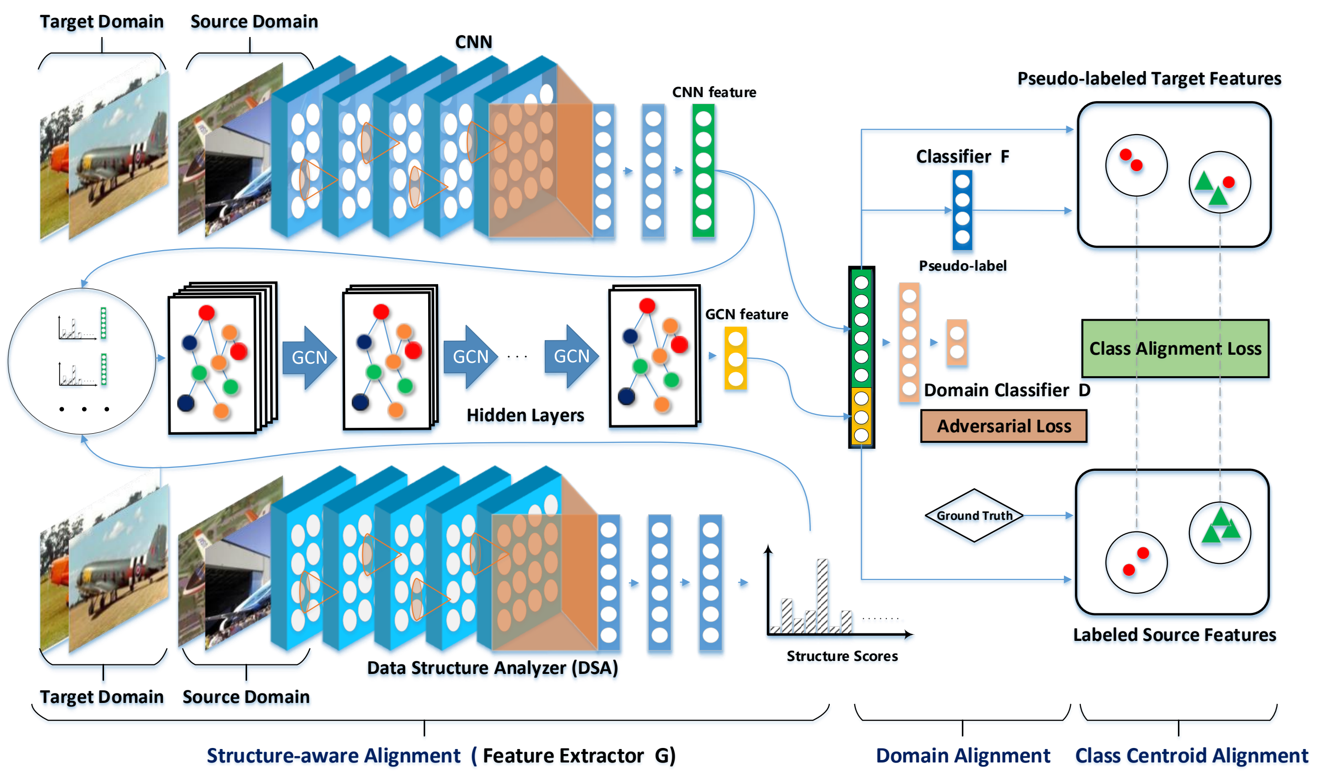


Figure 1. The architecture of the proposed Graph Convolutional Adversarial Network (GCAN). Our GCAN consists of three alignment mechanisms including structure-aware alignment, domain alignment, and class centroid alignment. In the structure-aware alignment, the Data Structure Analyzer network generates structure scores encoded with source data structure information while the CNN features are extracted by CNNs. Then, the structure scores and CNN features are used to construct dense-connected instance graphs for the GCN. The concatenated CNN and GCN features are input to the domain alignment and class centroid alignment modules. In the domain alignment, a domain alignment loss is applied to match the global domain statistics. In the class centroid alignment, pseudo-labeled target features and labeled source features are used to calculate the class centroid alignment loss to ensure that samples with same class from different domains can be embedded closely. For more details, please refer to the main body of the paper.

GCAN: Graph Convolutional Adversarial Network for Unsupervised Domain Adaptation

Xinhong Ma[1,2,3], Tianzhu Zhang[1,2,4] and Changsheng Xu[1,2,3]
[1] National Lab of Pattern Recognition (NLPR),
Institute of Automation, Chinese Academy of Sciences (CASIA)
[2] University of Chinese Academy of Sciences (UCAS)
[3] Peng Cheng Laboratory, ShenZhen, China
[4] University of Science and Technology of China
{xinhong.ma, csxu}@nlpr.ia.ac.cn, tzzhang10@gmail.com

1. Additional Experiments

In this section, we first illustrate the datasets, baseline methods and implementation details. Then, we show extensive experimental results and analysis on Office-Caltech and MNIST-USPS-SVHN. After that, we explore the ablation study and parameters analysis on GCAN. Finally, we show quantities of visualization results on Office-31, ImageCLEF and Office-Home.

1.1. Datasets

Office-Caltech released by [2], is comprised of 10 common categories shared by the Office-31 and Caltech-256 datasets. In our experiments, we construct 12 tasks across 4 domains: Amazon (A), Webcam (W), DSLR (D) and Caltech (C), with 958, 295, 157 and 1,123 image samples respectively.

MNIST-USPS-SVHN. We explore three digits datasets: MNIST [4], USPS and SVNH [9]. Different from Office-31, MNIST consists grey digits images of size 28×28, USPS contains 16×16 grey digits and SVHN composes color 32×32 digits images which might contain more than one digit in each image. MNIST-USPS-SVHN makes a good complement to previous datasets for diverse domain adaptation scenarios. We conduct experiments on two tasks: SVHN$\rightarrow$MNIST and MNIST$\rightarrow$USPS. Digit images are also cast to $28\times28\times1$ in all experiments for fair comparison.

1.2. Baseline Methods

As for the Office-Caltech dataset, we compare with some representative shallow and deep transfer learning methods: Transfer Component Analysis (TCA) [11], Geodesic Flow Kernel (GFK) [2], Deep Domain Confusion (DDC) [14], Gradient Reversal (RevGrad) [1], Residual Transfer Network (RTN) [6], and Moving Semantic Transfer Network (MSTN) [15]. All results are directly cited from these published papers except MSTN whose results are obtained by runing the released code[1] by ourselves.

To further validate our method, we also conduct experiments on MNIST-USPS-SVHN. We compare with Deep Domain Confusion (DDC) [14], RevGrad [1], Asymmetric Tri-Training [12], Couple GAN (CoGAN) [5], Label Efficient Learning (LEL) [7], Adversarial Discriminative Domain Adaptation (ADDA) [13], Moving Semantic Transfer Network (MSTN) [15]. Results of DDC, RevGrad, CoGAN, ADDA and MSTN are cited from [13]. For the rest, we cite the results in their papers respectively.

1.3. Implementation Details

For Office-Caltech dataset, we apply the same network architecture and experimental settings as Office-31, ImageCLEF-DA and Office-Home. For digits datasets, we redesign four modules in our model, *i.e.*, CNNs, Data Structure Analyzer (DSA), Domain Classifier, and GCNs. For CNNs, we use the same architecture as ADDA [13]: two convolution layers followed by max pool layers and two fully connected layers are placed behind. The Data Structure Analyzer (DSA) is implemented as the LeNet trained on source data with the triplet loss. For the architecture of GCNs, we only use one GCN. The dimension of node representation is 10 and the output dimension is 3. For Domain Classifier, we also use the same architecture as ADDA, *e.g.*, x$\rightarrow$500$\rightarrow$500$\rightarrow$1. Batch Normalization is inserted between convolutional layers.

1.4. Results and Analysis

Office-Caltech. From the results, we can draw the following conclusions. (1) Standard deep learning methods (AlexNet) perform comparably with traditional shallow transfer learning methods with deep DeCAF7 features as input (TCA and GFK). The only difference between these two sets of methods is that AlexNet can take the advantage of supervised fine-tuning on the source-labeled data, while TCA and GFK can take benefits of their domain adaptation procedures. This result confirms the current practice that supervised fine-tuning is important for transferring source classifier to target domain [10]. (2) Deep transfer learning methods that reduce the domain discrepancy by domain-adaptive deep networks (DDC, DAN, RevGrad, RTN and MSTN), substantially outperform standard deep learning methods (AlexNet) and traditional shallow transfer learning methods with deep features as the input (TCA and GFK). (3) The proposed GCAN performs the best and proves the effectiveness of jointly leveraging data structure, domain label and class label information in a unified deep network.

MNIST-USPS-SVHN. We follow the protocols in [13]: For adaptation between SVHN and MNIST, we use the training set of SVHN and test set of MNIST for evaluation. For adaptation between MNIST and USPS, we randomly sample 2000 images from MNIST and 1800 from USPS. For SVHN$\rightarrow$MNIST, the transfer gap is huge since images in SVHN might contain

[1]https://github.com/Mid-Push/Moving-Semantic-Transfer-Network

multiple digits. Results of MNIST-USPS-SVHN are shown in Table 2. It shows that our model outperforms all comparison methods. For MNIST→USPS, our method obtains a desirable performance. On the difficult transfer task SVHN→MNIST, our model outperforms existing methods.

Table 1. Classification accuracy (%) on OFFICE-10+Caltech-10 for unsupervised domain adaptation.

Source	A	D	W	A	D	W	A	W	D	C	C	C	Avg
Target	W	W	D	D	A	A	C	C	C	A	W	D	
TCA	84.4	96.9	99.4	82.8	90.4	85.6	81.2	75.5	79.6	92.1	88.1	87.9	87.0
GFK	89.5	97.0	98.1	86.0	89.8	88.5	76.2	77.1	77.9	90.7	78.0	77.1	85.5
AlexNet	79.5	97.7	**100.0**	87.4	87.1	83.8	83.0	73.0	79.0	91.9	83.7	87.1	86.1
DDC [14]	83.1	98.1	**100.0**	88.4	89.0	84.9	83.5	73.4	79.2	91.9	85.4	88.8	87.1
RevGrad [1]	90.8	98.3	98.7	89.2	90.6	93.8	85.7	86.9	83.7	92.8	88.1	87.9	88.9
DAN	91.8	98.5	**100.0**	91.7	90.0	92.1	84.1	81.2	80.3	92.0	90.6	89.3	90.1
RTN [6]	95.2	99.2	**100.0**	**95.5**	93.8	92.5	88.1	86.6	84.6	93.7	94.2	93.4	93.4
MSTN [15]	91.4	99.3	**100.0**	90.2	94.0	**94.2**	88.4	**89.9**	**89.3**	93.7	94.2	**94.9**	93.3
GCAN	**96.1**	**99.4**	**100.0**	93.9	**94.6**	**94.2**	**89.3**	89.5	85.8	**93.8**	**95.9**	94.3	**93.9**

Table 2. Classification accuracy (%) on digit recognitions tasks.

Source	SVHN	MNIST
Target	MNIST	USPS
Source Only	60.1±1.1	75.2±1.6
DDC [14]	68.1±0.3	79.1±0.5
RevGrad [1]	73.9	77.1±1.8
AsmTri [12]	86.0	-
CoGAN [5]	-	91.2±0.8
ADDA [13]	76.0±1.8	89.4±0.2
LEL [7]	81.0±0.3	-
MSTN [15]	91.7±1.5	92.9±1.1
GCAN	**92.0±1.0**	**93.3±0.9**

1.5. Ablation Study

In Table 3 and Table 4, we have shown different variants of our GCAN and the corresponding results on Office-31. Based on the results in Table 4, we can draw the following conclusions: (1) From GCAN-1 to Full GCAN, we demonstrate that all components contribute to the final classification performance. Note that, the structure-aware alignment can increase the performance by 2% on average, and it is quite effective for domain adaptation by modeling data structure information via the structure-aware alignment. (2) Comparing RevGrad and GCAN-2 (MSTN and GCAN-3), we observe that the performance cannot be improved by only expanding feature space. (3) Comparing GCAN-4 and GCAN-3, the structure-aware alignment with triplet loss can help to refine scores features and further slightly improve performance.

1.6. Parameters sensitivity Analysis

The hyper-parameter settings have been illustrated in the main paper. These parameters (except η and α_T) are set by following MSTN. The η and α_T are set according to the results in Figure 2(1) and Figure 2(2). Moreover, we test the robustness of our GCAN under other factors including batch size and GCN output size. Figure 2(3) shows the results of GCAN, MSTN and RevGrad under different batch sizes. The batch size affects all methods, but our GCAN is robust and performs the best in most cases. Figure 2(4) shows the classification accuracy with different GCN output sizes (from 50 to 250), which supports our GCAN is quite robust to the GCN output size.

1.7. Visualization

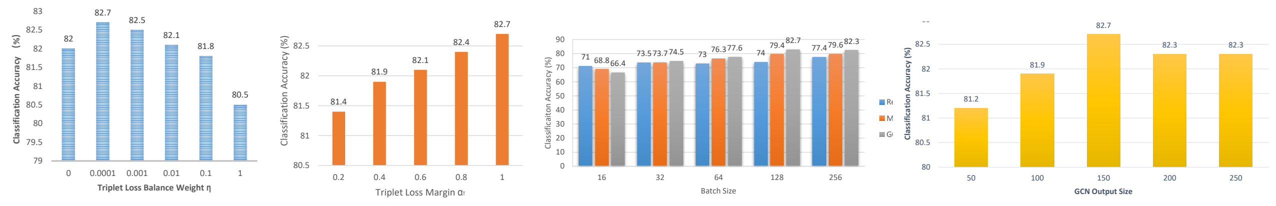
As shown in Figure 3, Figure 4 and Figure 5, we use the t-distributed stochastic neighbor embedding (t-SNE) to visualize the learned representations on Office-31, ImageCLEF-DA and Office-Home, receptively. Results on AlexNet show the

Table 3. Ablation study for different variants of our GCAN. Domain Alignment (DA), Class Alignment (CA), Structure-aware Alignment (SA), Triplet Loss (L_T).

Model	AlexNet (256 dim)	AlexNet (256+150 dim)	GCN (150 dim)	DA	CA	SA without L_T	SA with L_T
RevGrad	✓			✓			
MSTN	✓			✓	✓		
GCAN-1		✓					
GCAN-2		✓		✓			
GCAN-3		✓		✓	✓		
GCAN-4	✓		✓	✓	✓	✓	
Full GCAN	✓		✓	✓	✓		✓

Table 4. Ablation Study: Classification accuracy (%) on the Office-31 dataset.

Method	$A \rightarrow W$	$D \rightarrow W$	$W \rightarrow D$	$A \rightarrow D$	$D \rightarrow A$	$W \rightarrow A$	Avg
RevGrad	73.0±0.5	96.4±0.3	99.2±0.3	72.3±0.3	53.4±0.4	51.2±0.5	74.3
MSTN	80.5±0.4	96.9±0.1	**99.9±0.1**	74.5±0.4	62.5±0.4	60.0±0.6	79.1
GCAN-1	64.2±0.3	94.7±0.2	98.8±0.2	65.5±1.0	48.6±0.5	45.5±0.5	69.6
GCAN-2	75.6±0.4	96.1±0.1	99.4±0.2	70.8±0.4	55.5±0.3	55.1±0.5	75.4
GCAN-3	77.4±0.1	96.0±0.2	99.6±0.2	74.0±0.3	62.6±0.2	61.4±0.3	78.5
GCAN-4	81.1±0.1	96.9±0.1	**99.9±0.1**	**77.9±0.4**	62.9±0.1	**63.6±0.1**	80.4
Full GCAN	**82.7±0.1**	**97.1±0.1**	99.8±0.1	76.4±0.5	**64.9±0.1**	62.6±0.3	**80.6**



(1) Weight η (2) Margin α_T (3) Batch Size (4) GCN Output Size

Figure 2. The effect of several factors on final performance.

representations without any adaptation. As we can see, the distributions are separated between domains. This highlights the importance for transfer learning. As for results on RevGrad [1], features are successfully fused but it also exhibits a serious problem: features generated are near class boundary. For example, features of class A in target domain could be easily mapped to the intermediate space between class A and class B, which is obviously a damage to classification tasks. In contrast, the representations learned by our method are more discriminative and domain-invariant. Specifically, features in the same class are mapped closer. In particular, features with different classes are dispersed, making the features more discriminative. The well-behaved learned features illustrate the effectiveness of jointly leveraging data structure, domain label and class label information in a unified deep network.

2. Network Architectures and Parameters

In this section, we will introduce the network architecture and parameters in our experiments. We have designed different network architectures and parameter settings for different datasets, which can mainly be divided into two types: (1) object recognition, namely, Office-31, ImageCLEF-DA, Office-Home and Office-Caltech; (2) digits recognition, *e.g.*, MNIST-USPS-SVHN. For simplicity, we first define some notations about various layers in deep networks.

Let `conv1` denote a convolutional layer with 96 filters with size 11×11 and stride 4. `conv2` denotes a convolutional layer with 256 filters with size 5×5 and stride 1. `conv3` denotes a convolutional layer with 384 filters with size 3×3 and stride 1. `conv4` denotes a convolutional layer with 384 filters with size 3×3 and stride 1. `conv5` denotes a convolutional layer with 256 filters with size 3×3 and stride 1. `fc-k` denotes a fully-connected with `k`-dimensional output. `pool` denotes max pooling layer with size 3×3 and stride 1. `norm` denotes LRN layer proposed in [3]. `drop` denotes dropout layer. `gcn(A,B)` denotes GCN layer with A-dimensional node representations and B-dimensional output. `conv-m` denotes a convolutional layer with m filters with size 5×5 and stride 1. `relu` and `sigmoid` represent two kinds of activation functions.

Object Recognition Network Architecture.

- CNNs. We use AlexNet architecture and a bottleneck layer with 256 units is added after the fc7 layer in the AlexNet, *i.e.*, `conv1`→`pool`→`norm`→`conv2`→`pool`→`norm`→`conv3`→`conv4`→`conv5`→ `fc-4096`→`drop`→ `fc-4096`→`drop`→`fc-256`→**`CNN features`**.
- Data Structure Analyzer. It is implemented as the AlexNet with a 1000-dimensional output. It is initialized by the weights pretrained on ImageNet and then finetuned on source data with the triplet loss, *i.e.*, `conv1`→`pool`→`norm`→ `conv2`→`pool`→`norm`→`conv3`→`conv4`→`conv5`→`fc-4096`→`drop`→`fc-4096`→`drop`→`fc-1000`→ `relu`→**`Structure Scores`**→**`triplet loss`**.
- GCNs. We first need to construct densely-connected instance graph referring to the main body of the paper. The GCNs architecture is as following, **`Instance Graph`** → `gcn(256,150)` → `relu` → **`GCN features`**.
- Domain Classifier. we use the same architecture as the RevGrad [1], *i.e.*, **`The concatenated CNN and GCN features`**→`fc-1024`→`drop`→`fc-1024`→`drop`→`fc-1`→`sigmoid`→**`domain label`**→**`adversarial loss`**.

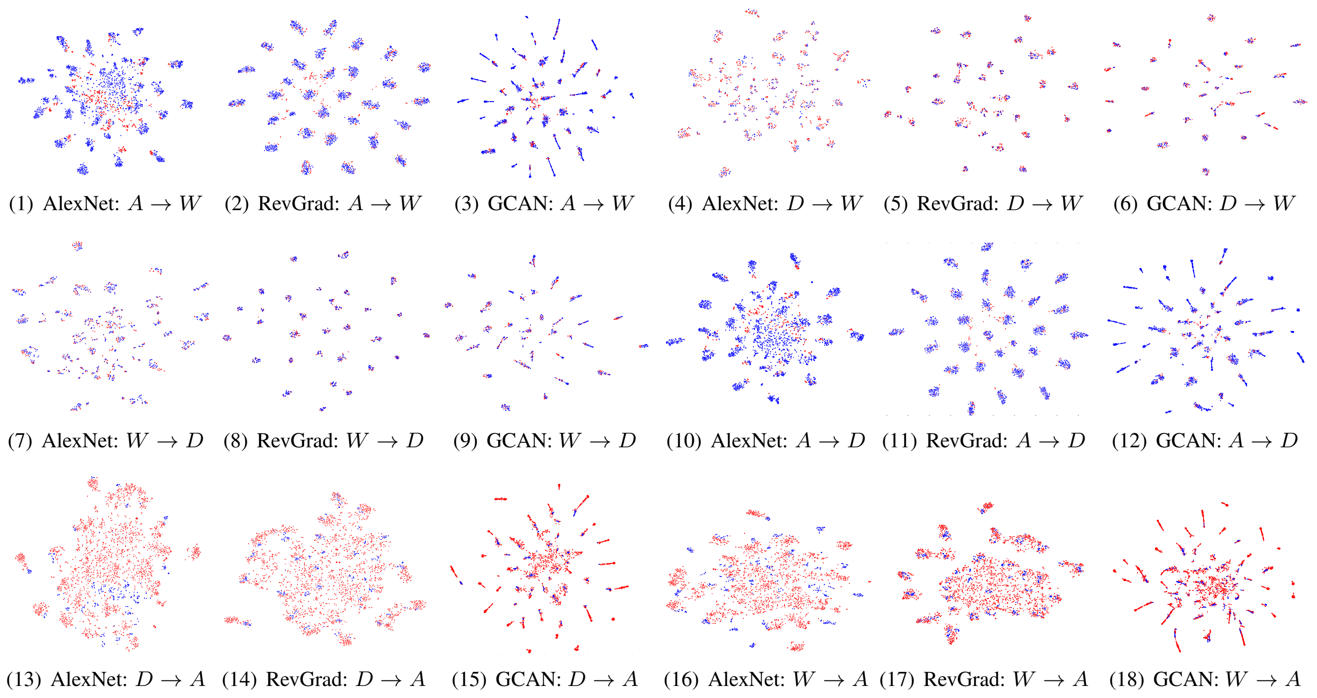
Digits Recognition Network Architecture.

- CNNs. We use LeNet architecture, *i.e.*, `conv-20`→`pool`→`conv-50`→`pool`→`fc-500`→`fc-10`→ **`CNN features`**.
- Data Structure Analyzer. It is implemented as the LeNet. It is initialized by the weights pretrained on source data and then finetuned on source data with the triplet loss, *i.e.*, `conv-20`→`pool`→`conv-50`→`pool`→`fc-500`→`fc-10`→ `relu`→**`Structure Scores`**→**`triplet loss`**.
- GCNs. We first need to construct densely-connected instance graph referring to the main body of the paper. The GCNs architecture is as following, **`Instance Graph`** → `gcn(10,3)` → `relu` → **`GCN features`**.
- Domain Classifier. we use the same architecture as the ADDA [13], *i.e.*, **`The concatenated CNN and GCN features`**→`fc-500`→`relu`→`fc-500`→`relu`→`fc-1`→`sigmoid`→**`domain label`** →**`adversarial loss`**.

In addition, the concatenated CNN and GCN features are applied to **class alignment loss** and final classifier implemented by a fully-connected layer.

Parameters.

- Batch Size: 128
- Base Learning Rate: 0.01
- Moving Average Coefficient θ: 0.7
- Weight Balance Parameters λ, γ, η: The λ and γ are set as $\frac{2}{1+\exp(-k \cdot p)}$, where the k is set to 10, and the p is changing from 0 to 1 within the training process. $\eta = 0.001$.
- Triplet loss threshold α_T: 1
- Domain Adversarial Loss Weight: 0.1
- 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 $\beta = 0.75$ [1]. As for digits recognition, a fixed learning rate 0.01 is used.

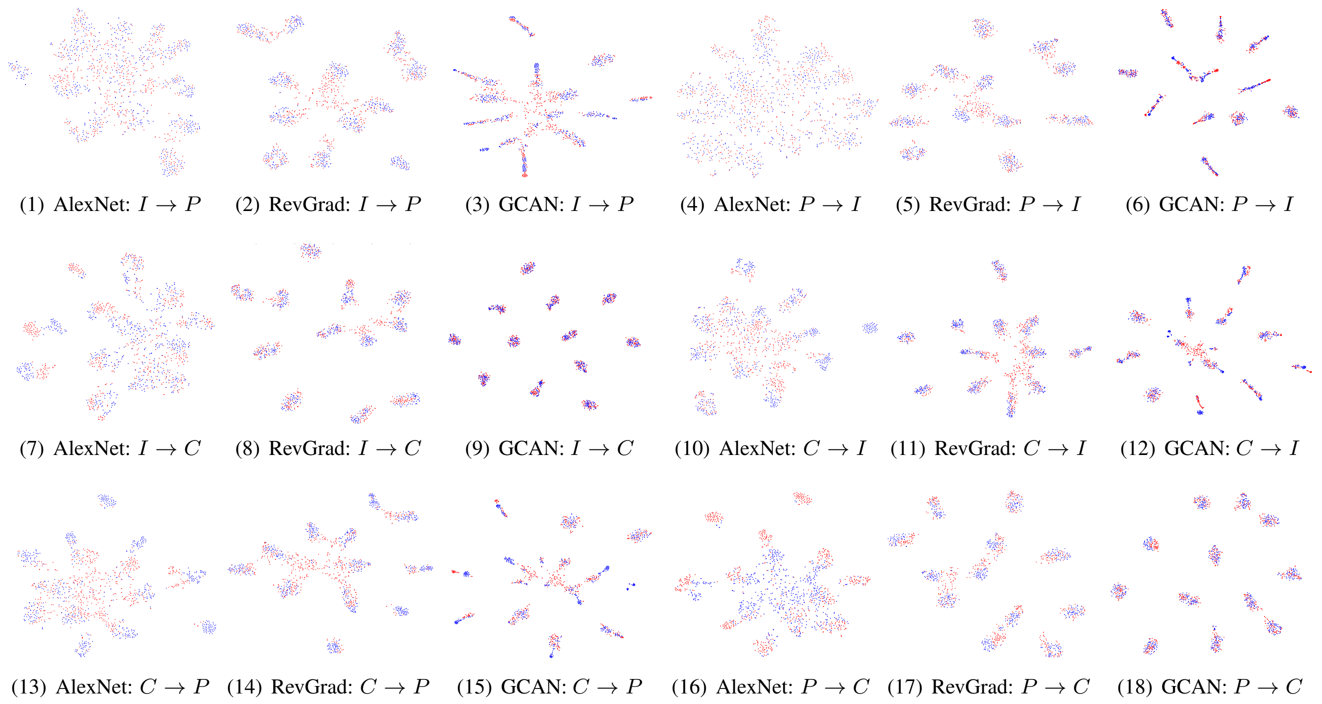


(1) AlexNet: $A \rightarrow W$ (2) RevGrad: $A \rightarrow W$ (3) GCAN: $A \rightarrow W$ (4) AlexNet: $D \rightarrow W$ (5) RevGrad: $D \rightarrow W$ (6) GCAN: $D \rightarrow W$

(7) AlexNet: $W \rightarrow D$ (8) RevGrad: $W \rightarrow D$ (9) GCAN: $W \rightarrow D$ (10) AlexNet: $A \rightarrow D$ (11) RevGrad: $A \rightarrow D$ (12) GCAN: $A \rightarrow D$

(13) AlexNet: $D \rightarrow A$ (14) RevGrad: $D \rightarrow A$ (15) GCAN: $D \rightarrow A$ (16) AlexNet: $W \rightarrow A$ (17) RevGrad: $W \rightarrow A$ (18) GCAN: $W \rightarrow A$

Figure 3. The learned representation visualization on Office-31 using the t-distributed stochastic neighbor embedding (t-SNE) [8]. Blue points are source samples and red ones are target samples.

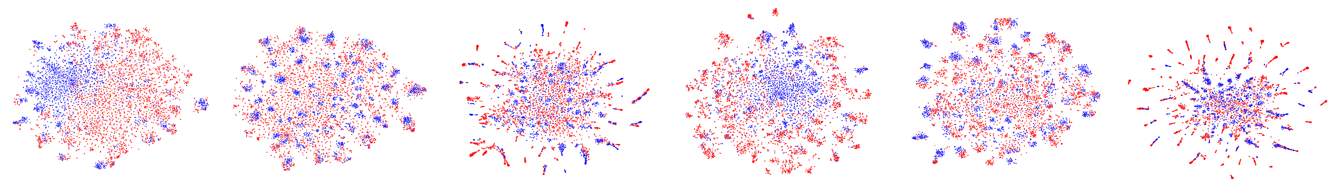


(1) AlexNet: $I \rightarrow P$ (2) RevGrad: $I \rightarrow P$ (3) GCAN: $I \rightarrow P$ (4) AlexNet: $P \rightarrow I$ (5) RevGrad: $P \rightarrow I$ (6) GCAN: $P \rightarrow I$

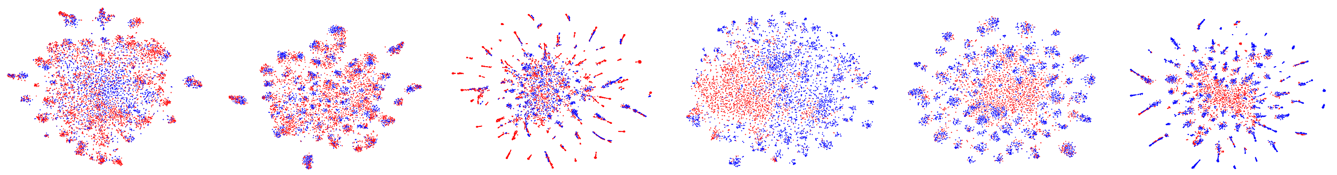
(7) AlexNet: $I \rightarrow C$ (8) RevGrad: $I \rightarrow C$ (9) GCAN: $I \rightarrow C$ (10) AlexNet: $C \rightarrow I$ (11) RevGrad: $C \rightarrow I$ (12) GCAN: $C \rightarrow I$

(13) AlexNet: $C \rightarrow P$ (14) RevGrad: $C \rightarrow P$ (15) GCAN: $C \rightarrow P$ (16) AlexNet: $P \rightarrow C$ (17) RevGrad: $P \rightarrow C$ (18) GCAN: $P \rightarrow C$

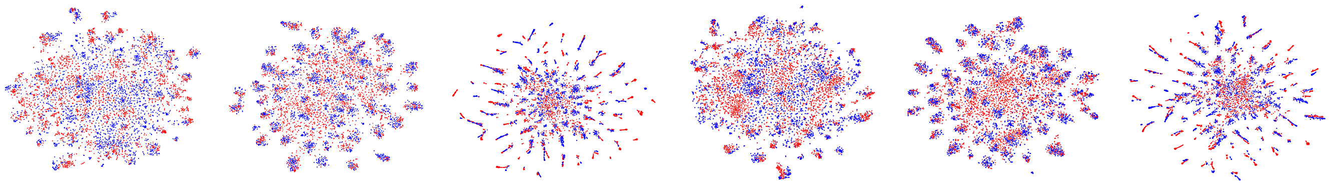
Figure 4. The learned representation visualization on ImageCLEF-DA using the t-distributed stochastic neighbor embedding (t-SNE) [8]. Blue points are source samples and red ones are target samples.



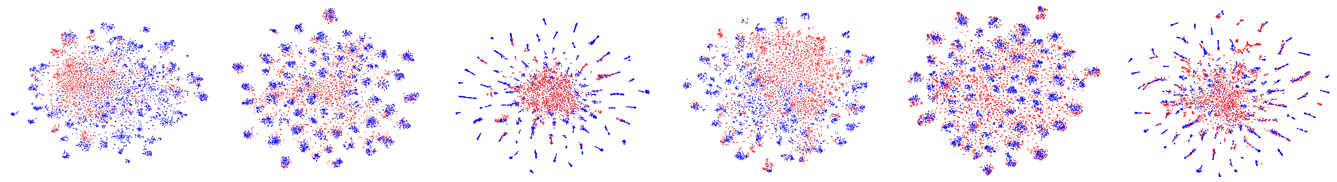
(1) AlexNet:$Ar \rightarrow Cl$ (2) RevGrad:$Ar \rightarrow Cl$ (3) GCAN:$Ar \rightarrow Cl$ (4) AlexNet:$Ar \rightarrow Pr$ (5) RevGrad:$Ar \rightarrow Pr$ (6) GCAN:$Ar \rightarrow Pr$



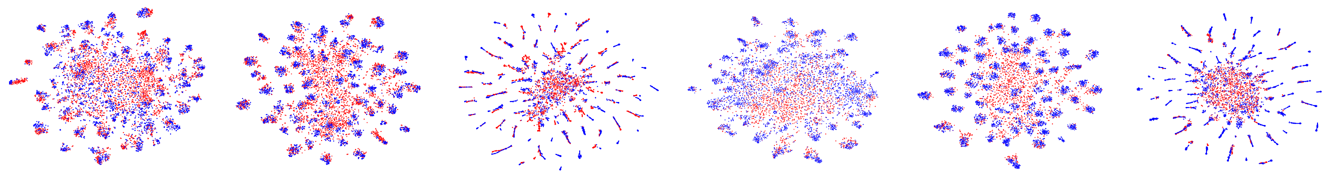
(7) AlexNet:$Ar \rightarrow Rw$ (8) RevGrad:$Ar \rightarrow Rw$ (9) GCAN:$Ar \rightarrow Rw$ (10) AlexNet:$Cl \rightarrow Ar$ (11) RevGrad:$Cl \rightarrow Ar$ (12) GCAN:$Cl \rightarrow Ar$



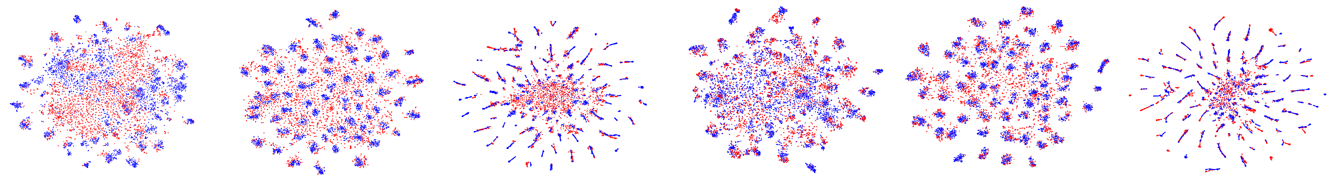
(13) AlexNet:$Cl \rightarrow Pr$ (14) RevGrad:$Cl \rightarrow Pr$ (15) GCAN:$Cl \rightarrow Pr$ (16) AlexNet:$Cl \rightarrow Rw$ (17) RevGrad:$Cl \rightarrow Rw$ (18) GCAN:$Cl \rightarrow Rw$



(19) AlexNet:$Pr \rightarrow Ar$ (20) RevGrad:$Pr \rightarrow Ar$ (21) GCAN:$Pr \rightarrow Ar$ (22) AlexNet:$Pr \rightarrow Cl$ (23) RevGrad:$Pr \rightarrow Cl$ (24) GCAN:$Pr \rightarrow Cl$



(25) AlexNet:$Pr \rightarrow Rw$ (26) RevGrad:$Pr \rightarrow Rw$ (27) RevGrad:$Pr \rightarrow Rw$ (28) AlexNet:$Rw \rightarrow Ar$ (29) RevGrad:$Rw \rightarrow Ar$ (30) GCAN:$Rw \rightarrow Ar$



(31) AlexNet:$Rw \rightarrow Cl$ (32) RevGrad:$Rw \rightarrow Cl$ (33) GCAN:$Rw \rightarrow Cl$ (34) AlexNet:$Rw \rightarrow Pr$ (35) RevGrad:$Rw \rightarrow Pr$ (36) GCAN:$Rw \rightarrow Pr$

Figure 5. The learned representation visualization on Office-Home using the t-distributed stochastic neighbor embedding (t-SNE) [8]. Blue points are source samples and red ones are target samples.

References

[1] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189, 2015.

[2] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073. IEEE, 2012.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NIPS*, pages 469–477, 2016.

[6] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NIPS*, pages 136–144, 2016.

[7] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li F Fei-Fei. Label efficient learning of transferable representations acrosss domains and tasks. In *NIPS*, pages 165–177, 2017.

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[9] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[10] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724, 2014.

[11] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[12] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *ICML*, pages 2988–2997, 2017.

[13] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176, 2017.

[14] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[15] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *ICML*, pages 5419–5428, 2018.