Aggregating Randomized Clustering-Promoting Invariant Projections for Domain Adaptation: Supplementary Materials

Jian Liang, Ran He, *Senior Member, IEEE,* Zhenan Sun, *Member, IEEE,* and Tieniu Tan, *Fellow, IEEE*
✉ {jian.liang,rhe,znsun,tnt}@nlpr.ia.ac.cn

Abstract—In this appendix, we mainly provide some additional experiments to verify the effectiveness in some severe cases and the generalization ability across different categories. Specifically, we perform a comparative test to study the robustness to non-uniform source label distribution. Then, we are interested in the case where the numbers of classes are not equal for source and target domains, namely, some target classes are dropped for such asymmetric transfer in this experiment. Finally, we also investigate the generalization capability of the learned projection, i.e., whether the optimal domain-invariant projection learned on classes 'a,b,c' works well for calculating the similarities within classes 'd,e,f'. Experimental results demonstrate the effectiveness of proposed methods, further, the ensemble one is much more robust to kinds of severe settings. Moreover, we also include a large-scale cross-domain digit dataset SVHN-MNIST to validate the efficiency and effectiveness of proposed 'sapling-and-fusion' strategy. Some extra discussions are also included to verify the convergence and robustness to different kinds of classifiers.

1 Uniform versus Non-uniform Source Label Distribution

The label distributions of two original digit datasets, i.e., MNIST and USPS, are shown in Fig. 1. Notably, the label distribution of MNIST is rather close to the uniform distribution, while USPS owns a contrarily non-uniform label distribution. To investigate the robustness of our ensemble strategy to different source label distributions, we introduce two additional uniform sources MNIST_un and USPS_un (abbreviated as Mu and Uu) for comparison. Concretely, each class in MNIST_un and USPS_un contains 170 and 120 digit images, respectively. To be fair, we also obtain two rescaled datasets, Mo and Uo from MNIST and USPS via uniform sampling, which own the same number of samples as Mu and Uu, and we run each method 10 times (picking samples are random) with different random seeds.

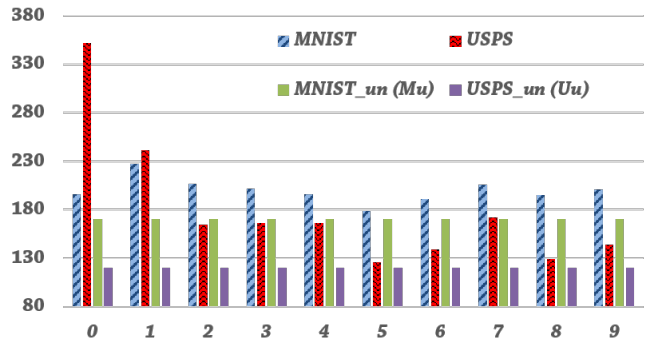


Fig. 1. The number of images in 10 digit classes on the MNIST, USPS, MNIST_un and USPS_un datasets.

We show the comparison results of our basic and ensemble methods DICE and WMV-1-NN with JDA [1] and JGSA [2] in Table 1, with the subspace dimensionalities of them being 10 and 30 in M→ U and U→M, respectively. For our ensemble method, the sampling densities $\delta_s, \delta_t, \delta_f$ are respectively fixed as 0.8, 0.6 and 0.9 with K being 10 which are suggested before. Then we run it 10 times and report the average and deviation values. As can be seen from Table 1, WMV-1-NN is the best performing method, and DICE always outperforms JGSA and JDA except for Uo→M. Notably, when the source label distribution becomes uniform, the recognition accuracies even decrease a little bit for JDA and JGSA while DICE and WMV-1-NN grows in contrast. All these methods are deemed to be robust to the change of source label distribution.

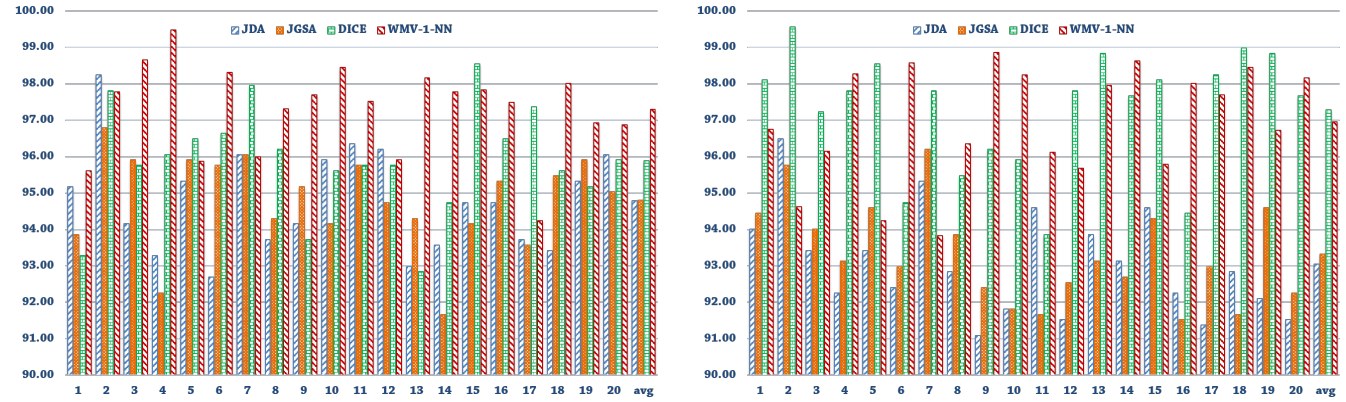
TABLE 1
Averaged recognition accuracies (%) comparisons on the **MNIST-USPS** dataset (u/o : uniform/ original non-uniform source label distribution).

Method	1-NN	JDA	JGSA	DICE	WMV-1-NN
Mo→U	63.77	70.52	76.20	76.82	**79.17** ± 0.44
Mu→U	65.13	70.79	75.08	75.98	**80.30** ± 0.98
Uo→M	43.73	59.61	**66.84**	62.87	66.49 ± 0.29
Uu→M	43.18	59.93	62.56	63.63	**65.69** ± 0.40

TABLE 2
Averaged recognition accuracies (%) comparisons on the **COIL20** dataset (Old: results without random sampling on source domain).

Method	1-NN	JDA	JGSA	DICE	WMV-1-NN
C1r→C2	82.92	94.63	94.35	96.22	**97.47** ± 0.89
C2r→C1	81.11	92.96	93.13	**98.85**	98.28 ± 0.67
Avg.	82.02	93.79	93.74	97.53	**97.88** ± 0.45
Old Avg.	83.20	94.20	94.66	**99.72**	99.28 ± 0.11

Besides the MNIST-USPS dataset, we also perform random sampling (sampling density is fixed as 0.7) on the source domain of COIL20 to disturb the original uniform label distribution. Concretely, for all methods, the subspace dimensionality is fixed as 10, and merely the sampling density δ_f of our ensemble method is 0.5 with K being



(a) C1 → C2 (b) C2 → C1

Fig. 2. Recognition accuracies (%) w.r.t. the index of one missing target category on the **COIL20** dataset.

10. The comparison results depicted in Table 2 illustrate that WMV-1-NN achieves the best results and DICE is also significantly superior to JDA and JGSA. Once compared with previous results, WMV-1-NN is more robust than DICE with a smaller decrease in terms of the mean accuracy.

2 Asymmetric Labels across Domains (Missing Target Categories)

In this section, we are interested whether missing some target categories would affect the adaptation performance. Intuitively, the asymmetric labels across domains may degenerate unsupervised DA methods, especially for conditional distribution matching methods where an identical label set is always assumed in advance.

In each trail, we first drop one category of the target domain on the COIL20 dataset to evaluate the effectiveness of asymmetric DA tasks, the comparison results are shown in Fig. 2. On both two tasks, DICE and WMV-1-NN significantly outperform JDA and JGSA, especially, JDA wins DICE and WMV-1-NN only when the 2-nd or 11-th category is dropped for C1 → C2. Concerning the mean accuracy, DICE outperforms JGSA by 1.14 and 4.24 (%) respectively. WMV-1-NN (99.31) is slightly inferior to DICE (99.72) in the full target domains situations, however, WMV-1-NN re-wins DICE under the asymmetric labels across domains situation. It indicates that the ensemble strategy is somewhat robust to the asymmetric label distribution.

Furthermore, we drop 5 out of 20 categories of the target domain with different random seeds for 5 times, and the results are shown in Table 3. Once again, DICE is superior to JDA and JGSA for almost 5 cases, and WMV-1-NN consistently outperforms DICE in both tasks. The overall accuracies are smaller than those in Fig. 2 for almost all methods except WMV-1-NN, which verifies our assumption that the more missing target categories, the more inefficiencies in learning domain-invariant projections. Looking the change of mean accuracies more carefully, the largest decreases occur at JGSA for both C1 → C2 (94.80 → 90.74) and C2 → C1 (93.33 → 90.89), which indicates that JGSA is sensitive to the missing target labels. However, our basic method DICE and ensemble method WMV-1-NN are really stable even the number of missing target categories increases, which indicates that the proposed objective is rather robust to such asymmetric labels.

TABLE 3
Recognition accuracies (%) w.r.t. the indexes of five missing target categories on the **COIL20** dataset.

Task	C1 → C2				C2 → C1			
Missing indexes	JDA	JGSA	DICE	WMV-1-NN	JDA	JGSA	DICE	WMV-1-NN
[2, 8, 9, 10, 11]	96.48	92.78	**98.15**	95.56 ± 0.40	90.74	88.89	90.56	**98.36** ± 0.56
[1, 4, 15, 18, 19]	89.07	87.78	94.07	**96.92** ± 0.78	92.04	92.96	94.81	**98.25** ± 0.40
[2, 6, 7, 8, 11]	95.74	93.33	95.19	**97.19** ± 0.90	94.44	94.07	97.59	**98.14** ± 0.40
[2, 8, 9, 15, 17]	93.52	92.22	94.26	**96.83** ± 0.74	92.04	93.52	95.37	**98.64** ± 0.32
[4, 7, 10, 12, 16]	89.26	87.59	95.93	**96.53** ± 0.72	91.85	85.00	96.30	**98.44** ± 0.36
Average	92.81	90.74	95.44	**96.61** ± 0.88	92.22	90.89	94.93	**98.37** ± 0.42

Besides the COIL20 dataset, we also attempt similar trials on the uniform PIE dataset, since there are 68 classes for each domain, we decide to drop 10, 20 and 30 classes within 5 times to again verify the robustness of domain adaptation methods. Concerning the parameter setting, the subspace dimensionalities of all methods are fixed as 100, the sampling densities for our method WMV-1-NN are 0.8, 0.6, and 0.9, respectively.

The results in Table 4 tell us that WMV-1NN always performs better than DICE, and both methods consistently outperform JDA CDDA [3] and JGSA significantly. As more categories are gradually dropped, all the methods undergo a decrease at different degrees. Regarding the falling rate, we find that JGSA and WMV-1-NN perform the best except 1-NN, however, the results of both 1-NN and JGSA are much lower than WMV-1-NN.

3 Generalization Capability of Learned Invariant Projection

In this section, we discuss one interesting and similar problem to zero-shot learning, i.e., cross-class learning, where data belonging to one-half classes is utilized for training and the remaining half classes of data are prepared for the testing phase, hence the training and testing data have none overlapping classes. In this sense, we wonder the optimal learned projection in the seen training classes can distinguish those unseen classes, specifically, the PIE dataset is again exploited where the images of the first 34 persons constitute the training set with the images of the remaining 34 persons being the testing set. It is worthwhile to note that during the testing phase, only labeled unseen source classes are utilized to infer the labels of unseen target classes.

TABLE 4
Averaged recognition accuracies (%) on the **PIE** dataset with {10,20,30} random target classes missing (l_2-normalization).

setting	10 target classes missing						20 target classes missing						30 target classes missing					
data	1-NN	JDA	CDDA	JGSA	DICE	WMV-1NN	1-NN	JDA	CDDA	JGSA	DICE	WMV-1NN	1-NN	JDA	CDDA	JGSA	DICE	WMV-1NN
P1→P2	26.25	69.68	70.87	60.74	75.23	**80.20**	25.46	66.97	66.22	57.79	73.95	**78.59**	24.85	62.31	60.33	55.54	69.00	**71.93**
P1→P3	26.92	67.99	70.26	59.10	76.38	**81.25**	25.26	62.79	62.90	56.31	71.75	**72.48**	24.62	59.66	58.15	53.17	67.28	**67.54**
P1→P4	31.11	89.93	89.13	80.71	93.24	**94.89**	29.54	84.87	84.63	77.54	89.27	**89.95**	29.29	81.48	79.75	73.53	85.55	**87.70**
P1→P5	16.93	54.44	59.70	45.58	64.62	**68.18**	16.22	50.56	54.23	42.27	60.06	**64.06**	16.06	51.51	51.12	40.20	59.50	**65.13**
P2→P1	24.19	71.45	71.00	68.17	76.68	**78.92**	23.79	68.95	69.66	67.21	**74.09**	73.94	23.50	64.65	65.57	64.93	70.63	**71.05**
P2→P3	45.92	71.69	72.40	63.51	70.88	**74.78**	45.73	69.57	69.30	60.78	69.80	**71.27**	44.31	66.90	64.96	57.59	66.13	**67.00**
P2→P4	53.92	81.73	82.21	77.37	86.34	**86.52**	53.83	81.34	80.27	77.06	83.89	**85.87**	53.44	78.28	76.19	74.55	81.09	**86.84**
P2→P5	26.62	59.35	59.58	51.62	64.51	**66.38**	25.49	58.04	57.55	50.76	61.86	**67.36**	25.49	54.27	52.36	47.87	**58.20**	56.14
P3→P1	21.96	67.25	73.92	62.48	76.60	**78.64**	19.24	60.27	65.78	57.74	67.89	**76.57**	18.87	57.27	59.28	55.44	64.72	**68.53**
P3→P2	41.02	72.63	69.68	59.90	74.13	**79.12**	38.82	68.50	64.87	55.12	69.87	**82.81**	37.93	64.91	61.81	53.28	65.72	**75.25**
P3→P4	46.99	81.11	80.34	71.01	85.71	**89.96**	43.86	76.04	75.13	66.91	83.20	**89.31**	43.56	74.90	71.76	65.15	**80.57**	78.00
P3→P5	26.82	61.40	63.85	51.88	**69.72**	68.18	24.96	57.14	62.40	48.88	67.48	**68.06**	24.64	54.32	58.09	47.27	64.69	**66.56**
P4→P1	33.45	88.35	89.69	86.25	92.68	**93.81**	30.71	84.80	85.59	84.05	89.84	**91.24**	30.23	80.66	81.54	80.64	**86.66**	85.82
P4→P2	63.12	88.82	90.12	82.95	**92.87**	91.58	60.80	86.05	86.78	80.15	90.75	**91.64**	59.90	84.05	84.01	77.63	**89.15**	88.71
P4→P3	73.38	88.69	90.21	82.14	91.17	**91.24**	71.28	86.98	88.31	80.95	89.15	**89.50**	71.34	85.15	86.24	79.11	88.02	**89.58**
P4→P5	37.74	67.39	73.05	64.73	78.03	**79.17**	36.42	66.35	67.90	62.38	76.09	**83.25**	36.25	61.98	62.80	58.66	73.37	**79.50**
P5→P1	18.04	59.48	67.00	52.23	71.88	**77.66**	16.89	55.35	62.30	50.18	70.28	**70.71**	16.87	53.15	58.96	48.50	66.04	**78.84**
P5→P2	23.98	64.02	62.76	56.22	66.97	**68.90**	21.91	58.80	56.87	51.87	62.94	**64.50**	21.57	56.92	56.15	49.38	59.12	**71.62**
P5→P3	28.17	69.07	70.58	56.08	74.30	**75.72**	26.88	65.41	65.01	53.31	70.75	**71.27**	26.47	60.84	62.41	49.61	**68.53**	64.80
P5→P4	31.49	72.17	75.63	62.89	78.81	**78.96**	29.28	66.43	69.90	57.06	**74.65**	71.39	29.41	62.79	64.54	55.42	70.77	**77.51**
Avg.	34.90	72.33	74.10	64.78	78.04	**80.20**	33.32	68.76	69.78	61.92	74.88	**77.69**	32.93	65.80	65.80	59.37	71.74	**74.90**

TABLE 5
Recognition accuracies (%) on the latter 34 classes of **PIE** dataset with learned projection with the former 34 classes (l_2-normalization).

data	1-NN	JDA	CDDA	JGSA	DICE	Conv-1NN	MV-1NN	WMV-1NN
P1→P2	38.25	65.07	68.27	62.62	**74.17**	73.60 ± 1.32	**74.64** ± 0.59	72.87 ± 0.47
P1→P3	35.42	60.66	58.70	55.15	**66.67**	72.08 ± 0.71	**72.82** ± 1.40	72.08 ± 1.44
P1→P4	41.79	82.92	87.13	74.79	**88.94**	90.99 ± 0.62	**91.43** ± 0.68	90.78 ± 0.57
P1→P5	27.21	54.90	63.48	49.14	**64.95**	72.92 ± 0.69	**74.02** ± 0.87	73.48 ± 0.67
P2→P1	29.17	65.37	66.09	68.79	**73.53**	77.44 ± 1.58	77.21 ± 1.18	**77.50** ± 1.36
P2→P3	50.49	69.36	64.83	**69.61**	65.93	70.51 ± 1.56	69.88 ± 0.71	**70.86** ± 0.77
P2→P4	66.69	79.56	80.76	76.11	**82.44**	**85.69** ± 0.61	85.62 ± 0.61	84.59 ± 0.47
P2→P5	39.09	53.06	**58.95**	49.75	58.82	67.89 ± 1.03	**68.26** ± 1.11	67.62 ± 0.84
P3→P1	30.61	55.10	57.26	59.12	**64.53**	71.20 ± 0.71	71.37 ± 0.51	**71.70** ± 0.51
P3→P2	49.82	62.61	54.86	**63.48**	60.02	68.95 ± 0.44	68.59 ± 1.62	**69.25** ± 0.96
P3→P4	56.16	71.32	74.80	69.45	**77.81**	80.04 ± 0.44	79.76 ± 0.60	**80.48** ± 0.62
P3→P5	36.03	45.71	56.00	47.55	**57.60**	65.71 ± 0.69	**65.83** ± 0.64	65.74 ± 0.54
P4→P1	44.06	80.61	87.27	78.33	**88.90**	89.80 ± 0.28	**90.10** ± 0.50	90.08 ± 0.38
P4→P2	70.73	84.87	88.19	87.38	**90.04**	90.80 ± 1.15	91.29 ± 0.43	**91.44** ± 0.22
P4→P3	80.51	87.99	90.81	84.56	**91.42**	91.81 ± 0.82	92.11 ± 0.43	**92.55** ± 0.45
P4→P5	46.94	65.93	73.65	65.07	**76.47**	80.88 ± 0.91	80.51 ± 0.77	**81.15** ± 0.46
P5→P1	26.59	53.36	58.04	45.02	**64.35**	67.82 ± 1.11	**68.60** ± 0.95	68.07 ± 0.87
P5→P2	35.18	55.60	**60.76**	41.91	60.27	65.12 ± 0.96	65.07 ± 1.17	**65.51** ± 1.44
P5→P3	37.87	46.57	60.29	54.41	**63.60**	69.63 ± 1.35	69.29 ± 0.96	**70.44** ± 1.38
P5→P4	43.30	60.67	70.17	59.72	**71.92**	75.57 ± 0.65	75.90 ± 0.50	**76.21** ± 0.29
Avg.	44.29	65.06	69.02	63.10	**72.12**	76.42 ± 0.11	76.61 ± 0.09	**76.62** ± 0.08

All the comparison results in terms of "unseen versus unseen" are depicted in Table 5. Observing the accuracies from Table 5, we discover several facts as below, firstly, DICE significantly beats JDA, CDDA and JGSA in 16 out of 20 tasks; secondly, all the fusion methods achieve similar and better performances and DICE is obviously inferior to WMV-1-NN by about 6.23%; finally, the recognition accuracy is rather acceptable even it is a bit lower than fully transductive setting.

TABLE 6
Recognition accuracies (%) on the cross-class **COIL20** dataset.

Task	Protocol	JDA	JGSA	DICE	DICE_f
C1 → C2	seen - seen	91.67	**94.02**	93.61	99.44
	unseen - unseen	98.61	97.22	**98.33**	100.0
	unseen - all	**97.78**	95.56	**97.78**	100.0
C2 → C1	seen - seen	**88.61**	86.94	87.50	99.44
	unseen - unseen	**99.44**	96.39	98.89	100.0
	unseen - all	96.39	94.17	**97.78**	100.0

Besides the "unseen versus unseen" protocol on the PIE dataset, we also exploit the COIL20 dataset for more analysis, concretely, we include the accuracies when both seen and unseen source classes are utilized in the testing phase ("unseen versus all") in addition to the training accuracies ("seen versus seen"). The seen classes are the first 10 classes while the unseen classes are the latter 10 classes, and DICE_f utilizes both seen and unseen classes for inferring the projection. As can be seen from the results in Table 6, JGSA and JDA obtain the highest training accuracies ("seen-seen") for two tasks respectively. While focusing on the testing accuracies, DICE achieves the best performance except under the "unseen-unseen" protocol, however, it maintains a higher accuracy when the classes of labeled data increase. It indicates that JDA is a little bit overfitting due to its higher training accuracy and lower testing accuracy. Compared with JGSA, DICE is superior for both training and testing accuracies. Nevertheless, the performance of DICE_f is rather promising since more labeled source classes are available for training. Generally speaking, DICE is the best performing method even for the challenging cross-class face and object recognition problems.

4 Domain Adaptation on SVHN→MNIST

The effectiveness of 'sampling-and-fusion' has been empirically witnessed among many comparison results in the main text. Hence, we further resort one favorable benchmark large-scale dataset SVHN-MNIST to study the computation efficiency.

Street View House Numbers (SVHN) dataset [4] is a collection of house numbers collected directly from Google street view images, while MNIST [5] contains massive clear handwritten digits. We follow the standard protocol that utilizes all the training images for unsupervised domain adaptation, namely, the dataset sizes of SVHN and MNIST are 73,257 (32×32×3, RGB) and 60,000 (28×28, gray-scale), respectively. SVHN images were gray scaled and rescaled to 28 × 28 in our paper, the simple LeNet architecture is shown below. Then we train the network with all labeled source

C : convolution, **P** : max-pooling, **R** : ReLU, **F** : fully connected

data, SVHN images in this task, and use the activations of the last feature layer pool2 as image representations for both SVHN and MNIST.

We reorganize and duplicate the results from several recent works, including DANN [6], kNN-Ad [7] and ADDA [8]. For different works, the baseline network 'Source Only' (fine-tuning the pre-trained network with labeled source examples) behave differently. Hence, we also provide the accuracy of baseline network and calculate the improvement (gain) in Table 7 for fair comparison. For our fusion method, the sampling densities of δ_s, δ_t are fixed to 0.1 or 0.05 with K being 10 and δ_f being 1, we run them 10 times and report the averaged accuracies.

The baseline network of our method merely performs better than that of kNN-Ad [6] while losing to others. Since we focus on the improvement of our method over the baseline network, we do not spend much time in fine-tuning the network. Even though, our shallow model MV-SVM still outperforms several deep domain adaptation methods including DANN, kNN-Ad, JAN and ADDA. The accuracy of MV-1-NN is slightly lower than that of MV-SVM, while the accuracy of Conv-1-NN is definitely higher than that of Conv-SVM, which indicates that 1-NN is somewhat robust for different fusion strategies.

Observing Table 7, WV-SVM obtains the best accuracy and WMV-1-NN obtains competitive performance with kNN-Ad. Even compared with semi-supervised domain adaptation method FADA, our methods still outperform it with relatively large margins. In terms of the improvement over baseline 'Source-Only', kNN-Ad obtains the second largest accuracy gain (43.5%), and our fusion method MV-SVM achieves the best performance (44.9%). Besides, WMV-1-NN achieves a promising accuracy gain (40.8%) which ranks the 3-rd highest among all methods. Generally, our methods outperform state-of-the-art deep domain adaptation methods, given a better baseline network, our methods are expected to achieve much higher cross-domain recognition accuracy. Concerning the computation time that is measured on a PC workstation with an Intel CPU (2.8 GHZ) and 88 GB of RAM, we find that each sampled domain adaptation task takes about 7.5 minutes ($\delta_{s,t} = [0.1, 0.1]$), which is quite promising for large-scale datasets. When sampling densities δ_s and δ_f decrease, the accuracy always tends to decrease within shorter computation time.

TABLE 7
Recognition accuracies on the **SVHN-MNIST** dataset.

Method	Accuracy (%)	Gain (↑)
Source Only [7]	54.9	–
SA* [9]	59.3	8.0%
DANN [6]	73.9	34.5%
kNN-Ad [7]	**78.8**	**43.5%**
Source Only [8]	60.1	–
Domain confusion [10]	68.1	13.3%
ADDA [8]	**76.0**	**26.5%**
Source + Target † [11]	65.5	–
FADA † [11]	**72.8**	**11.1%**
Source Only	56.8	–
1-NN	55.4	–
Conv-1-NN	74.8	35.0%
MV-1-NN	76.1	37.4%
WMV-1-NN	**78.0**	**40.8%**
SVM	55.8	–
Conv-SVM	69.0	23.6%
MV-SVM	**80.9**	**44.9%**
$K = 10$, $\delta_f = 1$, $\delta_{s,t} =$ **[0.1, 0.1]** ⇑ and $\delta_{s,t} =$ **[0.05, 0.05]** ⇓.		
Conv-1NN	73.4	32.5%
MV-1-NN	75.4	36.0%
WMV-1-NN	76.9	**38.7%**
Conv-SVM	66.4	19.9%
MV-SVM	**79.4**	**43.2%**

† Semi-supervised DA method utilizes one labeled target sample per category.

5 Details of Label Propagation Extension

As aforementioned before, we replace 1-NN with such a label propagation (LP) [12] extension in the pseudo label inference step to fairly compare with DGA-DA [13]. Particularly, once the domain-invariant projection A is learned, we can easily obtain the pseudo target labels via 1-NN, then we can construct an affinity matrix W for all instances from source and target domains, whose element is defined by, $w_{i,j} = exp(-\sigma\|A^T x_i - A^T x_j\|)$ if $i \neq j$ and $w_{ii} = 0$. Afterwards, we further construct a matrix $S = D^{-1/2} W D^{-1/2}$ in which D is a diagonal matrix with its (i,i)-element equal to the sum of the i-th row of W. In the following, we easily obtain a more reliable soft pseudo target label matrix $F \in \mathbb{R}^{(n_s+n_t)\times C}$ via local and global consistency,

$$F = (I - \alpha S)^{-1}[Ys; \hat{Y}_t]. \quad (1)$$

Then we discover the optimal class corresponding to the largest soft probability for pseudo target labeling. Here Ys and $\hat{Y}_t$ respectively denote the one-hot encodings of semantic source labels and pseudo target labels. In this paper, $\alpha = 0.8$ and $\sigma = 10$ are fixed for all datasets in the experiments.

6 Algorithm Analysis

6.1 Sensitivity to Subspace Dimensionality

To investigate the sensitivity to subspace dimensionality, we illustrate how the average accuracy changes with different values for m in Fig. 3 (m varies in the range of $[8, 9, \cdots, 20]$ for Office-Caltech and COIL20, and $[80, 90, \cdots, 200]$ for PIE). It is worthwhile to mention this parameter sensitivity experiment is carried out on the Office-Caltech ($DeCAF_6$ 'full-training'), PIE (l_2 normalization), and COIL20 datasets. Apparently, the average recognition accuracies for all three datasets first increase and then gradually decrease. The optimal m values are different, however, around them the accuracies are quite stable, which indicates that our method is robust to the subspace dimensionality parameter m.

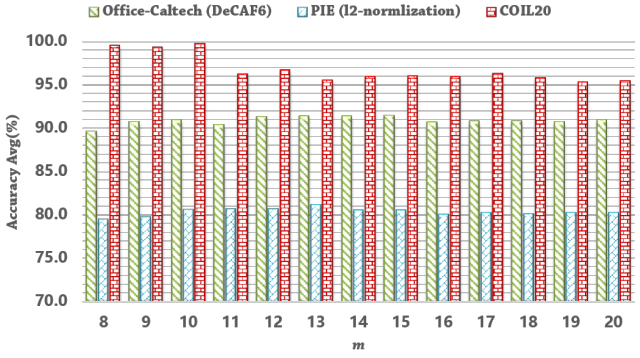


Fig. 3. Averaged recognition accuracy w.r.t. subspace dimensionality m ($m \leftarrow m \times 10$ for PIE).

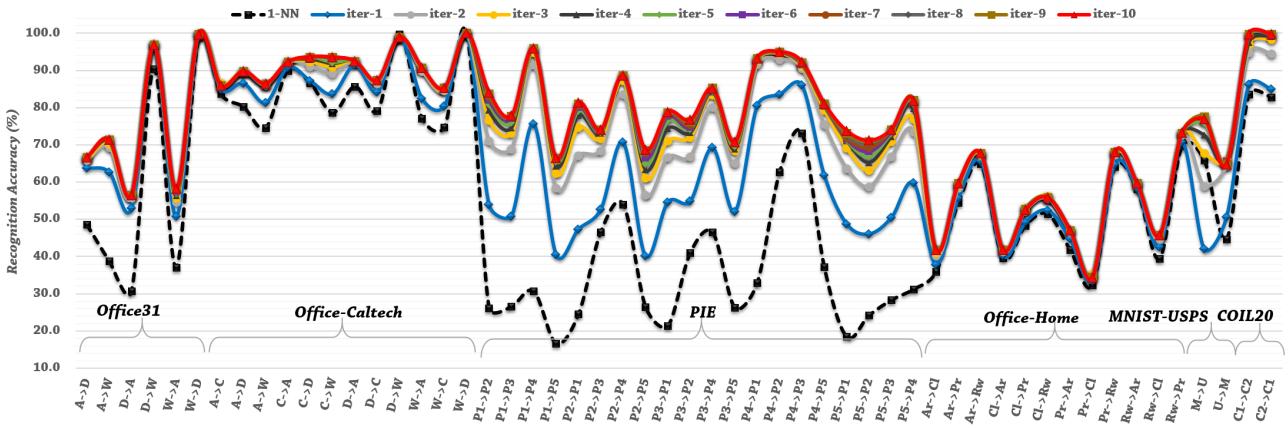


Fig. 4. Convergence study of our algorithm: accuracy with iterations. [**Office-Caltech**: DeCAF$_6$ & **PIE**: l_2-normalization & **Office-Home**: ResNet50-P$_5$]

TABLE 8
Recognition accuracies (%) on **Office-Caltech** (DeCAF$_6$) of Algorithm 1. Values in red and bold face indicate the best results (default parameters: λ = 1, γ = 0.1, m = 15, maximum iteration T = 10).

pse-class	*fin-class*	A→C	A→D	A→W	C→A	C→D	C→W	D→A	D→C	D→W	W→A	W→C	W→D	Avg.
1-NN		85.9	89.8	86.4	92.3	**93.6**	**93.6**	**92.5**	87.4	**99.0**	90.7	85.3	**100.0**	91.4
SVM	1-NN	**87.6**	**95.5**	86.1	**92.9**	92.4	86.8	92.1	**88.8**	**99.0**	**92.1**	**87.5**	99.4	**91.7**
LP		86.6	89.8	**87.8**	92.6	89.8	91.9	**92.5**	87.8	**99.0**	90.7	85.9	**100.0**	91.2
1-NN		87.6	91.1	88.1	93.4	**95.5**	**95.3**	**92.5**	88.5	**99.0**	91.1	88.0	**100.0**	**92.5**
SVM	SVM	88.0	**95.5**	84.7	**93.5**	91.7	84.7	91.9	**89.0**	**99.0**	**91.5**	86.7	96.8	91.1
LP		**88.1**	91.1	**89.2**	**93.5**	91.1	93.9	**92.5**	88.5	**99.0**	91.0	**88.1**	**100.0**	92.2
1-NN		86.4	89.8	87.1	92.6	**94.9**	**94.2**	92.6	87.4	**99.0**	90.9	85.8	**100.0**	**91.7**
SVM	LP	**88.0**	**95.5**	84.7	**93.6**	92.4	84.7	91.9	**89.0**	**99.0**	**91.5**	**86.7**	96.8	91.2
LP		86.9	89.8	**88.8**	92.9	91.1	93.2	**92.7**	87.6	**99.0**	90.9	86.0	**100.0**	91.6

6.2 Convergence Analysis

Besides the parameter sensitivity study, we discuss the convergence of our method in Algorithm 1 on all 6 datasets. As shown in Fig. 4, 1-NN is the initial state which can be considered as iter-0, for a majority of tasks, the accuracy achieves the highest and stop growing when the number of iterations arrives at 10. Obviously, we can draw the following conclusion: the accuracy of our method grows larger gradually and tends to be stable with the increase of the number of iterations.

Besides the quantitative analysis, we also perform qualitative analysis to find the visual changes in the distributions of different subspaces for each iteration. Specifically, we exploit the most popular visualization tool t-SNE [1] to show the visualizations of embedding spaces of one representative task C → W with DeCAF$_6$ features in Fig. 5. Obviously, classes 1, 3, 5 and 10 are not well separated in iteration # 1, while classes 3, 5 and 10 are not clearly separated in iteration # 2. After iteration # 5, only classes 3 and 5 are relatively mixed up. However, other 9 classes are very well separated in iteration # 8, 9 and 10.

7 Discussions on Standard Classifiers of Algorithm 1

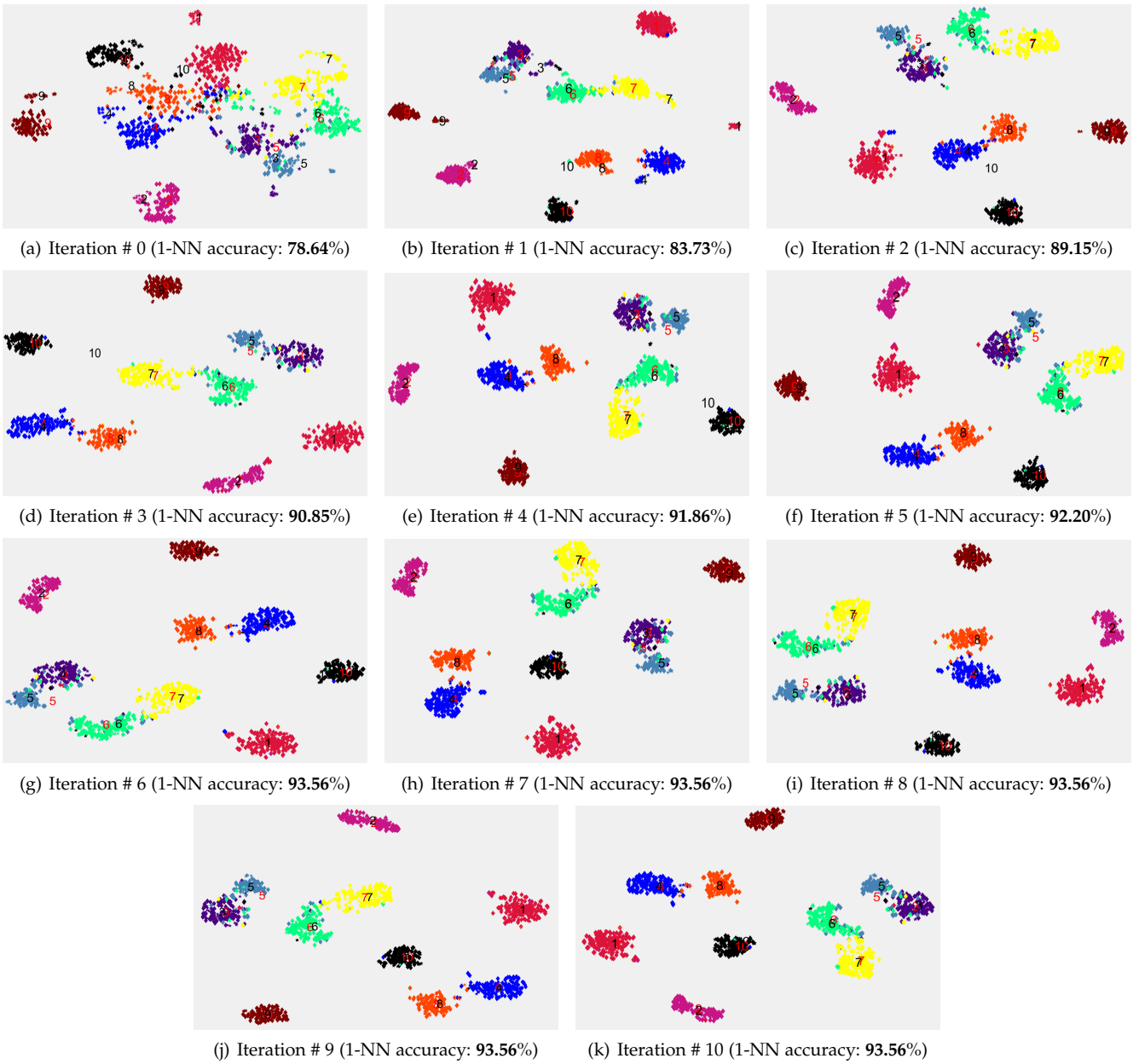
For simplicity, we indeed utilize a simple and non-parametric 1-NN classifier for pseudo-labeling for Algorithm 1. Actually, we also tried other classifiers like support vector classification (SVM, following the default setting in the package Liblinear [2]) and label propagation (LP, inspired by recent works [13], [14], details can be referred in last Section 5), the experimental results are shown here in Table 8, where the classifier (called pse-class, step 6 in Algorithm 1) for pseudo-labeling varies in [1-NN, SVM, LP] and the final classifier (fin-class, step 10 in Algorithm 1) for final target label prediction in a basic method also varies between [1-NN, SVM, LP].

As can be seen in Table 8, if we choose 1-NN as the final classifier, SVM is the optimal classifier for pseudo labeling during the training process, however, both three classifiers perform roughly comparable. On the other hand, if we choose SVM as the final classifier, 1-NN provides the best recognition results, utilizing SVM for pseudo labeling performs worse than others due to the bad performance in C→W. Furthermore, taking LP as the final classifier, 1-NN and LP perform almost equable, while SVM performs slightly badly in terms of the average accuracy. Carefully observing the table, we can find that once SVM is incorporated with LP, the performance stays unchangeable or increases for almost all tasks. In addition, 1-NN also benefits from the incorporation of LP and SVM as the final classifier, the averaged accuracy from 91.4% to 91.7% and 92.5%. Besides the non-parametric property and simplicity, 1-NN always obtains promising and stable results regardless of the final classifier, making it a default classifier for pseudo labeling in our algorithms.

1. https://lvdmaaten.github.io/tsne/
2. https://www.csie.ntu.edu.tw/~cjlin/liblinear/

References

[1] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proc. ICCV*, 2013.
[2] J. Zhang, W. Li, and P. Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," in *Proc. CVPR*, 2017.



(a) Iteration # 0 (1-NN accuracy: **78.64**%) (b) Iteration # 1 (1-NN accuracy: **83.73**%) (c) Iteration # 2 (1-NN accuracy: **89.15**%)

(d) Iteration # 3 (1-NN accuracy: **90.85**%) (e) Iteration # 4 (1-NN accuracy: **91.86**%) (f) Iteration # 5 (1-NN accuracy: **92.20**%)

(g) Iteration # 6 (1-NN accuracy: **93.56**%) (h) Iteration # 7 (1-NN accuracy: **93.56**%) (i) Iteration # 8 (1-NN accuracy: **93.56**%)

(j) Iteration # 9 (1-NN accuracy: **93.56**%) (k) Iteration # 10 (1-NN accuracy: **93.56**%)

Fig. 5. T-SNE visualizations of embedding subspaces w.r.t. the number of iterations. Iteration # 0 denotes the original features, and source and target instances are represented by diamond (◇) and five-pointed star (pentagram) ★, respectively. We also mark each class center with its corresponding class index for the source and target domains in red and black respectively.

[3] L. Luo, X. Wang, S. Hu, and L. Chen, "Robust data geometric structure aligned close yet discriminative domain adaptation," *arXiv preprint:1705.08620*, 2017.

[4] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshops*, 2011.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[7] O. Sener, H. O. Song, A. Saxena, and S. Savarese, "Learning transferrable representations for unsupervised domain adaptation," in *Proc. NIPS*, 2016.

[8] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. CVPR*, 2017.

[9] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proc. ICCV*, 2013.

[10] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proc. ICCV*, 2015.

[11] S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto, "Few-shot adversarial domain adaptation," in *Proc. NIPS*.

[12] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. NIPS*, 2004.

[13] L. Luo, L. Chen, S. Hu, Y. Lu, and X. Wang, "Discriminative and geometry aware unsupervised domain adaptation," *arXiv preprint:1712.10042*, 2017.

[14] C.-A. Hou, Y.-H. H. Tsai, Y.-R. Yeh, and Y.-C. F. Wang, "Unsupervised domain adaptation with label and structural consistency," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5552–5562, 2016.