVG	art	clip	prod	real	AVG	art	clip	prod	real	AVG
Binary	random	85.38	81.14	81.44	82.65	92.64	85.03	85.75	87.80	63.37	62.43	56.78	63.64	61.55	68.20	68.47	61.99	68.15	66.70
	entropy	89.62	81.11	81.53	84.08	98.16	86.44	87.20	90.60	63.27	63.80	57.17	63.44	61.92	70.05	71.35	65.25	70.19	69.21
	margin	89.80	82.24	83.46	85.16	**98.92**	87.89	87.75	91.52	63.75	64.98	57.90	64.31	62.73	71.12	71.86	**65.79**	70.64	69.85
	coreset	85.52	80.50	81.08	82.36	90.51	83.22	83.41	85.71	62.00	62.75	55.23	62.94	60.73	66.48	67.22	60.38	66.47	65.14
	BADGE	89.16	81.61	82.86	84.54	98.54	86.53	87.82	90.97	64.25	64.65	57.38	64.57	62.71	70.70	72.14	64.57	70.28	69.42
	AADA	89.48	81.78	82.64	84.63	98.33	86.15	87.25	90.58	63.15	63.44	57.48	63.73	61.95	69.77	71.33	64.53	69.91	68.88
	SDM	89.67	82.28	83.35	85.10	98.83	87.57	88.09	91.50	63.79	65.32	57.79	64.31	62.80	70.85	72.72	65.26	70.40	69.81
	LAMDA	90.12	81.53	82.66	84.77	98.67	87.36	87.88	91.30	64.31	64.29	57.87	64.93	62.85	70.37	71.67	65.13	70.39	69.39
	CLUE	90.22	**83.61**	83.92	85.92	97.73	87.80	88.49	91.34	**65.39**	**65.94**	58.17	**65.27**	63.69	71.68	71.83	64.54	70.35	69.60
	GU-KMeans	**90.68**	83.00	**85.58**	**86.42**	98.65	**87.98**	**89.38**	**92.00**	65.32	65.92	**58.46**	65.11	**63.70**	**72.40**	**72.94**	65.66	**70.71**	**70.43**
All-way	AADA	89.32	80.70	82.76	84.26	98.34	86.01	87.10	90.48	63.72	64.56	57.85	64.89	62.76	70.28	71.65	64.87	70.32	69.28
	SDM	89.85	81.94	83.55	85.11	**99.06**	87.62	88.32	91.67	64.23	64.85	58.09	65.65	63.21	71.03	72.72	65.88	70.92	70.13
	LAMDA	90.74	82.34	83.49	85.52	98.57	87.60	88.30	91.49	65.44	64.37	58.55	65.74	63.52	70.87	71.61	65.46	70.58	69.63
	CLUE	90.74	**84.21**	84.12	86.36	97.93	87.74	88.86	91.51	66.02	65.36	**59.40**	65.79	64.14	71.75	71.33	64.89	70.56	69.63
	GU-KMeans	**91.17**	83.88	**85.18**	**86.74**	98.62	**88.45**	**89.23**	**92.10**	**66.23**	**66.73**	59.19	**65.87**	**64.50**	**72.42**	**72.90**	**66.04**	**71.36**	**70.68**
Decomposed	AADA	89.50	81.85	82.85	84.73	98.08	86.80	87.94	90.94	63.91	64.81	58.16	65.53	63.10	70.96	71.98	65.11	70.57	69.65
	SDM	89.95	82.28	82.96	85.06	99.05	87.33	88.07	91.48	64.51	64.82	58.29	65.51	63.28	71.33	72.30	**65.78**	71.21	70.15
	LAMDA	91.09	83.46	84.29	86.28	98.70	88.24	88.16	91.70	64.94	64.94	58.87	65.57	63.58	71.05	71.78	**65.78**	70.47	69.77
	CLUE	**91.15**	84.22	84.61	86.66	98.16	88.40	88.65	91.73	**66.25**	64.93	59.23	65.62	64.01	71.70	71.20	65.15	70.69	69.68
	GU-KMeans	91.14	**84.23**	**85.16**	**86.84**	**99.16**	**88.55**	**89.29**	**92.33**	65.96	**66.53**	**59.29**	**66.30**	**64.52**	**72.36**	**72.65**	65.75	**71.43**	**70.55**

Table 3. MT-ADA accuracies with total budget 120 and 400 on Office31 and OfficeHome, respectively. We conducted 4 active learning stages with equal budgets. We mark the best results in **bold** and underline the second-best ones. The best average results across all the source domains on each dataset are marked in **red**.

DA	Method	C	I	P	Q	R	S	AVG
binary	SDM	43.06	43.26	43.82	40.42	42.97	46.41	43.32
	LAMDA	44.08	46.34	44.75	40.73	43.25	46.94	44.35
	CLUE	43.79	46.48	44.41	40.03	43.01	46.71	44.07
	GU-KMeans	44.16	**47.50**	**44.86**	40.61	**43.55**	**47.18**	**44.64**
all-way	GU-KMeans	42.72	46.06	43.52	39.95	41.17	44.94	43.06
decomposed	GU-KMeans	**44.40**	47.23	44.82	**40.90**	43.23	47.13	44.62

Table 4. MT-ADA accuracies on DomainNet with total budget $=10,000$. Capital letters are short for source domain names.

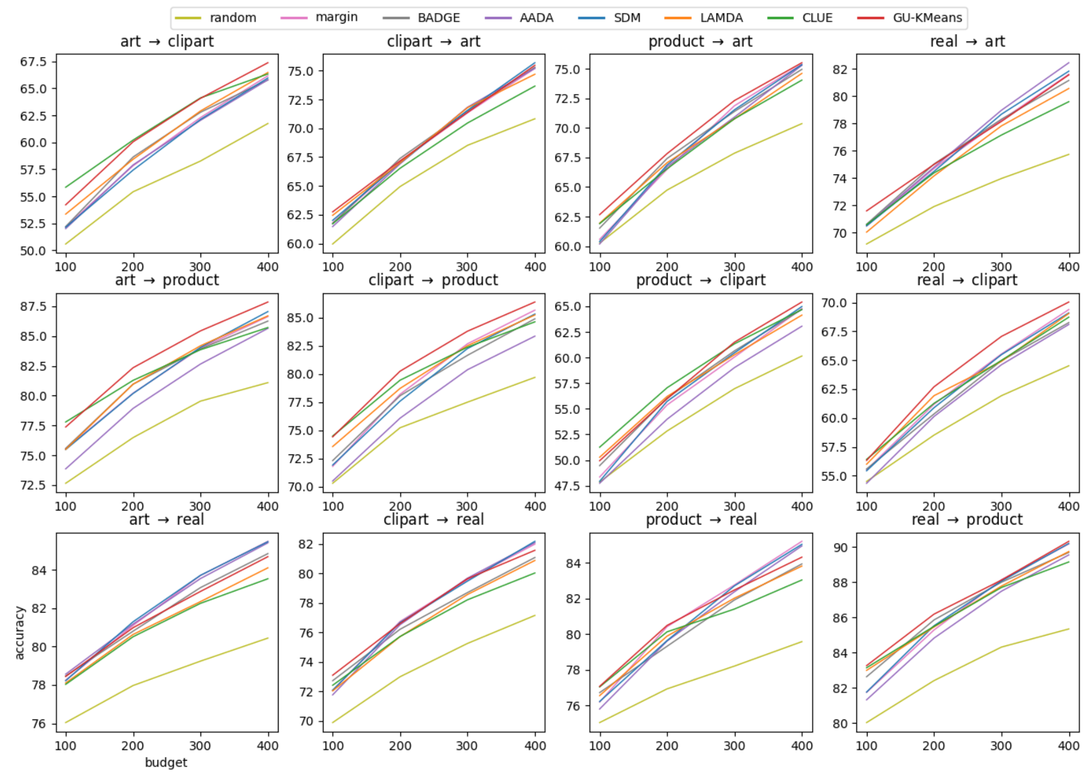

Figure 2. ST-ADA results on OfficeHome, with 100 annotation budget in each of the 4 active learning stages.

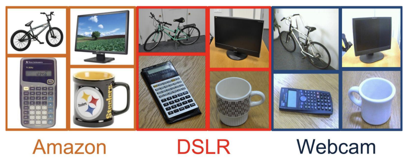

Figure 3. Office31 overview.

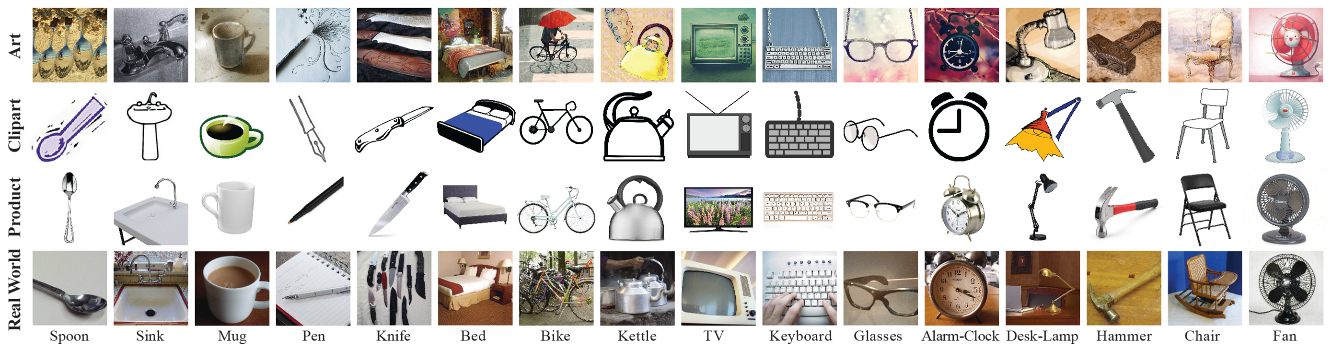

Figure 4. OfficeHome overview.

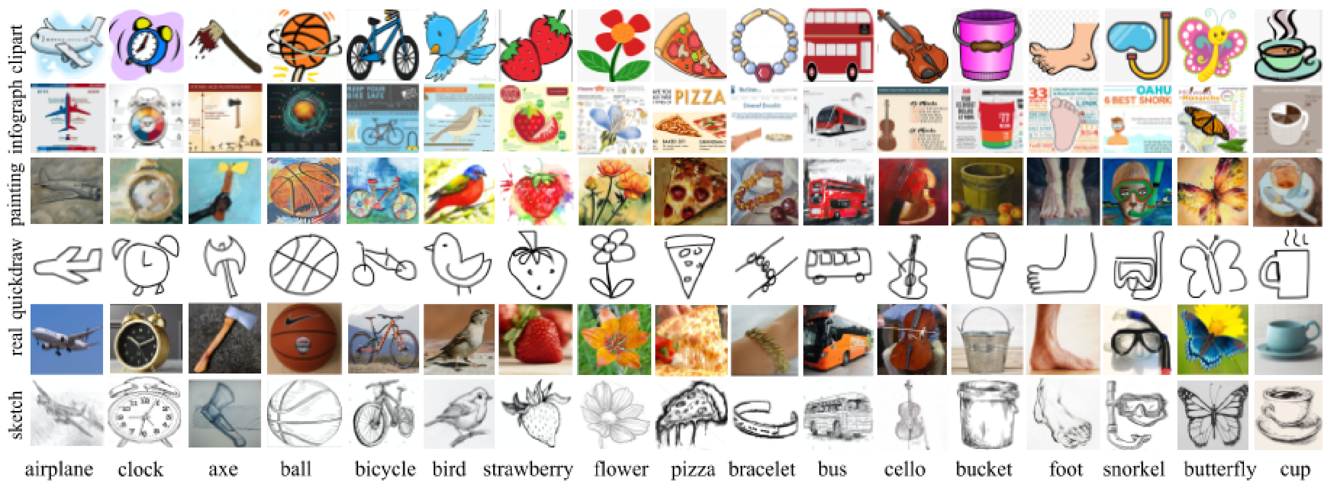

Figure 5. DomainNet overview.