Distant Supervised Centroid Shift: A Simple and Efficient Approach to Visual Domain Adaptation

Jian Liang, Ran He, Zhenan Sun and Tieniu Tan

CRIPAC & NLPR, Institute of Automation, Chinese Academy of Sciences (CAS), University of Chinese Academy of Sciences, CAS Center for Excellence in Brain Science and Intelligence Technology

Abstract

Conventional domain adaptation methods usually resort to deep neural networks or subspace learning to find invariant representations across domains. However, most deep learning methods highly rely on large-size source domains and are computationally expensive to train, while subspace learning methods always have a quadratic time complexity that suffers from the large domain size. This paper provides a simple and efficient solution, which could be regarded as a well-performing baseline for domain adaptation tasks. Our method is built upon the nearest centroid classifier, seeking a subspace where the centroids in the target domain are moderately shifted from those in the source domain. Specifically, we design a unified objective without accessing the source domain data and adopt an alternating minimization scheme to iteratively discover the pseudo target labels, invariant subspace, and target centroids. Besides its privacy-preserving property (distant supervision), the algorithm is provably convergent and has a promising linear time complexity. In addition, the proposed method can be readily extended to multi-source setting and domain generalization, and it remarkably enhances popular deep adaptation methods by borrowing the learned transferable features. Extensive experiments on several benchmarks including object, digit, and face recognition datasets validate that our methods yield state-of-the-art results in various domain adaptation tasks.

1. Introduction

Traditional machine learning paradigms always assume that the training data and the testing data come from the same distribution, however, this assumption does not always hold in real-world applications [52, 66]. To avoid the expensive and time-consuming data labeling step, massive efforts over the last decade have been devoted to transfer learning [64, 8, 34, 36] and multi-task learning [14, 30, 63, 70] that leverage the latent relationship with previous datasets to learn a new model for an emerging dataset.

Taking a night object image recognition problem for example, it is not desirable to neither train a model on existing day-time object images to recognize these night object images nor acquire massive labeled object images at night to re-train a model on them from scratch. By contrast, we expect to transfer the knowledge from existing day-time object images to recognizing these unlabeled night object images, which is also known as domain adaptation, and such day-time and night images are termed as source and target domains, respectively. Based on the availability of partial labeled target data, domain adaptation can be roughly divided into two categories, unsupervised and semi-supervised domain adaptation. In this paper, we mainly focus on the challenging unsupervised domain adaptation problem where the labels of target data are totally unknown.

Since the degradation in performance mainly arises from the covariate shift (i.e., the change in the data distribution of the source and target domains), early approaches [72, 58] favor an intuitive strategy named instance re-weighting, which tries to align two different domains via estimating naturally the ratio between the likelihoods of being a source or target example. Later studies [42, 7] exploit one favorite distribution measure named Maximum Mean Discrepancy (MMD) [24] to weigh data instances. However,
these instance-based adaptation methods require the strict assumptions [52] that are hard to satisfy.

Alternatively, a growing number of recent studies focus on learning transferable representations via deep neural networks or subspace discovery since they do not require the strong assumption. Deep domain adaptation methods are roughly divided into three main categories, discrepancy-based methods [60, 39, 44, 73, 68], adversarial-based methods [17, 38, 2, 63, 64], and reconstruction-based methods [19, 3, 76]. However, these batch-wise deep learning methods cannot fully exploit the global information and address small-size source domains well. Subspace alignment methods [23, 15] try to align the subspaces (e.g., PCA) of different domains together. Later studies [59, 32, 73] further consider the second-order and higher-order scatter matrices (moments) across different domains. These subspace-centric methods are rather easy and efficient to deploy, yet, they fail to minimize the data distributions between domains after aligning the subspaces. [41] is a typical data-centric subspace discovery based approach that learns to project both domains onto one subspace where the cross-domain joint distribution discrepancy is minimized. Following studies [74, 36, 67] are built upon [41] by considering coupled projections, discriminative target structure and joint classifier learning, respectively. Even these data-centric methods achieve promising results, yet, they always involve a heuristic pseudo target label estimation step in the EM-like algorithm, making the overall optimization problem hard to converge theoretically. Additionally, all the methods mentioned above involve several large MMD matrices [41], thus, they all have a quadratic time complexity w.r.t. the domain size and can not cope with large-scale datasets well.

In this paper, we propose a simple, efficient, yet effective approach via subspace discovery for unsupervised domain adaptation. Inspired by [12], we develop a unified objective that assumes the centroids in the target domain are moderately shifted from those in the source domain, and each instance is closest to its corresponding centroid for both domains in the projected subspace. \textit{Note that, this objective does not need to access the source domain data like [6], making it a privacy-preserving method.} Then we adopt an alternating minimization scheme to iteratively discover the pseudo target labels, adaptive target centroids, and invariant subspace learning. Specifically, each subproblem has a closed-form solution. Theoretical analysis shows that our algorithm is convergent and efficient with a linear time complexity. In addition, the proposed method can be readily extended to multi-source setting and domain generalization, and it even remarkably enhances popular deep domain methods by borrowing the learned transferable features. Extensive experiments on several benchmarks including object (i.e., Office31 [55], Office-Caltech [21] and Office-Home [66], VLCS [62]), Digits, and face (i.e., PIE [1]) recognition datasets validate that our methods achieve state-of-the-art results in the vanilla unsupervised domain adaptation, domain generalization, and multi-source domain adaptation tasks. Generally, our algorithm is impressively simple and efficient, making it a strong baseline for domain adaptation and generalization tasks.

2. Related Work

The last decade has witnessed a boom in studies towards domain adaptation and related applications. We refer the interested reader to [66, 9] for a survey focusing on computer vision applications. As stated above, both feature transformation [51, 41, 74, 36] and feature representation learning [19, 53, 63, 68, 53] are much more favored by recent domain adaptation approaches. Here we analyze several most closely related work from both cases to our method.

Regarding shallow feature transformation based approaches, [36] proposes a general objective in order to pursue that instances from the same class of both domains are dragged closer to each other, which includes many former methods [51, 41] as special cases. Our method can be considered to be built upon [36] by developing a built-in classifier to infer the pseudo target labels, and it only acquires distant supervisions, \textit{i.e.}, class-wise Gaussian estimators of means and covariance matrices, instead of accessing the entire source domain data. Besides, previous methods always involve several large MMD matrices [41, 74, 36, 67] in the optimization procedure with a $O(n^2)$ computation complexity, making it unsuitable for large-scale adaptation scenarios. In fact, there are several previous studies [11, 10] that attempt to investigate Nearest Class Means (NCM) for domain adaptation. However, they merely integrate NCM with other domain adaptation models [61, 5] as a novel classifier, ignoring the distribution discrepancy after projection.

Benefiting from the rapid development of deep neural networks, deep domain adaptation methods achieve much better performance. Generally, a majority of them (e.g., [65, 17, 60, 64, 68]) equip the source domain a source classification loss function and design another cross-domain loss function (e.g., MMD or adversarial loss in Generative Adversarial Network (GAN)) [22]) to align two domains. SimNet [53] replaces the conventional source classifier (e.g., soft-max) with a prototype similarity-based classifier and adopts a domain discriminator for domain confusion, which achieves state-of-the-art performances. Furthermore, [19] utilizes a reconstruction loss for target domain and [53] designed a similarity-based classifier for labeled source domain. However, these methods are always optimized in a batch-wise manner, making it not suitable to minimize a global loss function like MMD. Our approach, applied to learned transferable features extracted from fine-tuned models and deep domain adaptation methods like DAN [39], RevGrad [17] and GTA [56], achieves better per-
performance to these more complex methods and is expected to be incorporated directly into the network structure.

3. Methodology

3.1. Problem Definition

In conventional domain adaptation with two domains, we have access to labeled images \( X_s = \{ (x_i^s, y_i^s) \}_{i=1}^n \) drawn from a source domain distribution \( p_s(x, y) \) and target images \( X_t = \{ (x_i^t, y_i^t) \}_{i=1}^m \) drawn from a target domain distribution \( p_t(x, y) \). It is commonly assumed that the label space of both domains are identical, i.e., \( y_i^s, y_i^t \in \{ 1, 2, \ldots, C \} \) and the length of domain feature space is the same, i.e., \( x_i^s, x_i^t \in \mathbb{R}^d \). Note that, in the unsupervised setting, we have no information about the labels on the target domain. Without loss of generality, we assume that each feature vector is normalized to satisfy \( \| x_i \|^2 = 1, q \in \{ s, t \} \), then zero-centered, i.e., \( \sum_i x_i^q = 0 \) and \( \sum_i x_i^q = 0 \).

In fact, instead of using the original data in the source domain, we only require the number of samples in each class (i.e., \( m_r \)), the maximum likelihood estimators of the means \( \hat{\mu}_r \), and covariance matrices \( \hat{\Sigma}_r, r = [1, 2, \ldots, C] \) if we assume the data of the \( r \)-th class in the source domain follow a \( d \)-variate Gaussian distribution. Obviously, these two estimators are defined as \( \hat{\mu}_r = \frac{1}{m_r} \sum_{y_i^s = r} x_i^s \) and \( \hat{\Sigma}_r = \frac{1}{m_r} \sum_{y_i^s = r} (x_i^s - \hat{\mu}_r)(x_i^s - \hat{\mu}_r)^T \), respectively.

3.2. Formulation

To tackle the covariate shift, we propose a shallow feature transformation based domain adaptation method. Our method consists of two main components, i.e., source domain classification and target domain classification, and relate these two parametric classifiers by assuming small perturbations on parameters [71]. Since we have no access to the labels on the target domain, it is hard to directly consider a target domain classification task as well as some conditional distribution discrepancies. To address this issue, JDA [41] exploits the pseudo target labels as supervision signals for feature transformation learning. This strategy has proven to work well for unsupervised domain adaptation and has been re-utilized by later works [74, 36]. Thus, we also consider such a discriminative component to deal with pseudo-labeled target instances.

Inspired by the popular supervised adaptive feature reduction method [12], we first introduce a feature transformation matrix \( W \in \mathbb{R}^{d \times d} \) for the labeled source data points and expect the following objective with respect to \( W \) to be minimized as much as possible.

\[
\min_W \frac{\sum_r \sum_{y_i^s = r} d_W(x_i^s, \hat{\mu}_r)}{\sum_i d_W(x_i^s, 0)} \quad \text{s.t.} \quad W(S_w^T + S_t^T + \sum_r \beta_r \Delta_r \Delta_r^T + \lambda I)W^T = I,
\]

where \( d_W(x, x') = \|Wx - Wx'\|_2^2 \), and \( \hat{\mu}_s = 0 \) denotes the overall mean in the source domain. Furthermore, we can rewrite the objective above as

\[
\min_W \frac{\text{trace}(WS_wW^T)}{\text{trace}(WS_tW^T)},
\]

where \( S_w^T = \sum_i (x_i^s - \hat{\mu}_r^s)(x_i^s - \hat{\mu}_r^s)^T = \sum_{r=1}^C m_r \hat{\Sigma}_r \) and \( S_t^T = \sum_i (x_i^t)^T = S_w^T + \sum_r m_r(\hat{\mu}_r^s - \hat{\mu}_r^t)(\hat{\mu}_r^s - \hat{\mu}_r^t)^T \). are also well known as the within-class scatter and the total scatter matrices in the source domain, respectively.

The only information we know about the target domain is that it shares the same classes with the source domain, therefore, we also expect that data points from the target domain are well separated as those from the source domain,

\[
\min_{W, \Delta_r, \hat{\gamma}^r} \sum_r \sum_{y_i^t = r} d_W(x_i^t, \hat{\mu}_r^t + \Delta_r) + \sum_i d_W(x_i^t, 0),
\]

Since no labels are available for the target domain, we also need to estimate the optimal pseudo label \( \hat{\gamma}^r \) for each target instance and the target means \( \hat{\mu}_r^t, 1 \leq r \leq C \).

For simplicity, we follow the popular framework to learn a unique projection \( W \) for two different domains. Different from previous works [41, 74, 36] that generate pseudo target labels via merely exploiting the source data points, we aim to obtain a self-training target classifier and assume that the parameters are shifted from the parameters in the source classifier [71]. Concretely, we adopt the nearest centroid classifier and force the target centroids are close to their corresponding source centroids in Fig. 1, that is to say, we want to minimize the following objective function:

\[
\min_{W, \Delta_r, \hat{\gamma}^r} \frac{\sum_r \sum_{y_i^t = r} d_W(x_i^t, \hat{\mu}_r^t)}{\sum_i d_W(x_i^t, 0)} + \sum_i \beta_i \|W \Delta_i\|_2^2 + \sum_i d_W(x_i^t, 0)
\]

where \( \Delta_r \in \mathbb{R}^{d \times 1}, 1 \leq r \leq C \) are the the so-called perturbation variables and \( \beta_i \) are the trade-off parameters which rely on the cluster size of each class.

For the sake of simple optimization, we reformulate these three terms in one trace-ratio objective and obtain a relaxed ratio-trace objective in the following,

\[
\min_{W, \Delta_r, \hat{\gamma}^r} \frac{\text{trace}(WS_wW^T)}{\text{trace}(WS_tW^T)},
\]

\[
\min_{W, \Delta_r, \hat{\gamma}^r} \frac{\text{trace}(WS_wW^T + S_t^T + \sum_r \beta_r \Delta_r \Delta_r^T + \lambda I)W^T)}{\text{trace}(S_w^T + S_t^T)W^T},
\]

where \( S_w^T = \sum_i (x_i^s - \hat{\mu}^s_i - \Delta_{\hat{\gamma}})(x_i^s - \hat{\mu}^s_i - \Delta_{\hat{\gamma}})^T \) and \( S_t^T = \sum_i (x_i^t)^T \) are the corresponding scatter variances in the target domain. In fact, to avoid a solution where all
the rows in $W$ are identical [46], we impose an orthonormal constraint $W(S_{\theta}^T + S_{\theta})W^T = I$ on the projection $W$. Under this constraint, Fukunaga [16] showed that Eq. (5) is essentially identical to Eq. (6). Besides, we follow previous works [41, 74, 36, 51] in this literature and impose a regularization parameter $\lambda$ to guarantee the optimization problem to be well-defined.

3.3. Optimization

To optimize the above objective in Eq. (6), we exploit a popular alternating optimization scheme. In the following, we provide the solutions to each sub-problem.

$W$-step: Once we fix these two other groups of parameters $\{\Delta_l^t\}_{l=1}^{C}$ and $\{\hat{g}_i^t\}_{i=1}^{nt}$, the objective function w.r.t. $W$ becomes a classical ratio-trace optimization problem:

$$\min_{W \in \mathbb{R}^{d \times d}} \text{trace} \{(WS_tW^T)^{-1}(WS_nW^T)\}, \quad (7)$$

where $S_t = S_{t}^T + S_{s}^T$ and $S_n = S_{n}^T + S_{n}^T$ are two $d \times d$ scatter variance matrices. Interestingly, this problem can be efficiently solved by generalized eigenvalue decomposition (GEVD) $S_t w_a = \gamma_a (S_n + \lambda I) w_a$, where $\gamma_a$ is the $a$-th smallest generalized eigenvalue. The matrix $W$ is then constituted of the corresponding eigenvectors $w_a \in \mathbb{R}^{d \times 1}$, $1 \leq a \leq k$ as rows.

$Y^t$-step: Once we obtain the domain-invariant projection $W$ and perturbation variables $\{\Delta_l^t\}_{l=1}^{C}$, the objective function w.r.t. $\{\hat{g}_i^t\}_{i=1}^{nt}$ has the form

$$\min_{\{\hat{g}_i^t\}_{i=1}^{nt}} \text{trace} \{(WS_tW^T)^{-1}(WS_nW^T)\}$$

$$= \sum_{i=1}^{nt} \min_{\hat{g}_i^t \in [1, C]} \{(x_i^T - \hat{\mu}_{\hat{g}_i^t} - \Delta_l^t)^T S_p (x_i^T - \hat{\mu}_{\hat{g}_i^t} - \Delta_l^t)\}, \quad (8)$$

$$= \sum_{i=1}^{nt} \{(x_i^T S_p x_i^T) + \max_{\hat{g}_i^t \in [1, C]} (h_{\hat{g}_i^t} x_i^T - b_{\hat{g}_i^t})\},$$

where $S_p = W^T (WS_tW^T)^{-1} W$ is a positive definite matrix, and the parameters in the classification function $f_r(x_i^T) = h_r x_i^T - b_r$ of the $r$-th class are $h_r = 2(\hat{\mu}_r + \Delta_r)S_p \in \mathbb{R}^{1 \times d}$ and $b_r = -(\hat{\mu}_r + \Delta_r)^T S_p (\hat{\mu}_r + \Delta_r)$, respectively. That is to say, we can estimate each the pseudo target label $\hat{y}^t_r$ independently via the following rule

$$\hat{y}^t_r = \arg \max_{r \in [1, C]} h_r x_i^T - b_r. \quad (9)$$

$\Delta$-step: Once we get the domain-invariant projection $W$ and pseudo target labels $\{\hat{y}_i^t\}_{i=1}^{nt}$, the objective function w.r.t. each perturbation variable $\Delta_l$ can be written as

$$\min_{\Delta_l} \text{trace} \{S_p \left( \sum_{\hat{g}_i^t \in [1, C]} (x_i^T - \hat{\mu}_{\hat{g}_i^t} - \Delta_l)^T (\Delta_l^t + \beta_l \Delta_l) \right) \},$$

$$\Rightarrow \min_{\Delta_l} \text{trace} \{S_p ((\beta_l + n_r) \Delta_l - 2 \sum_{\hat{g}_i^t \in [1, C]} (x_i^T - \hat{\mu}_{\hat{g}_i^t})^T) \},$$

$$\Rightarrow \min_{\Delta_l} (\Delta_l (\sum_{\hat{g}_i^t \in [1, C]} x_i^T - n_r \hat{\mu}_r) - 2 \sum_{\hat{g}_i^t \in [1, C]} (x_i^T - \hat{\mu}_{\hat{g}_i^t})^T) S_p (\Delta_l - \sum_{\hat{g}_i^t \in [1, C]} x_i^T - n_r \hat{\mu}_r),$$

where $n_r$ is the size of the $r$-th class in the target domain. Since $S_p$ is easily proven to be a positive definite matrix, $i.e., \forall x \in \mathbb{R}^{d \times 1}, x^T S_p x \geq 0$. Thus, the optimal variable $\Delta_l$ can be obtained via the following equation

$$\Delta_l = (\sum_{\hat{g}_i^t \in [1, C]} x_i^T - n_r \hat{\mu}_r)/\beta_l + n_r, \quad r \in [1, C], \quad (11)$$

and

$$\Delta_l + \tilde{\mu}_r + \Delta_l = (\beta_l \mu_r + \sum_{\hat{g}_i^t \in [1, C]} x_i^T)/\beta_r + n_r. \quad (12)$$

Carefully checking the term above, we can easily discover that the learned perturbations indeed place the optimal target centroids/ prototypes in the routines between source class means and pseudo target class means. Besides, when the value $\beta_r/n_r$ becomes larger, the learned target centroids are much closer to their corresponding source class means.

Towards this end, we have provided three closed-form solutions in Eq. (7), Eq. (9), and Eq. (11) for each subproblem, and the complete algorithm is summarized in Algorithm 1.

**Algorithm 1** Unsupervised Domain Adaptation with Minimum Centroid Shift (MCS)

**Input:** Source domain information $\{m_r, \hat{\mu}_r, \Sigma_r\}$ and target domain $\{x_i^t\}_{i=1}^{nt}$, subspace dimensionality $k$, parameters $\lambda$ and $\alpha$, inner/outer maximum iterations $T_i = 5$ and $T_o = 10$.

**Output:** Feature transformation matrix $W \in \mathbb{R}^{k \times d}$, perturbation variables $\{\Delta_r\}_{r=1}^{C}$, and target labels $\{\hat{y}_i^t\}_{i=1}^{nt}$.

1. Compute the scatter matrices $S_{\theta}^t, S_{\theta}^T, S_{\theta}^s$;
2. Initialize $S_n^t$ and $\Delta_r$ as 0, and calculate $W$ via Eq. (7);
3. Estimate $\{\hat{y}_i^t\}_{i=1}^{nt}$ using $W$ and $\{\hat{y}_i^t\}_{i=1}^{nt}$, and update $S_n^t$;
4. Compute the size of the $r$-th target class $l_r$ and let $\beta_r = \alpha * l_r$;
5. while not converge iter $\leq T_i$, do
6. while not converge and iter $\leq T_i$, do
7. Update perturbation variables $\{\Delta_r\}_{r=1}^{C}$ via Eq. (11);
8. Update pseudo target labels $\{\hat{y}_i^t\}_{i=1}^{nt}$ via Eq. (9);
9. end while
10. Update $S_n^t$ and solve the GEVD problem in Eq. (7) for $W$.
11. end while

3.4. Convergence and Time Complexity

It is obvious that each solution above (i.e., Eqs. (7), (9), (11)) monotonically decreases the overall objective function trace \{(WS_tW^T)^{-1}(WS_nW^T)\} in each iteration and the objective value is always larger than zero, that is to say, the proposed iterative algorithm in Algorithm 1 converges to the local minimizer after certain iterations.

Regarding the computation complexity, the GEVD step occupies $O(k d^2)$, other matrix multiplications occupy $O(n_t d^2 + n_t d k)$, and the $Y^t$ step occupies $O(C n_t k^2 + C n_t d + k^3)$. In summary, the overall complexity of our method is $O(T_i k d^2 + T_o C n_t d)$. When compared with massive previous methods [41, 74, 36, 67] with $O(n^2)$ complexity, our method is more favorable by large-scale datasets. Specifically, for large-scale digit datasets like SVHN and MNIST, previous methods (eg., [41, 74, 36, 67]) are not flexible to the limited memory.
4. Experiment

In this section, we conduct extensive experiments to evaluate the effectiveness of the proposed approaches for unsupervised cross-domain image recognition problems, including vanilla domain adaptation, multi-source domain adaptation and domain generalization where the target domain is totally unknown in the training phase.

4.1. Unsupervised Domain Adaptation (UDA)

4.1.1 Datasets and Settings

The Office31 [55] dataset includes images of 31 objects taken from 3 domains, i.e., Amazon (A), images downloaded from the online web merchants), DSLR (D, high-resolution images captured by a digital SLR camera), and Webcam (W, low-resolution images recorded by a web camera). They contain 2,817, 498 and 795 images, respectively.

The Office-Caltech dataset consists of images from 10 overlapping object classes between Office31 and Caltech256 [25], and these domains include 958, 157, 295 and 1,123 samples for A, D, W and Caltech (C), respectively.

The Office-Home [66] dataset is a new benchmark that contains 4 domains, with each domain containing 65 kinds of everyday objects, i.e., Art (Ar), artistic depictions of objects), Clipart (Cl, clipart images), Product (Pr, objects without a background) and Real-World (Re, objects captured with a regular camera). Besides, they contain 2,421, 4,365, 4,428 and 4,357 samples, respectively.

**Baseline methods.** We compare the proposed method with several state-of-the-art unsupervised domain adaptation approaches that can be roughly divided into two main categories, shallow feature transformation approaches, and deep domain adaptation approaches. 1NN is a basic method that is trained on the raw source domain without any feature transformation. Shallow UDA methods mainly include SA [15], JDA [41], CORAL [59], Invariant Latent Space (ILS) [27], JGSA [74], LDA-inspired Domain Adaptation (LDADA) [45], and DICE [36]. There are also some popular baseline methods, including ATI [4], PUnDA [20], MEDA [67], and GAKT [13].

Regarding deep end-to-end UDA approaches, we collect the reported accuracies from recent studies that share the same protocol with our method. Some representative deep methods are listed in the following, DAN [39], RevGrad [17], DRCN [19], RTN-res [43], ADDA [64], JAN-A [44], GTA [56], and CDAN+E [40].

Note that A→B indicates that A is the source domain and B is the target domain. We evaluate different UDA methods in terms of classification accuracy (%) on the target.

### 4.1.2 Results on the Office31 dataset

We follow the full protocol that has been widely adopted in previous studies [44, 36, 56] by using the entire labeled data in the source domain and unlabeled data in the target domain. We further exploit the AlexNet-FC7 features [59] fine-tuned on the source domain, making it fair to be compared with deep UDA methods via AlexNet.

As shown in Table 1, our method outperforms all deep methods and other shallow counterparts in terms of the average accuracy. Firstly, for the small and easy tasks D→W, all the UDA methods achieve promising results, and our method performs worse than several shallow methods (e.g., JGSA [74]). Secondly, our method significantly outperforms all the shallow methods and is competitive to state-of-the-art deep method JAN-A [44] for some relatively challenging adaptation tasks, including A→D and A→W. Thirdly, when a small source domain (i.e., D and W) is adapted to a large target domain A, our method obviously ranks the first and second among all the methods.

Generally speaking, our method outperforms several state-of-the-art shallow methods (i.e., DICE [36] and MEDA [67]) by a large margin and achieves quite competitive results to the state-of-the-art deep UDA method. Note that, even I2I-Adapt [49] explores a more powerful ResNet-34 model, our method still performs much better than it.

### 4.1.3 Results on the Office-Caltech dataset

We explore two kinds of deep features [27] produced by different full convolution layers in VGG-net and follow the sampling protocol [21, 27, 20] by using few labeled data in the source domain. Concretely, we randomly select 8 instances per class for domain D and 20 instances per class for other domains (i.e., A, C and W) as final sources.

As can be seen from Table 2, JGSA [74], PUnDA [20] and DICE [36] are three best performing methods among
Figure 2. Domain adaptation performance accuracy (%) on the Office31 (A→D and A→W) and Office-Home (Ar→Cl and Ar→Pr) datasets w.r.t. dimensionality $k$, parameters $\lambda$, $\alpha$ and the number of iterations.

Table 2. Accuracy (%) on Office-Caltech using the VGG-FC$_6$ (above) and VGG-FC$_7$ (below) features with the evaluation setup of [21, 20]. The best (bold red), the second best (red).

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>C</th>
<th>D</th>
<th>W</th>
<th>A</th>
<th>D</th>
<th>W</th>
<th>A</th>
<th>D</th>
<th>W</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>INN</td>
<td></td>
<td>70.1</td>
<td>81.9</td>
<td>76.0</td>
<td>69.0</td>
<td>81.9</td>
<td>77.7</td>
<td>69.0</td>
<td>81.9</td>
<td>77.7</td>
<td>73.2</td>
</tr>
<tr>
<td>SA</td>
<td></td>
<td>77.1</td>
<td>86.9</td>
<td>76.0</td>
<td>70.0</td>
<td>86.9</td>
<td>77.7</td>
<td>70.0</td>
<td>86.9</td>
<td>77.7</td>
<td>80.9</td>
</tr>
<tr>
<td>CORAL</td>
<td></td>
<td>79.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>85.7</td>
</tr>
<tr>
<td>JGSA</td>
<td></td>
<td>79.9</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>85.7</td>
</tr>
<tr>
<td>JDA</td>
<td></td>
<td>82.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>85.7</td>
</tr>
<tr>
<td>LDADA</td>
<td></td>
<td>82.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>85.7</td>
</tr>
<tr>
<td>DICE</td>
<td></td>
<td>83.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>85.7</td>
</tr>
<tr>
<td>MCS(ours)</td>
<td>87.1</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>76.0</td>
<td>94.1</td>
<td>82.0</td>
<td>85.7</td>
<td>89.8</td>
</tr>
</tbody>
</table>

(a) Dimensionality $k$

(b) Parameter $\lambda$

(c) Parameter $\alpha$

(d) # Iterations

Note that, most previous UDA methods except LDADA [45] favor VGG-FC$_6$ features because they may not be much more discriminative than VGG-FC$_7$ features, making them suitable for general feature transformation based approaches. By contrast, MCS is somewhat robust to the feature type, since the dropping rate of accuracy from VGG-FC$_6$ to VGG-FC$_7$ is relatively small.

4.1.4 Results on the Office-Home dataset

For the Office-Home dataset, we explore PyTorch to fine-tune the ResNet-50 model [26] pre-trained on the ImageNet dataset with the labeled source images and extract the features after the 5-th pooling layer for each task.

LDADA [45] performs quite worse with a 2.6 average accuracy, thus, we do not report its results in Table 3. Obviously, MCS again performs the best among all the methods while CDAN+E [40] and DICE [36] achieves the second and third best performance. Carefully comparing our MCS with CDAN+E and DICE, we find that MCS consistently performs better than CDAN+E and DICE in 7 out of 12 tasks. Checking the results of MCS again, we find that the results are somewhat low when Cl is the source domain. This may be because our method adopts the nearest centroid classifier which may ignore the diversity of each class in the source domain like the ‘Clipart’ subset.

Table 3. Accuracy (%) on Office-Home with ResNet-50 features and the evaluation setup [66, 36]. [* using VGG-FC$_6$ features]

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Cl</th>
<th>Pr</th>
<th>Re</th>
<th>Cl</th>
<th>Pr</th>
<th>Re</th>
<th>Cl</th>
<th>Pr</th>
<th>Re</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>INN</td>
<td></td>
<td>44.1</td>
<td>61.8</td>
<td>69.2</td>
<td>49.0</td>
<td>60.6</td>
<td>63.2</td>
<td>51.6</td>
<td>63.1</td>
<td>70.4</td>
<td>58.5</td>
</tr>
<tr>
<td>SA</td>
<td></td>
<td>44.8</td>
<td>65.5</td>
<td>70.6</td>
<td>48.8</td>
<td>61.5</td>
<td>64.1</td>
<td>50.1</td>
<td>63.2</td>
<td>70.4</td>
<td>58.5</td>
</tr>
<tr>
<td>JDA</td>
<td></td>
<td>46.3</td>
<td>66.0</td>
<td>69.1</td>
<td>47.1</td>
<td>63.4</td>
<td>63.3</td>
<td>48.2</td>
<td>64.0</td>
<td>70.8</td>
<td>58.7</td>
</tr>
<tr>
<td>JGSA</td>
<td></td>
<td>47.1</td>
<td>67.3</td>
<td>74.5</td>
<td>52.3</td>
<td>63.6</td>
<td>66.9</td>
<td>51.0</td>
<td>61.9</td>
<td>72.6</td>
<td>60.4</td>
</tr>
<tr>
<td>JDA</td>
<td></td>
<td>48.7</td>
<td>66.4</td>
<td>69.1</td>
<td>48.7</td>
<td>64.0</td>
<td>69.1</td>
<td>50.6</td>
<td>67.9</td>
<td>75.1</td>
<td>64.0</td>
</tr>
<tr>
<td>ILS</td>
<td></td>
<td>46.7</td>
<td>64.3</td>
<td>69.7</td>
<td>44.3</td>
<td>60.9</td>
<td>64.1</td>
<td>47.9</td>
<td>62.7</td>
<td>70.4</td>
<td>61.4</td>
</tr>
<tr>
<td>DICE</td>
<td></td>
<td>53.2</td>
<td>72.4</td>
<td>74.5</td>
<td>56.7</td>
<td>70.1</td>
<td>69.1</td>
<td>58.9</td>
<td>51.5</td>
<td>77.0</td>
<td>66.5</td>
</tr>
<tr>
<td>GAKT [33]</td>
<td>34.5</td>
<td>43.6</td>
<td>53.5</td>
<td>36.1</td>
<td>52.3</td>
<td>53.5</td>
<td>31.6</td>
<td>40.6</td>
<td>61.4</td>
<td>45.6</td>
<td></td>
</tr>
<tr>
<td>CDAN[10]</td>
<td>49.0</td>
<td>69.3</td>
<td>74.5</td>
<td>54.4</td>
<td>66.0</td>
<td>68.4</td>
<td>55.6</td>
<td>68.3</td>
<td>75.9</td>
<td>59.4</td>
<td></td>
</tr>
<tr>
<td>CDAN+E[40]</td>
<td>50.7</td>
<td>70.6</td>
<td>70.6</td>
<td>57.0</td>
<td>70.0</td>
<td>70.0</td>
<td>57.4</td>
<td>50.9</td>
<td>73.3</td>
<td>76.9</td>
<td></td>
</tr>
<tr>
<td>MCS(ours)</td>
<td>55.9</td>
<td>73.8</td>
<td>79.0</td>
<td>57.5</td>
<td>69.1</td>
<td>71.3</td>
<td>55.4</td>
<td>50.3</td>
<td>78.2</td>
<td>67.9</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Accuracy improvement (%) of RevGrad [17] and DICE, we find that MCS consistently performs better than CDAN+E and DICE in 7 out of 12 tasks. Checking the results of MCS again, we find that the results are somewhat low when Cl is the source domain. This may be because our method adopts the nearest centroid classifier which may ignore the diversity of each class in the source domain like the ‘Clipart’ subset.

4.1.5 Parameter Sensitivity Analysis

To investigate the sensitivity of dimensionality $k$, parameters $\lambda$, $\alpha$ and $\beta$ in our method, we exploit four UDA tasks on the Office31 and Office-Home datasets, i.e., A→D, A→W, Ar→Cl, and Ar→Pr. We respectively display all these parameter sensitivity analysis results in Figures 2(a)~2(c), with a wide range of $d \in [100, 110, \ldots, 200]$ and $\lambda, \alpha \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5\}$. 

![Image](https://example.com/image.png)
Table 4. Domain generalization performance accuracy (%) on the VLCS dataset with the evaluation setup of CIDG [35].

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>INN</th>
<th>KPCA</th>
<th>DICA</th>
<th>Undo-bias</th>
<th>SCA</th>
<th>CIDG</th>
<th>MCS$_5$(ours)</th>
<th>MCS$_5$(ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L,C,S</td>
<td>V</td>
<td>53.27±1.52</td>
<td>58.62±1.44</td>
<td>58.29±1.51</td>
<td>57.73±1.02</td>
<td>57.48±1.78</td>
<td>65.60±0.52</td>
<td>65.54±1.26</td>
<td>65.17±0.44</td>
</tr>
<tr>
<td>V,C,S</td>
<td>L</td>
<td>50.35±0.94</td>
<td>53.80±1.78</td>
<td>50.35±1.45</td>
<td>52.07±0.86</td>
<td>52.07±0.86</td>
<td>60.43±1.57</td>
<td>57.42±0.48</td>
<td>59.95±0.40</td>
</tr>
<tr>
<td>V,L,S</td>
<td>C</td>
<td>76.82±1.56</td>
<td>85.84±1.64</td>
<td>71.32±4.13</td>
<td>82.18±1.77</td>
<td>70.39±1.42</td>
<td>91.12±1.62</td>
<td>92.40±0.62</td>
<td>89.90±0.37</td>
</tr>
<tr>
<td>V,L,C</td>
<td>S</td>
<td>51.78±2.07</td>
<td>53.23±0.62</td>
<td>54.97±0.61</td>
<td>55.02±2.53</td>
<td>54.46±2.71</td>
<td>60.85±0.95</td>
<td>62.07±0.82</td>
<td>65.16±0.68</td>
</tr>
</tbody>
</table>

Table 5. Average accuracy (%) on the Office31 dataset via ResNet-50 with the evaluation setup of [44, 53, 40].

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Methods</th>
<th>Accuracy</th>
<th>Methods</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>before 2018</td>
<td></td>
<td>after 2018</td>
<td></td>
</tr>
<tr>
<td>L,C</td>
<td>V</td>
<td>GTA</td>
<td>87.5</td>
<td>SimNet</td>
<td>87.8</td>
</tr>
<tr>
<td>V,C</td>
<td>L</td>
<td>iCAN</td>
<td>89.7</td>
<td>CDAN</td>
<td>85.3</td>
</tr>
<tr>
<td>V,L</td>
<td>S</td>
<td>TEM</td>
<td>88.5</td>
<td>CDAN+E</td>
<td>87.9</td>
</tr>
<tr>
<td>V,L,C</td>
<td>S</td>
<td>MNC</td>
<td>86.5</td>
<td>MCS(ours)</td>
<td>87.5</td>
</tr>
<tr>
<td>V,C,S</td>
<td>L</td>
<td>RevGrad</td>
<td>88.5</td>
<td>+MCS(ours)</td>
<td>87.8</td>
</tr>
<tr>
<td>V,L,C</td>
<td>S</td>
<td>ADDA</td>
<td>88.0</td>
<td>+MCS(ours)</td>
<td>87.8</td>
</tr>
<tr>
<td>V,L,C</td>
<td>S</td>
<td>ADDA</td>
<td>88.0</td>
<td>+MCS(ours)</td>
<td>87.8</td>
</tr>
</tbody>
</table>

It can be observed that MCS is robust with regard to different values of $k$ in Figure 2(a). For high-dimensional features via AlexNet and ResNet-50, $k = 150$ is an optimal choice. When $\lambda \rightarrow 0$, the optimization problem is ill-defined. When $\lambda \rightarrow \infty$, the minimum centroid shift is not performed, and MCS cannot construct robust representation for cross-domain classification. Concerning the sensitivity of regularization parameter $\lambda$, we plot the accuracies in Figure 2(b), which indicates that $\lambda \in [0.1, 1.0]$ is an optimal choice.

Theoretically, smaller values of $\alpha$ can make self-learning in the target domain (i.e., using the class mean itself) more important in MCS. Observing the accuracy in Figure 2(c), we can discover that $\alpha \in [0.1, 1.0]$ is an optimal choice under which both the source centroids and the pseudo target class means are effectively considered. As expected, larger values of $\alpha$ can degenerate the adaptation performance due to the essential heterogeneity.

Finally, we plot the accuracy w.r.t. the number of iterations in Figure 2(d) where we consider the inner loop as well as the outer loop. Obviously, the accuracy always grows gradually until convergence within 4 outer iterations.

4.1.6 Combination with Deep UDA Methods

The comparisons with recent state-of-the-art shallow methods demonstrate the effectiveness of our method. However, we still doubt that whether deep adaptation methods can be enhanced with our simple method. Here we investigate this problem by exploiting two popular deep domain adaptation methods, i.e., RevGrad [17] and DAN [39] for cross-domain object recognition and GTA [56] for cross-domain digit recognition. Regarding these deep UDA methods [17, 39, 56], we extract the intermediate features for both domains and take them as input for our method MCS.

As can be seen from Figure 3, once MCS is integrated with deep UDA methods, i.e., RevGrad and DAN on the Office31 and Office-Home datasets, the results of most cases are improved in 17 and 15 out of 18 tasks, respectively. Checking the average accuracy (%) in Table 5, with the help of MCS, both the results of RevGrad and DAN are significantly improved, from 81.8 to 87.8 and 81.7 to 87.4. For generic cross-domain object recognition, we explore the pre-trained ResNet-50 model as our basic network and compare it with recent state-of-the-art methods [56, 53, 75, 29, 40] in Table 5. Specifically, RevGrad+MCS obtains the best average accuracy, which even beats the best result reported in CDAN+E [40]. It can be expected that MCS can achieve much higher results when combined with better adaptation methods like TEM.

Regarding the digit recognition task, we explore 3 popular digit datasets, i.e., MNIST [33], USPS [28] and SVHN [50]. Specifically, we follow the standard protocol that uses the corresponding entire training sets as domains, and the testing sets for validation. All the images are rescaled to $32 \times 32$. In fact, we exploit the Source-only and GTA [56] models as a baseline network, and compare them with DRCN [19], PixelDA [2], ADDA [64], and SBADA-GAN [54] in Table 6. We easily observe that Source-only+MCS and GTA+MCS are much better than Source-only and GTA, respectively. Compared with recent state-of-the-art results [19, 2, 64, 54], GTA+MCS obtains the best results. Besides, MCS beats DICE$_{MV-SVM}$ with the features provide in [36]. Regarding the per-class classification accuracy (%) in M→U and U→M, MCS scores 97.5 and 98.2 that are higher than 96.4 and 95.6 in SimNet [53].

4.2. Domain Generalization
Here we conduct experiments on a real world image classification dataset VLCS like [35], including four domains: VOC2007 (V), LabelMe (L), Caltech-101 (C) and SUN09 (S). These datasets share 5 object categories: bird, car, chair, dog, and person. The datasets from source domains are split into two parts: 70% for training and 30% for validation, following [35]. The whole target domain is used for testing. We repeat the random selection 10 times and report the mean classification accuracy and standard deviation.

For the domain generalization task, we compute all the total scatter matrices for each source domain and obtain $S_i$ by adding them up. Similarly, we compute all the within-class scatter matrices for each source domain and obtain $S_w$ via adding them up, where each $\Delta_r$ is a fixed variable by measuring the difference between different source class means. To this end, we obtain the optimal projection matrix $W$ via Eq. (7). Note that there is no need to learn $Y_i$ and $A_r$, making this problem to be a one-pass algorithm. For $MCS_A$, each class has multiple centroids instead of one, then we estimate $Y_i$ via Eq. (9). $MCS_B$ needs to know all the source instances instead of the estimated distributional parameters $\hat{\mu}_r, \hat{\Sigma}_r$, it follows $DICE_2$ with training a linear SVM classifier on the projected source instances.

We compare $MCS_A$ and $MCS_B$ with KPCA [57], DICA [48]. Undobias [31], SCA [18] and CIDG [35]. As can be seen from Table 4, both $MCS_A$ and $MCS_B$ win 4 out of 10 tasks, while the best performing baseline CIDG merely win 2 out of 10 tasks. Besides, the standard deviations are much smaller than CIDG, which indicates that our methods are somewhat robust.

### 4.3. Multi-source Domain Adaptation

To address the multi-source domain adaptation task, we develop two different methods $MCS_C$ and $MCS_D$. Specifically, $MCS_C$ naively combines multiple source domains into one source domain and becomes a vanilla domain adaptation task. Similar to $MCS_A$, $MCS_D$ sums up all the within-class scatter matrices and total matrices and adopt them in Eq. (7), and assumes the optimal centroid on the target to be close to the mean of different source class means.

We utilize the PIE dataset that includes facial images of 68 people with various pose, illumination, and expression changes. Following [41, 37], we select 5 out of 13 poses, i.e., C05 (left), C07 (upward), C09 (downward), C27 (frontal) and C29 (right). These images are cropped to the size $32 \times 32$, constituting 1,024-dimensional features. We compare $MCS_C$ and $MCS_D$ with StP [37] in Table 7. Generally, $MCS_C$ and $MCS_D$ achieve the best results in 7 and 19 out of 27 tasks, respectively. They significantly outperform StP for almost all tasks except (C05, C09)→C29, and $MCS_D$ performs slightly better than $MCS_C$. This may be because that $MCS_C$ ignores the size of different source domains, which mainly relies on the larger source.

In addition, we consider the Office31 (AlexNet features [59]) and Office-Caltech (DeCAF features [74, 36]) datasets with the protocols in DLD [47] and StP [37], respectively. As can be seen from Table 8, we discover that $MCS_D$ is always superior to $MCS_C$ and both of them outperform other methods in terms of the average accuracy.

### 5. Conclusion

In this paper, we have proposed a simple, efficient, yet effective approach for visual domain adaptation. The key idea is to seek a subspace where the target centroids are moderately shifted from those in the source domain. Then a unified objective is designed to derive several sub-problems with closed-form solutions to subspace discovery and target pseudo-labeling, and the alternating minimization algorithm is guaranteed to converge. Note that, our method only acquires some class-wise distribution estimators from source data as distant supervisions, hence it also provides a privacy-preserving way for source domain data. Besides, it can be easily extended for domain generalization and multi-source domain adaptation problems. Extensive experiments on several visual benchmarks demonstrate the superiority of the proposed method over many existing state-of-the-art methods. Generally, our method is impressively simple and efficient, hence, it can be considered as a promising baseline for domain adaptation and generalization tasks.
Supplementary Material for “Distant Supervised Centroid Shift: A Simple and Efficient Approach to Visual Domain Adaptation”

Jian Liang, Ran He, Zhenan Sun and Tieniu Tan
liangjian92@gmail.com

1. Details of Domain Generalization

Regarding the domain generalization task, we adopt the same objective function as unsupervised domain adaptation, the only difference lies in that we know the labels and centroids in the so-called ‘target’ domain, i.e., we merely need to learn the projection matrix $W$ for $\theta$ source domains.

$$
\min_W \{ \frac{\text{trace}(W(S_w + \beta \sum_{r,i,j;i,j} \Delta_{r,i,j}^T \Delta_{r,i,j}^T + \lambda I)W^T)}{W S_i W^T} \},
$$

where $S_t = \sum_i S_i^T$ and $S_w = \sum_i S_i^w$, and $S_i^T$ and $S_i^w$ are the total and within-class scatter variance matrices of the $i$-th source domain, respectively. $\Delta_{r,i,j} = \hat{\mu}_{r,i} - \hat{\mu}_{r,j}$ denotes the distance of the $r$-th class mean between the $i$-th source domain and the $j$-th source domain, and $\hat{\mu}_{r,i}$ is the $r$-th class mean (centroid) in the $i$-th source domain.

Once the optimal projection $W$ is obtained via the generalized eigenvalue decomposition problem, we propose two methods $\text{MCS}_A$ and $\text{MCS}_B$ to infer the labels of the target instances that are not accessed in the training phase. Specifically, the classification rule of $\text{MCS}_A$ with $\theta \times C$ centroids $\{\hat{\mu}_i\}_{i=1}^{\theta C}$ is shown in the following,

$$
y = \text{mod}(\arg\min_{\theta \in [1, \theta C]} \{ (x_i^T - \hat{\mu}_i) S_p(x_i^T - \hat{\mu}_i) \}, C). \tag{2}
$$

Note that, $y = 0 \rightarrow y = C$ is executed to calculate the cross-domain classification accuracy.

Unlike $\text{MCS}_A$ that does not need to access the source domain data, $\text{MCS}_B$ utilizes all the projected source domain data to train a linear SVM classifier. For each target instance, we first obtain the projected feature and predict the label via the learned SVM classifier.

2. Details of Multi-source Domain Adaptation

Given $\theta$ source domains, we also develop two methods $\text{MCS}_C$ and $\text{MCS}_D$ to infer the labels of the target instances that are utilized in the training phase. Specifically, $\text{MCS}_C$ naively combines multiple source domains into one source domain and becomes a vanilla domain adaptation task. Then we obtain the labels of the target instances via Algorithm 1. Similar to $\text{MCS}_A$, $\text{MCS}_D$ sums up all the within-class scatter matrices and total matrices and adopt them, and assumes the optimal centroid on the target to be close to the mean of different source class means. The detailed objective function is shown below:

$$
\min_W \{ \frac{\text{trace}(W(S_w + \sum_r \beta_r \Delta_r^T \Delta_r + \lambda I)W^T)}{W S_i W^T} \},
$$

where $S_w = \sum_i (x_i^T - \hat{\mu}_{y_i} - \Delta_{y_i}) (x_i^T - \hat{\mu}_{y_i} - \Delta_{y_i})^T + \sum_i S_i^w$, and $S_i = \sum_i (x_i^T)^T S_i^T + \sum_i S_i^T$, and $S_i^T$ and $S_i^w$ are the total and within-class scatter variance matrices of the $i$-th source domain, respectively. $\hat{\mu}_y = \sum_{i \neq j} \hat{\mu}_{y_i}$ denotes the $r$-th source centroid of multiple source domains, the computation of $\beta_r$ is the same as vanilla domain adaptation in Algorithm 1.

3. t-SNE Visualization Results

For visualization comparisons, we select the $A \rightarrow D$ in the Office31 dataset and the $A \rightarrow C$ task in the Office-Caltech dataset for our method and several classical shallow methods [3, 5, 6, 2]. The visualization results are shown in Fig. 1 and Fig. 2, and the source data and target data are marked as red and blue dots, respectively. As can be seen from Fig. 1, JGSA [6] and DICE [2] performs better than JDA [3] and CORAL [5] since each class are well separated. For our method MCS, the source domain and the target domain are not well aligned since we assume the centroid shift, however, the classes are still well separated. Compared with other methods, MCS owns the best accuracy among these
Figure 1. t-SNE visualizations of sought subspaces by different unsupervised domain adaptation methods for the A→D task on the Office31 dataset with the evaluation setup of [2] via AlexNet-FC7 features. [# source domain A: 2,817, # target domain D: 498.]

Figure 2. t-SNE visualizations of sought subspaces by different unsupervised domain adaptation methods for the A→C task on the Office-Caltech dataset with the evaluation setup of [1] via VGG-FC6 features. [# source domain A: 200, # target domain C: 1,193.]
Table 1. Objective function value on the Office31 (A→D and A→W) and Office-Home (Ar→Cl and Ar→Pr) datasets w.r.t. the number of iterations. $T_o$ indicates the number of outer iteration and $T_i$ indicates the number of inner iteration.

<table>
<thead>
<tr>
<th>$T_o$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-D</td>
<td>290.52</td>
<td>289.89</td>
<td>289.68</td>
<td>289.58</td>
<td>289.51</td>
<td>284.05</td>
<td>284.04</td>
<td>283.97</td>
<td>283.95</td>
<td>283.95</td>
<td>283.93</td>
<td>283.92</td>
<td>283.92</td>
<td>283.92</td>
<td>283.92</td>
<td>283.92</td>
</tr>
<tr>
<td>A-W</td>
<td>283.46</td>
<td>282.56</td>
<td>282.25</td>
<td>282.10</td>
<td>282.04</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
<td>274.55</td>
</tr>
<tr>
<td>Ar-Cl</td>
<td>127.52</td>
<td>126.07</td>
<td>125.70</td>
<td>125.54</td>
<td>125.43</td>
<td>111.19</td>
<td>111.17</td>
<td>111.17</td>
<td>111.17</td>
<td>111.17</td>
<td>111.04</td>
<td>111.03</td>
<td>111.03</td>
<td>111.03</td>
<td>111.03</td>
<td>111.02</td>
</tr>
<tr>
<td>Ar-Pr</td>
<td>122.60</td>
<td>120.52</td>
<td>120.00</td>
<td>119.69</td>
<td>119.56</td>
<td>106.37</td>
<td>106.37</td>
<td>106.37</td>
<td>106.37</td>
<td>106.37</td>
<td>106.34</td>
<td>106.34</td>
<td>106.34</td>
<td>106.34</td>
<td>106.34</td>
<td>106.34</td>
</tr>
</tbody>
</table>


In addition to the final classification results, we exploit the intermediate features extracted from GTA [4] and explore the domain adaptation task on the U→M and M→U datasets. Specifically, we provide the cross-domain digit recognition accuracy with regards to the number of epochs in Fig. 3.

Carefully looking at the figures, we can easily find that GTA+MCS always outperforms GTA. Even in the early training period, GTA performs worse, after its integration with MCS, the accuracy improves significantly. Generally, GTA+MCS always obtains stable accuracies with the increase of epochs. Taking into consideration the quite short training time MCS takes, the integration of state-of-the-art deep adaptation method with MCS is really appealing.

5. Convergence Analysis

In the section, we provide the value of proposed objective function with the increase of iterations in Table 1. As expected, the objective function monotonically decreases with the increase of iterations and quickly converges with 3 outer iterations.

References