When Unsupervised Domain Adaptation Meets Tensor Representations Supplementary Materials[*]

Hao Lu[†], Lei Zhang[‡], Zhiguo Cao[†], Wei Wei[‡], Ke Xian[†], Chunhua Shen[§], Anton van den Hengel[§]
[†]School of Automation, Huazhong Unviersity of Science and Technology, China
[‡]School of Computer Science and Engineering, Northwestern Polytechnical University, China
[§]School of Computer Science, The University of Adelaide, Australia
{poppinace,zgcao}@hust.edu.cn

In this Supplementary, we will expand more details that are not included in the main text due to the page limitation. In particular, we supplement the following content on

- how to implement the optimization of our approach efficiently;
- how to perform spatial pooling normalization to convolutional activations; we only briefly mention this procedure in Section 5.1 of the main text;
- detailed introduction regarding used datasets;
- additional results evaluated on Office and ImageNet–VOC2007 datasets;
- parameters sensitivity.

1. Towards efficient optimization

In this section, we will reveal several important details towards efficient practical implementations. Note that $\mathcal{X}_s \in \mathbb{R}^{n_1\times\cdots\times n_K\times N_s}$ is a $(K{+}1)$-mode tensor, the unfolding matrix $\boldsymbol{X}_{s(k)}$ is of size $n_k{\times}n_{\backslash k}N_s$, where $n_{\backslash k}=n_1\cdots n_{k-1}n_{k+1}\cdots n_K$. When computing $\boldsymbol{Q}^{(k)}=\boldsymbol{X}_{s(k)}\boldsymbol{M}_{\backslash k}^T$ in Eq. (13), $\boldsymbol{M}_{\backslash k}^T$ will be of size $n_{\backslash k}N_s\times n_{\backslash k}N_s$, which is extremely large and consume a huge amount of memory to store. In fact, such a matrix even cannot be constructed in a general-purpose computer. To alleviate this, we choose to solve an equivalent optimization problem by reformulating Eq. (13) into its sum form as

$$\begin{aligned}\min_{\boldsymbol{M}^{(k)}}\ &\sum_{n=1}^{N_s}\|\boldsymbol{M}^{(k)}\boldsymbol{Q}^n_{(k)}-\boldsymbol{Y}^n_{(k)}\|_F^2-\lambda\|\boldsymbol{M}^{(k)}\boldsymbol{X}^n_{s(k)}\|_F^2\\ \text{s.t. }&\forall k,\ \boldsymbol{M}^{(k)}\boldsymbol{M}^{(k)T}=\boldsymbol{I}\end{aligned}\ , \qquad (1)$$

where

$$\begin{aligned}\boldsymbol{Q}^n_{(k)}&=\boldsymbol{X}^n_{s(k)}\hat{\boldsymbol{M}}_{\backslash k}^T\\ \hat{\boldsymbol{M}}_{\backslash k}^T&=\boldsymbol{M}^{(K)}\otimes\cdots\otimes\boldsymbol{M}^{(k+1)}\otimes\boldsymbol{M}^{(k-1)}\otimes\cdots\otimes\boldsymbol{M}^{(1)}\end{aligned}\ , \qquad (2)$$

$\boldsymbol{Y}^n_{(k)}=\boldsymbol{Y}_{(k)}(:,:,n)$ ($\boldsymbol{Y}_{(k)}$ has been reshaped into the size of $n_k\times n_{\backslash k}\times N_s$), and $\boldsymbol{X}^n_{s(k)}$ denotes the k-th mode unfolding matrix of $\mathcal{X}^n_s$. In following expressions, we denote $\boldsymbol{Q}^n_{(k)}$, $\boldsymbol{Y}^n_{(k)}$, and $\boldsymbol{X}^n_{s(k)}$ by $\boldsymbol{Q}_n$, $\boldsymbol{Y}_n$, and $\boldsymbol{X}_n$ for short, respectively. By replacing $\boldsymbol{M}^{(k)T}$ with $\boldsymbol{P}$, we arrive at

$$\begin{aligned}\min_{\boldsymbol{P}}\ &\sum_{n=1}^{N_s}\|\boldsymbol{Q}_n^T\boldsymbol{P}-\boldsymbol{Y}_n^T\|_F^2-\lambda\|\boldsymbol{X}_n^T\boldsymbol{P}\|_F^2\\ \text{s.t. }&\forall k,\ \boldsymbol{P}^T\boldsymbol{P}=\boldsymbol{I}\end{aligned}\ . \qquad (3)$$

[*]H. Lu and L. Zhang contributed equally. This work was done when H. Lu, L. Zhang, and K. Xian were visiting The University of Adelaide. Z. Cao is the corresponding author.

Considering that a standard solver needs the loss function $\mathcal{F}$ and its gradient $\partial\mathcal{F}/\partial\boldsymbol{P}$ as the input, we can compute them in the following way to speed up the optimization process. For the loss function $\mathcal{F}$, we have

$$\begin{aligned}
\mathcal{F} &= \sum_{n=1}^{N_s} \|\boldsymbol{Q}_n^T\boldsymbol{P} - \boldsymbol{Y}_n^T\|_F^2 - \lambda\|\boldsymbol{X}_n^T\boldsymbol{P}\|_F^2 \\
&= \sum_{n=1}^{N_s} Tr\left[(\boldsymbol{Q}_n^T\boldsymbol{P} - \boldsymbol{Y}_n^T)^T(\boldsymbol{Q}_n^T\boldsymbol{P} - \boldsymbol{Y}_n^T)\right] - \lambda\sum_{n=1}^{N_s} Tr\left[(\boldsymbol{X}_n^T\boldsymbol{P})^T(\boldsymbol{X}_n^T\boldsymbol{P})\right] \\
&= \sum_{n=1}^{N_s} Tr\left[\boldsymbol{P}^T\boldsymbol{Q}_n\boldsymbol{Q}_n^T\boldsymbol{P} - 2\boldsymbol{P}^T\boldsymbol{Q}_n\boldsymbol{Y}_n^T + \boldsymbol{Y}_n\boldsymbol{Y}_n^T\right] - \lambda\sum_{n=1}^{N_s} Tr\left[\boldsymbol{P}^T\boldsymbol{X}_n\boldsymbol{X}_n^T\boldsymbol{P}\right] \\
&= Tr\left[\boldsymbol{P}^T(\sum_{n=1}^{N_s}\boldsymbol{Q}_n\boldsymbol{Q}_n^T)\boldsymbol{P}\right] - 2Tr\left[\boldsymbol{P}^T(\sum_{n=1}^{N_s}\boldsymbol{Q}_n\boldsymbol{Y}_n^T)\right] + Tr\left[\sum_{n=1}^{N_s}\boldsymbol{Y}_n\boldsymbol{Y}_n^T\right] - \lambda Tr\left[\boldsymbol{P}^T(\sum_{n=1}^{N_s}\boldsymbol{X}_n\boldsymbol{X}_n^T)\boldsymbol{P}\right]
\end{aligned}, \quad (4)$$

where $Tr[\cdot]$ denotes the trace of matrix. For the gradient $\partial\mathcal{F}/\partial\boldsymbol{P}$, we have

$$\begin{aligned}
\partial\mathcal{F}/\partial\boldsymbol{P} &= 2\sum_{n=1}^{N_s}\boldsymbol{Q}_n(\boldsymbol{Q}_n^T\boldsymbol{P} - \boldsymbol{Y}_n^T) - 2\lambda\sum_{n=1}^{N_s}\boldsymbol{X}_n\boldsymbol{X}_n^T\boldsymbol{P} \\
&= 2(\sum_{n=1}^{N_s}\boldsymbol{Q}_n\boldsymbol{Q}_n^T)\boldsymbol{P} - 2\sum_{n=1}^{N_s}\boldsymbol{Q}_n\boldsymbol{Y}_n^T - 2\lambda(\sum_{n=1}^{N_s}\boldsymbol{X}_n\boldsymbol{X}_n^T)\boldsymbol{P}
\end{aligned}. \quad (5)$$

Notice that both $\mathcal{F}$ and $\partial\mathcal{F}/\partial\boldsymbol{P}$ share some components. As a consequence, we can precompute $\sum_{n=1}^{N_s}\boldsymbol{Q}_n\boldsymbol{Q}_n^T$, $\sum_{n=1}^{N_s}\boldsymbol{Q}_n\boldsymbol{Y}_n^T$, $\sum_{n=1}^{N_s}\boldsymbol{Y}_n\boldsymbol{Y}_n^T$, and $\sum_{n=1}^{N_s}\boldsymbol{X}_n\boldsymbol{X}_n^T$ before the $\mathcal{M}$-step optimization instead of directly feeding the original variables and iteratively looping over N_s samples inside the optimization. Such a kind of precomputation speeds up the optimization significantly.

2. Feature normalization with spatial pooling

Since we allow the input image to be of arbitrary size, a normalization step need to perform to ensure the consistency of dimensionality. The idea of spatial pooling is similar to the spatial pyramid pooling in [5]. The difference is that we do not pool pyramidally and do not vectorize the pooled activations, in order to preserve the spatial information. Intuitively, Fig. 1 illustrates this process. More concretely, convolutional activations are first equally divided into N_{bin} bins along the spatial modes ($N_{bin} = 16$ in Fig. 1). Next, each bin with size of $h \times w$ is normalized to a $s \times s$ bin by max pooling. In our experiments, we set $N_{bin} = 36$ and $s = 1$.

3. Datasets and protocol details

Office–Caltech10 dataset. As mentioned in the main text, [4] extends Office [8] dataset by adding another *Caltech* domain. They select 10 common categories from four domains, including *Amazon*, *DSLR*, *web-cam*, and *Caltech*. *Amazon* consists of images used in the online market, which shows the objects from a canonical viewpoint. *DSLR* contains images captured with a high-resolution digital camera. Images in *web-cam* are recorded using a low-end webcam. *Caltech* is similar to *Amazon* but with various viewpoint variations. The 10 categories include `backpack`, `bike`, `calculator`, `headphones`,

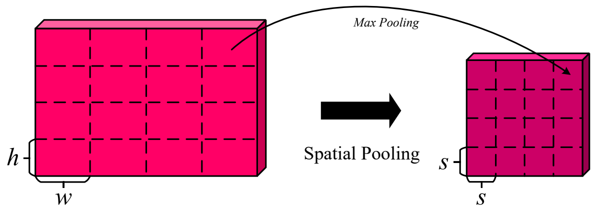

Figure 1: Illustration of spatial pooling normalization. Any size of convolutional representations will be normalized to a fixed-size tensor.

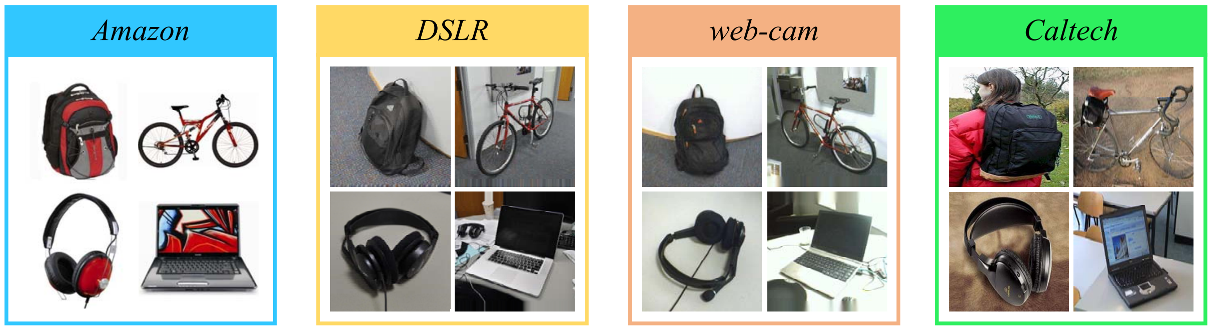

Figure 2: Some images from Office–Caltech10 dataset. 4 categories of `backpack`, `bike`, `headphone`, and `laptop computer` are selected.

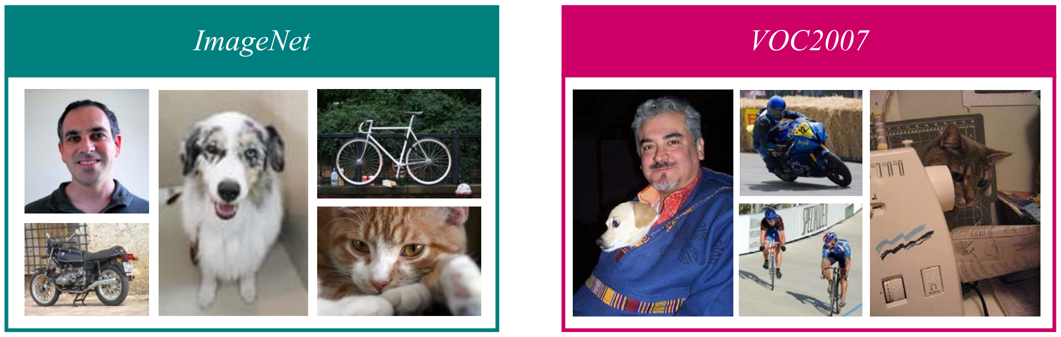

Figure 3: Some images from ImageNet-VOC2007 dataset. 5 categories of `person`, `dog`, `motorbike`, `bicycle`, and `cat` are presented.

`keyboard`, `laptop computer`, `monitor`, `mouse`, `mug`, and `projector`. Some images of four domains are shown in Fig. 2. Overall, we have about 2500 images and 12 domain adaptation problems. For each problem, we repeat the experiment 20 times. In each trail, we randomly select 20 images from each category for training if the domain is *Amazon* and *Caltech*, or 8 images if the domain is *DSLR* or *web-cam*. All images in the target domain are employed in the both adaptation and testing stages. The mean and standard deviation of multi-class accuracy are reported.

Office dataset. Office dataset is developed by [8] and turns out to be a standard benchmark for the evaluation of domain adaptation. It consists of 31 categories and 3 domains, leading to 6 domain adaptation problems. Among these 31 categories, only 16 overlap with the categories contained in the 1000-category ImageNet 2012 dataset[1] [6], so Office dataset is more challenging than its counterpart Office-Caltech10 dataset. We follow the same experimental protocol mentioned above to conduct the experiments, so in each task we have 620 images in all from the source domain.

ImageNet–VOC2007 dataset. As described in the main text, ImageNet and VOC 2007 datasets are used to evaluate the domain adaptation performance from single-label to multi-label situation. The same 20 categories as the VOC 2007 dataset are chosen from original ImageNet dataset. These 20 categories are `aeroplane`, `bicycle`, `bird`, `boat`, `bottle`, `bus`, `car`, `cat`, `chair`, `cow`, `dining table`, `dog`, `horse`, `motorbike`, `person`, `potted plant`, `sheep`, `sofa`, `train`, and `tv monitor`. The 20-category ImageNet subset is adopted as the source domain, and the `test` subset of VOC2007 is employed as the target domain. Some images of two domains are illustrated in Fig. **??**. Also, the similar experimental protocol mentioned above is used. The difference, however, is that we report the mean and standard deviation of average precision (AP) for each category, respectively.

[1]The 16 overlapping categories are `backpack`, `bike helmet`, `bottle`, `desk lamp`, `desk computer`, `file cabinet`, `keyboard`, `laptop computer`, `mobile phone`, `mouse`, `printer`, `projector`, `ring binder`, `ruler`, `speaker`, and `trash can`.

Method	Feature	A→D	D→A	A→W	W→A	D→W	W→D	MEAN
NA	VCONV	53.8(2.3)	39.3(1.7)	47.7(1.7)	36.3(1.6)	77.4(1.7)	81.3(1.5)	56.0
PCA	VCONV	40.5(3.3)	38.2(2.6)	36.5(2.9)	37.8(2.9)	68.7(2.5)	70.5(2.6)	48.7
DAUME	VCONV	48.4(2.5)	35.2(1.5)	42.5(2.0)	33.6(1.8)	68.4(2.5)	74.2(2.4)	50.4
TCA	VCONV	30.3(4.5)	20.1(4.4)	27.0(3.1)	18.1(3.0)	51.1(3.2)	53.0(3.2)	33.3
GFK	VCONV	47.4(4.7)	36.2(2.9)	41.5(3.5)	33.4(2.6)	75.3(1.6)	78.0(2.4)	51.9
DIP	VCONV	36.8(4.5)	13.8(1.8)	29.6(5.0)	17.8(2.6)	77.4(1.8)	81.5(2.0)	42.8
SA	VCONV	28.6(3.5)	37.1(2.1)	29.0(2.1)	34.9(2.9)	75.1(2.4)	75.1(2.7)	46.6
LTSL	VCONV	32.0(5.5)	28.6(1.6)	24.2(3.7)	27.1(2.0)	60.9(4.0)	73.9(3.3)	41.1
LSSA	VCONV	**56.6(2.0)**	45.6(1.6)	**52.2(1.6)**	40.7(2.0)	73.0(2.1)	63.5(3.8)	55.3
CORAL	VCONV	39.9(1.7)	42.7(0.9)	39.7(1.7)	40.7(1.0)	82.0(1.3)	79.5(1.4)	54.1
NTSL	TCONV	56.1(2.4)	45.7(1.5)	50.8(2.3)	42.6(2.2)	84.4(1.6)	88.2(1.4)	61.3
TAISL	TCONV	56.4(2.4)	**45.9(1.1)**	50.7(2.0)	**43.2(1.7)**	**84.5(1.4)**	**88.5(1.2)**	**61.5**

Table 1: Average multi-class recognition accuracy (%) on Office dataset over 20 trials. The highest accuracy in each column is boldfaced, the second best is marked in red, and standard deviations are shown in parentheses.

4. Recognition results

We compare against the same methods used in the main text, including the baseline no adaptation (NA), principal components analysis (PCA), transfer component analysis (TCA) [7], geodesic flow kernel (GFK) [4], domain-invariant projection (DIP) [2], subspace alignment (SA) [3], low-rank transfer subspace learning (LTSL) [9], landmarks selection subspace alignment (LSSA) [1], and correlation alignment (CORAL) [11]. Our approach is denoted by NTSL (the naive version) and TAISL. We also extract convolutional activations from the CONV5_3 layer of the VGG–VD–16 model [10]. We mark the feature as VCONV and TCONV for vectorized and tensor-form convolutional activations, respectively. The same parameters described in the main text are set to report the results.

Office results. Results of the Office dataset are listed in Table 1. Similar to the tendency shown by the results of Office-Caltech10 dataset in the main text, our approach outperforms or is on par with other comparing methods. It is interesting that sometimes NTSL even achieves better results than TAISL. We believe such results are sound, because a blind global adaptation cannot always achieve accuracy improvement. However, it is clear that learning an invariant tensor space works much better than learning a shared vector space. Furthermore, the joint learning effectively reduces the standard deviation and thus improves the stability of the adaptation.

ImageNet–VOC2007 results. Table 2 shows the complete results on ImageNet–VOC2007 dataset (only partial results are presented in the main text due to the page limitation). Our approach achieves the best mean accuracy in 4 and the second best in 6 out of 20 categories. In general, when noisy labels exist in the target domain, our approach demonstrates a stable improvement in accuracy. Moreover, compared to the baseline NTSL, the standard deviation is generally reduced, which means aligning the source domain to the target not only promotes the classification accuracy but also improves the stability of tensor space.

5. Parameters Sensitivity

Here we investigate the sensitivity of 3 parameters involved in our approach. Specifically, they are the spatial mode dimensionality d_s (d_1 and d_2 in the main text, we assume $d_1 = d_2 = d_s$), the feature mode dimensionality d_f (d_3 in the main text), and the weight coefficient λ. We monitor how the classification accuracy changes when these parameters vary. At each time, only one parameter is allowed to change. By default, $d_s = 6$, $d_f = 128$, and $\lambda = 1e^{-5}$. A DA task of W→C from the Office-Caltech10 dataset is chosen. Results are illustrated by Fig. 4. According to Fig. 4, we can make the following observations:

- In general, there exhibits a tendency for increased d_s to increased classification accuracy, which implies that the adaptation can benefit from extra spatial information. This is why we preserve the original spatial mode as it is.

VOC 2007 test		aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	
NA	vCONV	66.4(2.1)	65.3(2.9)	65.6(4.0)	56.1(9.0)	29.5(2.1)	51.2(3.4)	70.9(4.5)	70.6(3.4)	19.3(2.0)	30.3(8.0)	
PCA	vCONV	28.9(5.8)	25.3(7.2)	30.2(3.9)	14.0(4.8)	23.3(5.2)	15.6(6.3)	41.5(7.5)	44.9(4.5)	11.2(0.9)	6.0(1.8)	
Daumé III	vCONV	64.1(3.7)	60.4(4.2)	59.7(7.4)	53.5(7.8)	26.6(3.3)	49.0(5.1)	66.3(5.1)	65.7(5.3)	18.6(3.5)	26.9(8.5)	
TCA	vCONV	43.2(9.8)	46.0(17.0)	44.4(10.5)	25.3(13.0)	20.7(1.7)	30.4(7.7)	59.5(8.6)	56.7(8.2)	17.1(3.0)	16.9(6.3)	
GFK	vCONV	70.0(6.9)	**66.0(7.6)**	74.6(3.8)	40.7(11.8)	32.5(4.4)	55.0(6.9)	71.3(5.2)	73.1(6.5)	16.3(3.6)	28.9(5.3)	
DIP	vCONV	69.8(5.5)	65.8(7.2)	78.4(4.6)	34.2(9.1)	29.1(5.0)	54.4(7.3)	**75.7(3.9)**	75.9(3.7)	20.1(4.6)	25.5(5.0)	
SA	vCONV	64.4(10.1)	54.4(9.3)	69.3(5.4)	50.8(12.7)	34.4(4.6)	50.8(6.5)	64.3(9.5)	67.4(4.9)	11.2(1.9)	18.4(6.6)	
LTSL	vCONV	56.9(10.4)	59.8(6.3)	61.0(7.7)	50.6(15.6)	34.9(6.2)	50.9(9.6)	66.9(3.6)	70.8(8.8)	11.4(1.5)	21.9(6.3)	
LSSA	vCONV	**78.7(2.0)**	71.8(1.5)	**79.7(1.2)**	18.5(2.0)	**38.4(4.6)**	**64.1(3.2)**	69.4(2.2)	**81.7(0.5)**	**57.2(2.4)**	29.5(1.9)	
CORAL	vCONV	71.4(3.3)	63.3(4.3)	71.7(3.6)	58.6(9.5)	35.2(2.4)	61.9(3.6)	62.7(7.1)	72.0(4.3)	18.7(2.7)	**36.0(5.7)**	
NTSL	TCONV	76.3(4.3)	61.6(5.5)	71.0(3.9)	**65.9(8.3)**	35.7(3.7)	56.1(7.1)	70.1(4.8)	71.3(3.2)	16.6(2.6)	34.7(9.8)	
TAISL	TCONV	76.4(5.1)	62.3(4.8)	71.6(3.1)	64.9(7.7)	36.7(3.5)	57.0(6.6)	71.2(4.3)	72.0(2.1)	15.7(2.9)	33.3(6.6)	
		table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
NA	vCONV	35.7(5.5)	47.9(6.4)	35.5(11.4)	47.0(8.0)	69.3(2.9)	25.6(3.9)	44.9(6.9)	46.9(5.3)	71.8(4.4)	56.4(3.3)	50.3
PCA	vCONV	29.0(6.9)	32.5(4.6)	23.2(6.2)	25.0(5.0)	70.2(1.9)	9.3(4.3)	11.7(3.5)	16.2(2.8)	29.0(6.7)	29.0(6.6)	25.8
Daumé III	vCONV	30.0(5.4)	43.6(6.9)	28.3(8.7)	40.5(6.6)	68.5(2.5)	23.6(3.5)	37.7(7.4)	44.5(5.6)	67.6(5.4)	51.9(4.4)	46.4
TCA	vCONV	27.6(8.7)	43.2(7.6)	29.0(14.6)	31.8(10.2)	58.1(5.7)	11.6(4.5)	22.7(8.0)	24.0(9.4)	52.3(8.9)	33.6(10.2)	34.7
GFK	vCONV	48.3(10.4)	56.7(7.4)	59.2(16.7)	58.3(4.8)	**75.8(3.6)**	15.2(4.6)	52.5(4.8)	44.7(6.0)	79.9(4.9)	57.1(4.5)	53.8
DIP	vCONV	42.2(8.1)	53.7(5.4)	**64.7(7.1)**	56.3(5.7)	73.5(3.1)	14.7(4.2)	48.9(4.3)	39.8(10.0)	**80.5(5.6)**	59.4(5.2)	53.1
SA	vCONV	36.9(12.8)	54.2(5.7)	39.5(15.5)	53.7(10.9)	68.9(2.4)	20.9(6.7)	31.4(10.2)	29.3(6.1)	73.5(5.2)	55.2(5.7)	47.4
LTSL	vCONV	43.7(12.4)	55.4(7.4)	53.4(13.1)	52.5(10.7)	69.9(4.3)	18.8(8.2)	38.2(9.5)	28.9(13.2)	67.1(9.9)	54.0(7.5)	48.3
LSSA	vCONV	33.7(3.4)	56.9(2.5)	41.1(5.4)	56.3(9.3)	51.2(2.0)	15.3(5.7)	32.5(10.6)	43.4(8.4)	81.1(1.4)	51.6(4.6)	52.6
CORAL	vCONV	40.6(6.7)	53.8(5.3)	34.8(6.8)	57.3(5.6)	67.6(2.0)	24.2(1.5)	**54.8(2.9)**	47.7(6.2)	71.7(3.5)	56.9(3.6)	53.0
NTSL	TCONV	49.8(10.4)	**58.3(5.1)**	40.9(12.9)	59.7(10.2)	72.0(4.6)	25.0(4.5)	53.4(6.0)	**49.8(4.6)**	75.3(3.6)	60.2(3.5)	55.2
TAISL	TCONV	**50.7(10.0)**	57.6(3.8)	39.0(14.0)	**60.3(8.7)**	72.2(3.8)	**26.6(5.4)**	53.6(5.6)	49.8(5.6)	74.2(4.9)	**60.4(3.5)**	**55.3**

Table 2: Average precision (%) on ImageNet-VOC2007 dataset over 10 trials. The highest AP in each column is boldfaced, the second best is marked in red, and standard deviations are shown in parentheses.

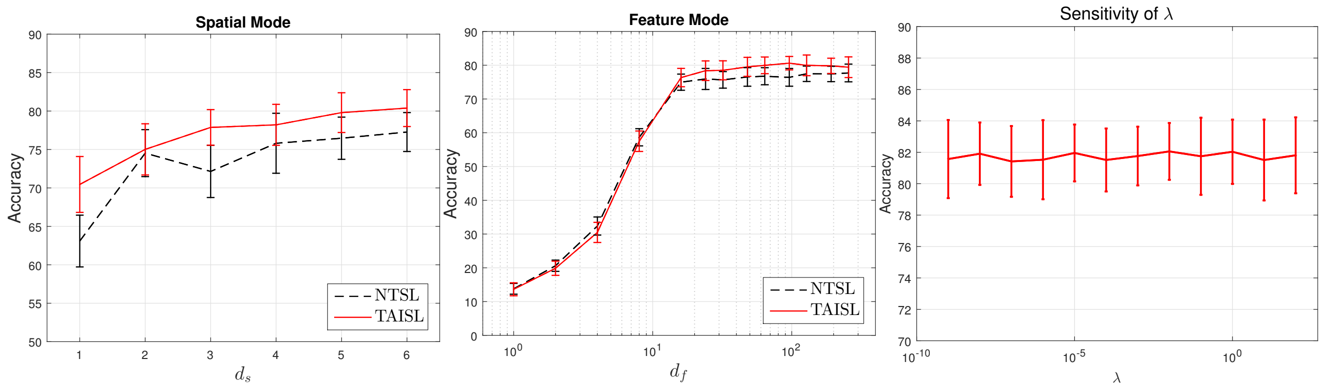

Figure 4: Sensitivity of tensor subspace dimensionality and weight coefficient λ on the DA task of W→C.

- As per the feature mode dimensionality d_f, a dramatic growth appears when d_f increases from 1 to 16. However, the classification accuracy starts to level off when d_f exceeds 16. Such results make sense, because when the feature dimensionality is relatively small, the discriminative power of feature representations cannot be guaranteed. Overall, our approach demonstrates stable classification performance over a wide range of feature mode dimensionality.
- Only a slight fluctuation occurs when λ varies between $1e^{-9}$ and $1e^{1}$. The classification accuracy is virtually insensitive to the weight coefficient λ. This is another good property of our approach.

References

[1] R. Aljundi, R. Emonet, D. Muselet, and M. Sebban. Landmarks-based kernelized subspace alignment for unsupervised domain adaptation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 56–63, 2015. 4

[2] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 769–776, 2013. 4

[3] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 2960–2967, 2013. 4

[4] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073, 2012. 2, 4

[5] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015. 2

[6] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and T. Darrell. One-shot adaptation of supervised deep convolutional models. In *Proc. International Conference on Learning Representations Workshops (ICLRW)*, 2013. 3

[7] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, Feb 2011. 4

[8] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Proc. European Conference on Computer Vision (ECCV)*, pages 213–226, 2010. 2, 3

[9] M. Shao, D. Kit, and Y. Fu. Generalized transfer subspace learning through low-rank constraint. *International Journal of Computer Vision*, 109(1-2):74–93, 2014. 4

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4

[11] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *Proc. AAAI Conference on Artificial Intelligence*, 2016. 4