Multimedia Tools and Applications (2021) 80:23255–23274
https://doi.org/10.1007/s11042-020-09336-0

Domain adaptation for object recognition using subspace sampling demons

Youshan Zhang[1] · **Brian D. Davison**[1]

Received: 3 November 2019 / Revised: 26 May 2020 / Accepted: 13 July 2020 /
Published online: 29 August 2020

Abstract
Manually labeling data for training machine learning models is time-consuming and expensive. Therefore, it is often necessary to apply models built in one domain to a new domain. However, existing approaches do not evaluate the quality of intermediate features that are learned in the process of transferring from the source domain to the target domain, which results in the potential for sub-optimal features. Also, transfer learning models in existing work do not provide optimal results for a new domain. In this paper, we first propose a fast subspace sampling demons (SSD) method to learn intermediate subspace features from two domains and then evaluate the quality of the learned features. To show the applicability of our model, we test our model using a synthetic dataset as well as several benchmark datasets. Extensive experiments demonstrate significant improvements in classification accuracy over the state of the art.

Keywords Domain adaptation · Object recognition · Subspace sampling

1 Introduction

Modern society produces a huge amount of data in a variety of forms, e.g., text, image, audio, and video. Industry and the research community have a great demand for automatic classification and analysis of different forms of data [3, 9, 40]. However, it is time-consuming and expensive to acquire enough labeled data to train machine learning models from scratch. Therefore, it is valuable to learn a model for a new target domain from abundant labeled samples in a different existing domain. In addition, due to the differences between different domains, termed data bias or domain shift [25], machine learning models often do not generalize well from an existing domain to a novel unlabeled domain. To address the domain shift issue, mechanisms for extracting feature representations from a continuous intermediate space (between source and target representations) have been widely

✉ Youshan Zhang
yoz217@lehigh.edu

Brian D. Davison
bdd3@lehigh.edu

[1] Computer Science and Engineering, Lehigh University, Bethlehem, PA, USA

used in many tasks such as sentiment classification [3, 7], and object recognition [12, 13]. There are also several approaches to address the domain shift problem; a prominent one is domain adaptation [12, 18, 32]. There have been efforts made in domain adaptation for both semi-supervised [2, 5, 24] and unsupervised [4, 6, 23, 41, 42] variations of domain adaptation for the object recognition problem. We have completely labeled data in the source domain in both cases. In the former case, the target domain contains a small amount of labeled data, whereas, in the latter case, the target domain is entirely unlabeled. Often the small amount of labeled target data alone is insufficient to construct a good classifier because of the domain shift. Thus, how to effectively leverage sufficient labeled source data to facilitate the usage of unlabeled target data is the key problem in domain adaptation. In this paper, we focus on unsupervised domain adaptation.

However, a critical challenge remains: to find and identify useful features that span the representation of two domains. The quality of learned features will directly affect classification accuracy. We cannot expect to train a high-quality classifier if the learned features are poor. Therefore, it is essential to define an intermediate space connecting the representation of the source domain to the target domain producing what we term a path and such a path contains all intermediate spaces.

Manifold learning is an approach to identify the intermediate features between the source and target domains via a Grassmannian manifold. Several methods exist to find such features. Gopalan et al. [13] proposed a sampling geodesic flow (SGF) method to learn the intermediate features (which reflect features as they change from those specific to the source domain to those specific to the target domain in the sub-space via the geodesic (shortest path) on a Grassmannian manifold. However, SGF has several limitations. Gong et al. [12] noted that it is difficult to choose an optimal sampling strategy. Moreover, SGF has high time complexity making sampling slow when many points are needed. They proposed a geodesic flow kernel (GFK) model to overcome the limitations of unknown sampling size in SGF. They integrated all samples along the geodesic as calculated by Gopalan et al. [13]. In prior work [43], we proposed geodesic sampling on Riemannian manifolds (GSM) and found that the SGF "geodesic" is not the true geodesic. The GSM model provided a correct way to sample the intermediate features along the correct geodesic. However, the points are sampled in an analogous linear manner, but does not generate problematic deformations as we will demonstrate in Section 5.1.

A number of efforts have been made to address the alignment of marginal distribution and conditional distribution of data in domain adaptation. Wang and Mahadevan [34] aligned the source and target domain by preserving the "neighborhood structure" of the data points. Wang et al. [36] proposed a manifold embedding distribution alignment method (based on work of Gong et al. [12]) to align both the degenerate feature transformation and the unevaluated distributions of both domains. However, none of these models explore the quality of the learned features. Others have applied deep learning models to domain adaptation [8, 14, 21, 28, 31, 32, 39]. However, these methods mainly align the distribution of source and target domains, without considering the complex multimode model structure underlying the data distribution, which leads to false alignment of data.

Our principal contributions are three-fold:

1. We are the first to propose a subspace sampling demon (SSD) approach for fast intermediate feature learning;
2. We provide a quantitative evaluation of learned features, so that we are able to select the best features for the image recognition problem. Evaluation of learned features can be of value to future research on domain adaptation.

3. To better represent the learned features and train a robust classifier, we align both marginal and conditional distributions of source and target domains.

Extensive experiments on five highly competitive benchmark image datasets show that SSD can significantly improve the classification accuracy over the state of the art.

2 Motivation

Existing methods that connect subspaces only learn features between source and target domains in subspace without examining the quality of their learned features [12, 13], which implies that the learned features may be sub-optimal, leading to reduced accuracy. In contrast, we formulate a subspace sampling demon method (focusing on shaped-based feature learning), which generates more meaningful features. Specifically, we update the transformation changes in each iteration for better feature learning in Section 4.1. We discuss the subspace sampling problem and then apply the sampling strategy to the recognition problem in Sections 2.1 and 2.2.

2.1 Subspace sampling

Given training data $X_S = \{x_i\}_{i=1}^{N_1}$ (with dimensionality $N_1 \times d$, where N_1 is number of observations, and d is the number of features), and test data $X_T = \{x_j\}_{j=1}^{N_2}$ ($N_2 \times d$), to learn features that connect X_S to X_T, we should solve the dimensionality disagreement of source and target data, since the number of examples N_1 can be different from N_2, though the number of features (d) of examples in X_S and X_T are the same. Therefore, we need to construct a low-dimensional sub-space representation of the training and test data. There are several methods for constructing low-dimensional representations (e.g., principal component analysis (PCA) [15], Laplacian eigenmaps [1], and principal geodesic analysis (PGA) [10]). In our paper, we use PCA to reduce the dimensionality of X_S and X_T as shown in Algorithm 1. By using PCA, we get the source subspace X'_S ($d \times d'$) and target subspace X'_T ($d \times d'$) domains to represent original source X_S and target X_T data, respectively. Here d' is the reduced dimensionality. The parameter d' can be tuned to get better application-specific performance, which we discuss in Section 4.3. We then aim to sample more intermediate subspaces between X'_S and X'_T. However, the samples should learn the geometry and distribution in both the source and target domain. Now the issue is how to obtain an accurate and meaningful intermediate sample between X'_S and X'_T, which we resolve in Section 4.1.

Algorithm 1 Principal Component Analysis.

Input: ${X_S}^\top$ with $(d \times N_1)$and ${X_T}^\top$ with $(d \times N_2)$
Output: New projected matrix X'_S and X'_T

$\mu_{S/T} = \frac{1}{N_1/N_2} \sum_{i=1}^{N_1/N_2} {X_{S_i/T_i}}^\top$
If $d <= N_1/N_2$
  $S = \frac{1}{N} \sum_{i=1}^{N} ({X_{S_i/T_i}}^\top - \mu_{S/T})({X_{S_i/T_i}}^\top - \mu_{S/T})^\top$
  $X_S/X_T =$ eigenvectors of S
Else
  $S = \frac{1}{N} \sum_{i=1}^{N} ({X_{S_i/T_i}}^\top - \mu_{S/T})^\top ({X_{S_i/T_i}}^\top - \mu_{S/T})$
  $Vec_S/Vec_T =$ eigenvectors of S
  $X_S/X_T = X_S^\top/X_T^\top \times Vec_S/Vec_T$
End

2.2 Subspace learning for recognition

Given training data: X_S, with its labels $Y_S = \{y_i\}_{i=1}^{N_1} \in \{1, 2, 3, \cdots, C\}, \forall i$, one of C categories, and the test data: X_T with its labels $Y_T = \{y_j\}_{j=1}^{N_2} \in \{1, 2, 3, \cdots, C\}, \forall j$ and $N_2 \leq N_1$, that implies that we might not have all labels for testing data. If $N_2 = N_1$, which means we have labels for every element of X_T, then we want to build a model that can predict labels of X_T with accuracy as high as possible. If $N_2 < N_1$, we not only want to get a high enough predictive accuracy, but also to predict the labels for the unlabeled data. Another issue is how to predict the labels of test data by using the learned intermediate sub-spaces between source X'_S and target X'_T, which we address in Section 4.3.

3 Background: demon registration

Before discussing our proposed demon sampling method, we first review the background of demon registration.

In image registration, the task is to find a transformation in which the corresponding locations (e.g., perhaps anatomical parts) are the same within two images. Let f be a point in the target domain (X'_T), and s be another point in the source domain (X'_S). Here we treat X'_S and X'_T as two images. Proposed by Thirion [30], demon registration estimates the displacement $\mathbf{u}$ (velocity) for any point in X'_T to match the corresponding point in X'_S:

$$\mathbf{u} = \frac{(s - f)\nabla f}{|\nabla f|^2 + (s - f)^2}, \tag{1}$$

where $\mathbf{u} = (u_x, u_y)$ in 2D registration, and $\mathbf{u} = (u_x, u_y, u_z)$ in 3D registration; ∇f is the gradient of target image, and it is an internal edge force. $s - f$ is the external force, and $(s - f)^2$ can make velocity more stable. Algorithm 2 describes the process for demon registration, where U consists of $\mathbf{u}$ ($\mathbf{u}$ updates the x, y or x, y, z in translation of the pixel from U), and $\circ$ is the interpolation operation, which will update the image according to $\phi + U$, and $\phi + U$ are the coordinate points.

Algorithm 2 Demon Registration.

Input: Source image X'_i, target image X'_T, and number of iterations: $itr1$
Output: Registered image X''_S
1: Initialize transformation field ϕ, initialize $X_0 = X'_S$
2: **For** $i = 1$ to $itr1$
3: Calculate U according to (1)
4: $U = Ker \star U$
5: Update image $X_i = X_{i-1} \circ (\phi + U)$
6: **end**

Ker is the Gaussian kernel for smoothing the velocity field, and $Ker \star U$ stands for performing Gaussian kernel on transformation U. However, demon registration will take a long time to register source and target image if there is significant difference between them. $itr1$ should be large enough to guarantee good registration results. Also, the transformation U will keep updating on X'_S; there are no stages to show the step-wise changes in the image. In the following section, we will show how our fast and multi-stage sampling demon method can overcome these limitations.

4 Experimental method

4.1 Fast sampling demon for feature learning

Unlike typical image registration techniques, we aim to get the intermediate features instead of just the final transformation. We define a fast and muti-stage sampling demon model. Given source data X'_S and target data X'_T, the sampling task is to find the intermediate sub-space between X'_S and X'_T. (Notice that both source and target data are derived from extracted features from a pre-trained network; we apply the image registration techniques for features of the deep network). We aim to minimize following energy function:

$$E = \frac{1}{2}\sum_{k=1}^{K} ||X'_T - X_k \circ (\phi_{k-1} + U_k)||^2 + \frac{\alpha^2}{2}\sigma^2||U_k||^2, \tag{2}$$

where ϕ is the transformation field in x and y direction, ϕ_0 is the initial transformation which is a null matrix with the same size as X'_S and X'_T; U_k is the update transformation field of each input image at phase k. α and σ control the regularity term ($||U_k||^2$), which regulates the smoothness of transformation. X_k is the intermediate feature, which is what we want to learn from source domain to target domain and it can be updated by $X_k = X_{k-1} \circ (\phi_{k-1} + U_k)$. The first similarity term measures the similarity of the source data and intermediate features. In the first iteration, $X_0 = X'_S$, then X_k will keep updating at each iteration.

To minimize the energy function in (2), we use Taylor expansion, and rewrite (2) as:

$$E' = \frac{1}{2}\sum_{k=1}^{K} ||X'_T - X_k + U_k\nabla X_k)||^2 + \frac{\alpha^2}{2}\sigma^2||U_k||^2, \tag{3}$$

Taking the gradient of (3) with respect to U_k, we obtain:

$$\nabla_{U_k} E' = \nabla X_k(X'_T - X_k + U_k\nabla X_k) + \alpha^2\sigma^2 U_k \tag{4}$$

Let $\nabla_{U_k} E' = 0$, we get the update transformation U_k as:

$$U_k = \frac{(X_k - X'_T)\nabla X_k}{|\nabla X_k|^2 + \alpha^2\sigma^2} \tag{5}$$

Equation (5) only contains the internal force of learned intermediate feature X_k, to improve the registration convergence speed and the stability. We add another internal force of $\nabla X'_T$, and let σ be $X_k - X'_T$ to get the new update transformation in our multi-stage model:

$$\begin{aligned} U_k &= \frac{(X_k - X'_T)\nabla X_k}{|\nabla X_k|^2 + \alpha^2(X_k - X'_T)^2} + \frac{(X_k - X'_T)\nabla X'_T}{|\nabla X'_T|^2 + \alpha^2(X_k - X'_T)^2} \\ X_k &= X_{k-1} \circ (\phi_{k-1} + U_{k-1}), \end{aligned} \tag{6}$$

where $k = 1, \cdots K$ (K is the number of phases). By estimating U_k and X_k using gradient descent, we can keep learning feature deformations from source to the target. Note that X_k is just X_S in the original demon registration [30], which cannot be updated in each phase. Our SSD model could quickly find corresponding positions between the source and the target image compared with earlier methods, but in addition it can explain model differences in image appearance. We describe our SSD in Algorithm 3, where Ker' is the Gaussian kernel.

Algorithm 3 Fast Demon Sampling.

Input: Source data X'_S, target data X'_T, noise α, number of iterations $itr2$, number of phases K, and smoothing stage kk
Output: Intermediate feature X_k

```
Initialize transformation field U, and X_k
For k = 1 to K
  For i = 1 to itr2
    Calculate U_k according to (6)
    if (k * iter2 >= kk)
      U = Ker' * U
    Update transformation φ_k = φ_{k-1} + U_{k-1}
    Update feature X_k = X_{k-1} ∘ φ_k
  end
end
```

The time complexity of Algorithm 2 is $\mathcal{O}(itr1 \times Ker)$, and the time complexity of Algorithm 3 is $\mathcal{O}(K \times itr2 \times Ker')$. However, $itr1 > K \times itr2$ and $\mathcal{O}(Ker) >> \mathcal{O}(Ker')$, therefore, our SSD model is faster than the original demon registration. In addition, Algorithm 2 applies a Gaussian kernel in each iteration, which will reduce the resolution of the registered image. In our SSD model, we have parameter kk, which controls when to start the smoothing stage to help maintain the resolution of learned features.

4.2 Evaluation of learned features

Learning an intermediate feature from a source domain to a target domain is an unsupervised process since data that can be used to give feedback to the model is rare. This is one reason why there is no existing work that measures the quality of features in sub-space learning. However, evaluating features is essential; we cannot accurately learn the domain changes with poor feature learning methods. Here we explore two approaches.

4.2.1 Correlation (Corr)

First, we consider an initial method: correlation. We calculate the correlation of source and target using (7) to measure the similarity between them, and such correlation can impact final recognition results in Section 5.

$$r(A, B) = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n A_{mn} - \bar{A})^2 (\sum_m \sum_n B_{mn} - \bar{B})^2}}, \tag{7}$$

where $\bar{A}$ and $\bar{B}$ are the averages of the matrix elements. The range of the correlation is from -1 (strong negative) to 1 (strong positive), while 0 indicates there is no correlation between source data and target data.

4.2.2 Structural similarity (SSIM)

However, correlation is unable to represent the structure and position difference. Therefore, we also consider a structure similarity metric to measure the similarity between learned features X_k and target (X'_T) data. The quality of an image can be assessed by three characteristics of an image: luminance, contrast, and structure [37]. Based on these three

characteristics, the structural similarity (SSIM) index can be constructed by the combination of the three terms as described by Wang et al. [37]:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where $\mu_x, \mu_y, \sigma_x, \sigma_y$, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x, y. The default is: $\alpha = \beta = \gamma = 1$ and $C_1 = C_2/2$. The range of the SSIM is from 0 to 1, where 1 indicates high similarity between source data and target data, and 0 means they are not similar.

The evaluation of learned features (via the SSIM metric) will provide guidance for selecting the best X_k for the recognition problem in Section 5.3.

4.3 Image recognition problem

As previously mentioned in Section 2.2, we need to predict the labels of test data by using the learned intermediate sub-spaces. We address that issue here. By estimating the quality of sub-space features from our SSD model in Section 4.2, we can select the best intermediate features X_k and we are able to apply them in a recognition problem via constructing new training and test data. The new training data is $X_{train} = X_S \times X_k$, and new test data is $X_{test} = X_T \times X_k$, where $\times$ is the multiplication operation and k is selected as the best phase. In the initial stage, X_k only contains the source information, and it will contain both the source and target information after several interactions. Therefore, we only need one best feature X_k using evaluation methods, which can learn propagation information from the source domain to the target domain. By training a classifier on X_{train}, we can predict the labels of X_{test}.

To better use the learned features, we also integrate the Manifold Embedded Distribution Alignment (MEDA) method [36] to construct new features to train the classifier. MEDA can align learned features from manifold learning. It has three fundamental steps: 1) learn features from the manifold based on Gong et al. [12]; in our case, the features will be generated from our SSD method (from Algorithm 3); 2) use dynamic distribution alignment to estimate the marginal and conditional distributions of data; and, 3) construct a new classifier from previous steps (please refer to Wang et al. [36] for more details). However, the original MEDA used the GFK model to learn manifold features, and GFK model is based on the SGF model. We show the defects of the SGF model in Section 5.1. The modified classifier (f) is defined as:

$$f = \underset{f \in \sum_{i=1}^{n} \mathcal{H}_k}{\arg\min} \; l(f(X_{train_i}), Y_{S_i}) + \eta ||f||_K^2 + \lambda \overline{D_f}(\mathcal{D}_s, \mathcal{D}_t) + \rho R_f(\mathcal{D}_s, \mathcal{D}_t) \quad (8)$$

where l is the sum of squared error loss; X_{train_i} is new training data; $||f||_K^2$ is the squared norm of f; $\overline{D_f}(\cdot, \cdot)$ represents the dynamic distribution alignment; $R_f(\cdot, \cdot)$ is a Laplacian regularization; η, λ, and ρ are regularization parameters. Specifically, $\overline{D_f}(\mathcal{D}_s, \mathcal{D}_t) = (1 - \mu) D_f(P_\mathcal{S}, P_\mathcal{T}) + \mu \sum_{c=1}^{C} D_f^c(Q_\mathcal{S}, Q_\mathcal{T})$, where μ is an adaptive factor to balance the marginal distribution ($P_\mathcal{S}$, $P_\mathcal{T}$), and conditional distribution ($Q_\mathcal{S}$, $Q_\mathcal{T}$). By training the classifier from (8), we can predict labels of test data. We solve the recognition problem by using Algorithm 4.

Algorithm 4 Image recognition using SSD.

Input: X_S, Y_S, X_T, Y_T
Output: Predicted accuracy of X_T
1: Compute source X'_S and target X'_T according to Algorithm 1
2: Compute X_k according to Algorithm 3
3: Train classifier (f) according to (8) using training data
4: Apply the trained classifier to the test data, and calculate the accuracy

5 Results

In this section, we evaluate the performance of our SSD model through one synthetic dataset and extensive experiments on large-scale public image datasets.

5.1 Synthetic data

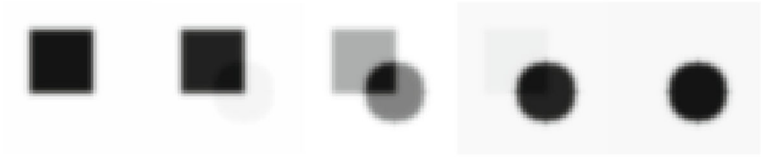
In this experiment, we show the value of our learned features via the SSD method. We compare our method with manifold-based feature learning methods SGF [13] and GSM [43] to show shape deformation across samples. We first create one synthetic source square and circle target image. The pixel values range from 0 to 1 (the pixel will be darker as it gets closer to 1), and the image size is 50×50. As shown in Fig. 1, the source image is a square (the leftmost of Fig. 1a), and the target image is a circle (the rightmost of Fig. 1a). The progress of sampled images of the SSD, GSM and SGF models are shown in Fig. 1a-c. To evaluate the quality of samples, there are two criteria. The sample should be similar to the source image when $t = 0$, and the sample should be similar to the target image when $t = 1$ (t is the time step). The sampled images of the SSD model when $t = 0.05$ and $t = 0.95$ in Fig. 1a are almost the same as the true square and circle, respectively. However, the sampled images of the SGF model are far from the source and target images when $t = 0.05$ and $t = 0.95$ [43].

There is one concern in the sampled image of both GSM and SGF models: the input data are normalized as shown in the leftmost and rightmost images of Fig. 1b. Comparing the original input images (leftmost and rightmost in Fig. 1a) with the normalized images (leftmost and rightmost in Fig. 1b), a light gray background is mistakenly introduced in GSM and a heavy gray in SGF. We also observe that the square shape is continuously deformed in our SSD model, while both GSM and SGF only show a linear changed from a square to a circle. In addition, there are two more issues in the sampled images of the SGF method: first, its background is quite dark; beyond the normalization above, this is caused by the Log map (the subtraction operation in the Riemannian space) not being correctly calculated in Gopalan et al. [13] (as there are some negations of the estimated velocity v between the source image and target image, which indicates the calculated "geodesic" is not the true geodesic, and which will then similarly cause problems in the GFK model). The second is that the shape is never unified, and this is caused by the Exp map (the add operation in the Riemannian space) not approaching the target at $t = 1$ [13].

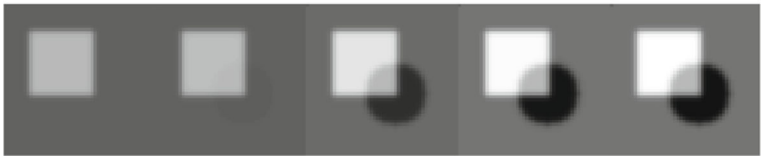
5.2 Public image recognition datasets

In these experiments, we show how our SSD method can enhance image recognition accuracy. We test our model using four public image datasets: Office10, Caltech10, Office-31

(a) The progress of samples using our SSD model



(b) The progress of samples using GSM model from Zhang et al. [43]



(c) The progress of samples using SGF model from Gopalan et al. [13]

Fig. 1 The comparison of sampling results between the two images (square and circle) with $t = 0, 0.05, 0.5, 0.95, 1$. Obviously, the SGF model does not generate a correct sample in (c), but our SSD model can generate the correct sample in (a). For reference, the source image is on the far left at $t = 0$ and the target image is the far right at $t = 1$ in Fig. 1a

and Office-Home [27, 36]. These datasets are widely used in many publications [12, 13, 36], and are the benchmarking data for evaluating the performance of domain adaptation algorithms by training in one domain and testing on another. Based on our prior experience [40, 43], we extract the features from the last fully connected layer for the four datasets from the pre-trained Inception-ResNet-v2 (IR) network [29].[1]

Table 1 lists the statistics of these datasets. **Office-10** [12] contains 1410 images in 10 classes from three domains: Amazon (A), Webcam (W), DSLR (D). Amazon images are mostly from online merchants; Webcam and DSLR images are mostly from offices. **Caltech-10** [12] contains 1123 images in 10 classes from domain Caltech (C). **Office-31** [27] consists of 4,110 images in 31 classes from three domains: Amazon (A), Webcam (W), and DSLR (D). Figure 2 shows mugs from three domains in the Office-31 dataset. **Office-Home** [33] contains 15,588 images in 65 categories across four domains: Art (Ar), denotes artistic depictions for object images, Clipart (Cl), describes picture collection of clipart, Product (Pr), shows object images with a clear background, and Real-World (Rw) represents object images collected with a regular camera.

In addition, we combine Office-10 and Caltech10 to be one dataset, and we perform twelve tasks ($A_4^2 = 12$) in this dataset: C $\rightarrow$ A, C $\rightarrow$ W, $\cdots$, D $\rightarrow$ W (where

[1] Inception-ResNet-v2 is a proposed architecture that performs highly on the ImageNet object recognition task.

Table 1 Statistics of benchmark datasets

Dataset	# Samples	# Features	# Classes	Domain(s)
Office-10	1410	1000	10	A, W, D
Caltech-10	1123	1000	10	C
Office-31	4110	1000	31	A, W, D
Office-Home	15588	1000	65	Ar, Cl, Pr, Rw

C → A means learning knowledge from domain C and applying it to the recognition task in domain A). There are another six tasks ($A_3^2 = 6$) and twelve tasks ($A_4^2 = 12$) in Office-31 and Office-Home datasets. Therefore, we have a total of 30 tasks in our experiment.

To show the robustness of our extracted Inception-ResNet-v2 (IR) features based on Keras, we show a t-SNE [22] view of extracted features. As shown in Fig. 3, the extracted IR features produce clearer and better separated clusters than SURF [12], DeCAF [36], and Resnet50 [39] features. Therefore, we can assume that the IR features will lead to better classification results than the others. Similarly, the visualizations based on the IR features have the lowest loss values.

Fig. 2 Example images from the mug category in Office-31. The first row is from Amazon domain, the second row is from DSLR domain and the last row is from Webcam domain

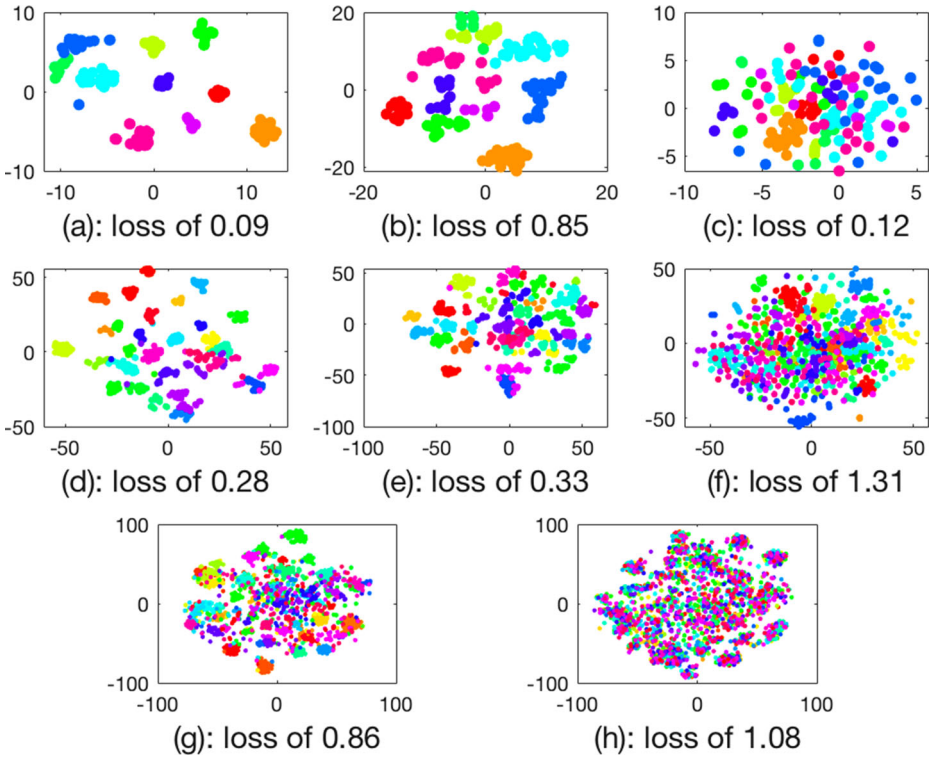


Fig. 3 The t-SNE [22] view of the comparison of our IR features (**a**, **d** and **g**) with DeCAF (**b** and **e**), Resnet50 (**h**), and SURF features (**c** and **f**). Different color means different classes. The first row is from DSLR domain in Office+Caltech-10 datasets, and the second row is from the Webcam domain in the Office-31 dataset, and (**g**) and (**h**) are from the Art domain in Office-Home dataset

5.3 Comparison to state-of-the-art methods

Table 2 shows the correlation between source and target data after using PCA to reduce the dimensionality of the original source and target data in Office-31 dataset. Here, A $\longleftrightarrow$ W means the correlation from A $\longrightarrow$ W is the same as W $\longrightarrow$ A. We observe that A $\longleftrightarrow$ W has relatively low correlation, which implies that the source domain is quite different from target domain since 0.0021 is close to 0 (indicating no correlation). We also measure structure similarity to evaluate learned features from the SSD model. As shown in Fig. 4, there are six tasks (Office-31 dataset), and as the number of iterations grows, the SSIM index increases, which implies that the sampled intermediate features X_k from our SSD model are meaningful since they contain information from both source and target data. $X_{k=0}$ only contains information from the source domain; with increasing values of k, more information from the target data can be learned, whereas source information is also included. We

Table 2 Correlation of source and target in the Office-31 dataset

Task	A $\longleftrightarrow$ W	A $\longleftrightarrow$ D	W $\longleftrightarrow$ D
Correlation	0.0021	-0.0096	-0.0057

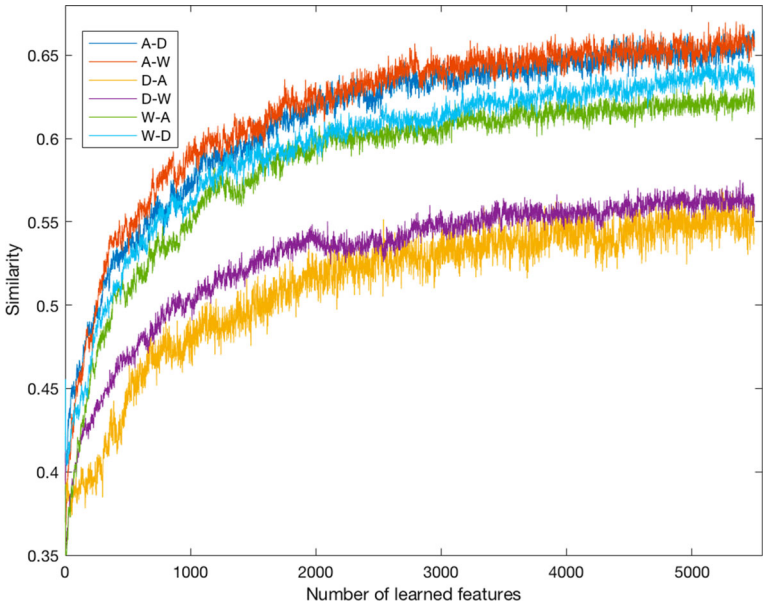


Fig. 4 The similarity between learned intermediate features (X_k) and target domain in the Office-31 dataset. We then select the best X_k with highest similarity for the recognition task

find the SSIM index tends to converge after a large number of iterations, and the convergence suggests that the learned features should be good enough for predicting the labels in the test data. Table 3 shows the results of using each of the evaluation methods; the SSIM index presents a higher performance, so we use the SSIM index as the evaluation measure in all experiments.

We compare the performance of our SSD model with 22 state-of-the-art methods. To ensure a fair comparison, all traditional methods are tested using our IR features (and named ending with "-IR" in Tables 4, 5, 6 and 7) and Figs. 5, 6 and 7. Bold numbers reflect the highest accuracy in each task in Tables 2, 3, 4, 5, 6, and 7.

5.4 Analysis of experimental results

As previously mentioned, we compare our predictive accuracy with 22 benchmark methods. From Tables 4, 5 and 7 and Figs. 5, 6, and 7 we can make several observations: first of all, the accuracy of our SSD model is above all other methods in most tasks. Notably, our model beats all other models in the Office-Home datasets. Across all four datasets, the overall average performance significantly improves upon the state-of-the-art baseline methods.

Table 3 Accuracy (%) on Office-31 datasets using two evaluation methods

Task	A → W	A → D	W → A	W → D	D → A	D → W	Average
Corr	90.3	90.8	**77.7**	98.8	77.3	96.9	88.6
SSIM	**92.7**	**91.6**	77.5	**99.4**	**77.8**	**98.0**	**89.4**

Table 4 Accuracy (%) on Office-Home datasets

Task	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar
SVM-IR	47.1	71.2	77.6	60.4	68.8	68.9	62.9
1NN-IR	45.6	67.7	73.1	59.9	67.2	67.5	65.1
GFK-IR [12]	37.8	63.1	68.3	52.7	61.3	62.8	55.6
BDA-IR [35]	41.6	65.5	70.2	53.6	63.3	65.7	56.3
JDA-IR [18]	47.1	69.4	73.2	59.1	66	68.5	62.7
TJM-IR [19]	46.7	69.1	73.1	57.3	65.2	68.2	59.3
JGSA-IR [38]	41.1	67.7	70.4	40.2	62.4	60.8	50.4
MEDA-IR [36]	47.6	71.4	76.7	63.5	73.6	73.4	63.7
GSM-IR [43]	55.0	**80.6**	**82.1**	66.6	81.5	80.1	68.0
DCORAL [28]	32.2	40.5	54.5	31.5	45.8	47.3	30.0
RTN [20]	31.3	40.2	54.6	32.5	46.6	48.3	28.2
DAH [33]	31.6	40.8	51.7	34.7	51.9	52.8	29.9
MDDA [26]	35.2	44.4	57.2	36.8	52.5	53.7	34.8
DAN [16]	43.6	57.0	67.9	45.8	56.5	60.4	44.0
DANN [11]	45.6	59.3	70.1	47.0	58.5	60.9	46.1
JAN [21]	45.9	61.2	68.9	50.4	59.7	61.0	45.8
CDAN-RM [17]	49.2	64.8	72.9	53.8	62.4	62.9	49.8
CDAN-M [17]	50.6	65.9	73.4	55.7	62.7	64.2	51.8
SSD	**54.1**	80.1	80.9	**68.2**	**81.1**	**80.1**	**69.9**

Task	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	**Average**
SVM-IR	48.4	78.1	68.6	49.0	79.3	65.0
1NN-IR	48.1	76.6	68.6	49.8	76.4	63.8
GFK-IR [12]	41.6	72.8	61.6	43.7	72.8	57.8
BDA-IR [35]	44.8	73.3	62.8	48.9	74.4	60.0
JDA-IR [18]	48.7	75.8	67.0	51.8	77.9	63.9
TJM-IR [19]	47.8	75.9	65.9	52.0	76.8	63.1
JGSA-IR [38]	42.2	66.0	51.5	43.8	68.5	55.4
MEDA-IR [36]	49.4	78.0	68.5	50.4	79.4	66.3
GSM-IR [43]	54.0	81.9	70.6	**57.8**	83.6	71.8
DCORAL [28]	32.3	55.3	44.7	42.8	59.4	42.8
RTN [20]	32.9	56.4	45.5	44.8	61.3	43.5
DAH [33]	39.6	60.7	45.0	45.1	62.5	45.5
MDDA [26]	37.2	62.2	50.0	46.3	66.1	48.0
DAN [16]	43.6	67.7	63.1	51.5	74.3	56.3
DANN [11]	43.7	68.5	63.2	51.8	76.8	57.6
JAN [21]	43.4	70.3	63.9	52.4	76.8	58.3
CDAN-RM [17]	48.8	71.5	65.8	56.4	79.2	61.5
CDAN-M [17]	49.1	74.5	68.2	56.9	80.7	62.8
SSD	**55.8**	**82.1**	**72.9**	57.2	**84.3**	**72.3**

Secondly, the performance of other baselines methods includes distribution alignment-based methods (TJM [19], BDA [35], JDA [18], JGSA [38]) and manifold learning-based methods (GFK [12], MEDA [36]) which have lower performance than our methods. And

Table 5 Accuracy (%) on Office + Caltech 10 datasets

Task	C → A	C → W	C → D	A → C	A → W	A → D	W → C
SVM-IR	95.2	94.6	93.6	92.3	88.8	87.3	93.2
1NN-IR	92.6	93.9	**96.2**	91.9	89.8	92.4	93.4
GFK-IR [12]	91.8	94.9	94.9	92.3	87.5	90.4	93.1
BDA-IR [35]	93.7	94.6	94.9	92.6	90.8	90.4	92.8
JDA-IR [18]	93.5	94.6	94.9	92.3	91.9	91.7	93.2
TJM-IR [19]	93.8	95.3	94.9	92.1	92.2	91.7	93.4
JGSA-IR [38]	95.8	95.9	94.9	93.1	89.8	95.5	94.9
MEDA-IR [36]	**96.2**	95.9	**96.2**	**95.2**	98.0	96.8	94.5
GSM-IR [43]	96.1	95.3	**96.2**	94.2	98.6	98.1	95.1
DAN [16]	92.0	90.6	89.3	84.1	91.8	91.7	81.2
DDC [32]	91.9	85.4	88.8	85.0	86.1	89.0	78.0
DCORAL [28]	89.8	**97.3**	91	91.9	**100**	90.5	83.7
RTN [20]	93.7	96.9	94.2	88.1	95.2	95.5	86.6
MDDA [26]	93.6	95.2	93.4	89.1	95.7	96.6	86.5
SSD	**96.2**	96.2	**96.2**	94.9	**98.3**	**98.3**	**98.1**

Task	W → A	W → D	D → C	D → A	D → W	**Average**
SVM-IR	95.4	98.7	92.5	94.5	98.6	93.7
1NN-IR	94.9	99.4	92.3	94.2	98.6	94.1
GFK-IR [12]	95.2	99.4	92.4	94.6	98.0	93.7
BDA-IR [35]	95.3	98.7	91.7	93.9	98.0	94.0
JDA-IR [18]	95.4	98.7	92.2	94.20	98	94.2
TJM-IR [19]	95.5	98.7	92.2	94.7	98.3	94.4
JGSA-IR [38]	95.5	100	93.9	**96.2**	99.0	95.4
MEDA-IR [36]	96.2	99.4	93.8	95.4	98.6	96.4
GSM-IR [43]	96.3	100	94.2	95.7	98.6	96.5
DAN [16]	92.1	**100**	80.3	90.0	98.5	90.1
DDC [32]	83.8	**100**	79.0	87.1	97.7	86.1
DCORAL [28]	81.5	90.1	88.6	80.1	92.3	89.7
RTN [20]	92.5	**100**	84.6	93.8	99.2	93.4
MDDA [26]	94.8	**100**	84.7	94.7	**99.4**	93.6
SSD	**98.1**	98.7	**96.2**	**96.2**	99.3	**97.2**

each method has their own limitations; they cannot guarantee the high accuracy of most tasks (e.g., although TJM and JDA have the highest accuracy in some tasks, they lose superiority in other tasks). Our model also wins against a number of deep learning models (DAN [16], DDC [32], MDDA [26], DCORAL [28], RTN [20], ADDA [31], CAN [39], JDDA [8] and JAN [21]). As shown in Table 6, we also test our model using the lower quality Resnet50 features; it still shows that our SSD model is better than other methods, as the average accuracy of the Resnet50 feature (94.8%) is lower than that of the IR feature (98.1%). This further demonstrates that we not only take advantage of better input features (which are extracted from the pre-trained Inception-ResNet-v2 model) but SSD can better align the features from source to target domain.

Table 6 Accuracy (%) on Office + Caltech 10 datasets with Resnet50 feature

Task	SVM	1NN	GFK [12]	BDA [35]	JDA [18]	TJM [19]	JGSA [38]	MEDA [36]	SSD
C → A	94.2	90.2	91.6	90.8	90.8	91.9	93.4	94.4	**95.5**
C → W	89.5	84.1	88.1	90.8	90.8	91.5	87.8	**93.9**	**93.9**
C → D	91.1	90.4	88.5	86.0	88.5	87.9	88.5	**95.5**	**95.5**
A → C	85.3	85.6	85.6	86.4	88.1	87.0	86.7	90.8	**91.5**
A → W	83.1	85.8	87.5	89.2	89.5	90.2	95.9	96.0	**96.9**
A → D	84.1	89.2	90.4	88.5	89.2	88.5	93.6	94.1	**94.3**
W → C	85.1	86.6	86.4	85.2	87.9	87.8	90.1	91.5	**91.1**
W → A	89.0	91.0	91.3	91.1	90.8	92.2	**94.8**	94.1	94.7
W → D	100	98.7	98.7	98.7	98.1	97.5	**100**	**100**	**100**
D → C	81.5	82.7	83.9	81.6	83.8	84.1	**89.8**	88.8	89.4
D → A	86.4	86.7	89.1	88.5	90.4	90.0	94.7	94.2	**94.8**
D → W	93.6	97.6	97.6	96.6	96.3	95.6	99.7	99.0	**100**
Average	88.6	89.1	89.9	89.5	90.4	90.4	92.9	94.4	**94.8**

Two compelling reasons cause the high accuracy of our SSD model. First, we implement subspace sampling demons to learn better features connecting source and target domains. As shown in Fig. 1, our SSD model can reveal a smooth transition from a square to a circle, while other methods based on SGF cannot show this smooth deformation. Second, we extract features from a well-trained Inception-ResNet-v2 (backbone) model, which is also beneficial for better feature representation as shown in Fig. 3.

Table 7 Accuracy (%) on Office-31 datasets

Task	A → W	A → D	W → A	W → D	D → A	D → W	Average
SVM-IR	81.5	80.9	73.4	96.6	70.6	95.1	83.0
1NN-IR	80.3	81.1	71.8	99.0	71.3	96.4	83.3
GFK-IR [12]	78.1	78.5	71.7	98.0	68.9	95.2	81.7
BDA-IR [35]	77.0	79.3	70.3	97.0	68.0	93.2	80.8
JDA-IR [18]	79.1	79.7	72.9	97.4	71.0	94.2	82.4
TJM-IR [19]	79.1	81.1	72.9	96.6	71.2	94.6	82.6
JGSA-IR [38]	81.1	84.3	76.5	99.0	75.8	97.2	85.7
MEDA-IR [36]	90.8	91.4	74.6	97.2	75.4	96.0	87.6
GSM-IR [43]	90.7	91.2	74.8	96.8	75.2	95.3	87.3
DAN [16]	80.5	78.6	62.8	99.6	63.6	97.1	80.4
RTN [20]	84.5	77.5	64.8	99.4	66.2	96.8	81.6
DANN [11]	82.0	79.7	67.4	99.1	68.2	96.8	81.6
ADDA [31]	86.2	77.8	68.9	98.4	69.5	96.2	82.9
CAN [39]	81.5	65.9	**98.2**	85.5	**99.7**	63.4	82.4
JDDA [8]	82.6	79.8	66.7	99.7	57.4	95.2	80.2
JAN [21]	85.4	84.7	70.0	**99.8**	68.6	97.4	84.3
SSD	**92.7**	**91.6**	77.5	99.4	77.8	**98.0**	**89.4**

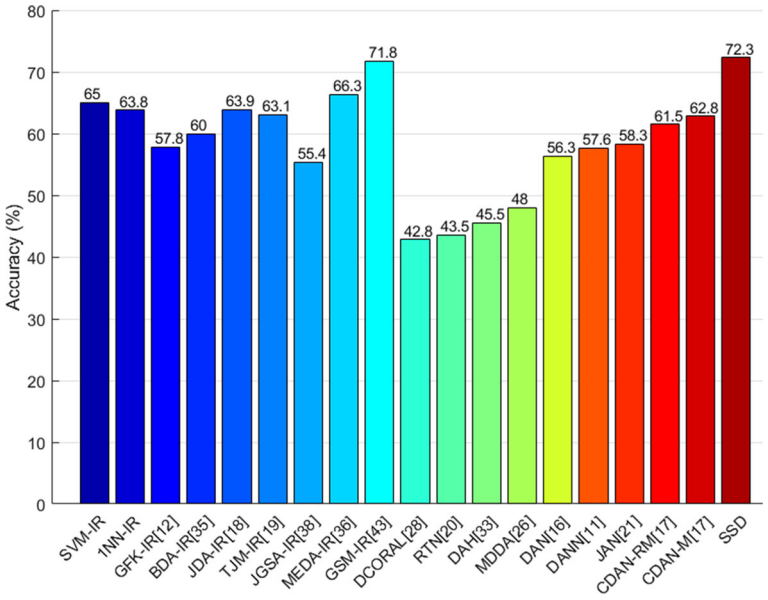


Fig. 5 The mean accuracy of all tasks in Office-Home dataset

6 Discussion

One obvious strength of our model is the higher accuracy than other methods. We compare our model SSD with the best baseline method (GSM), the p-values of relative improvement using student t-test of Office + Caltech 10, Office-31 and Office-Home datasets are

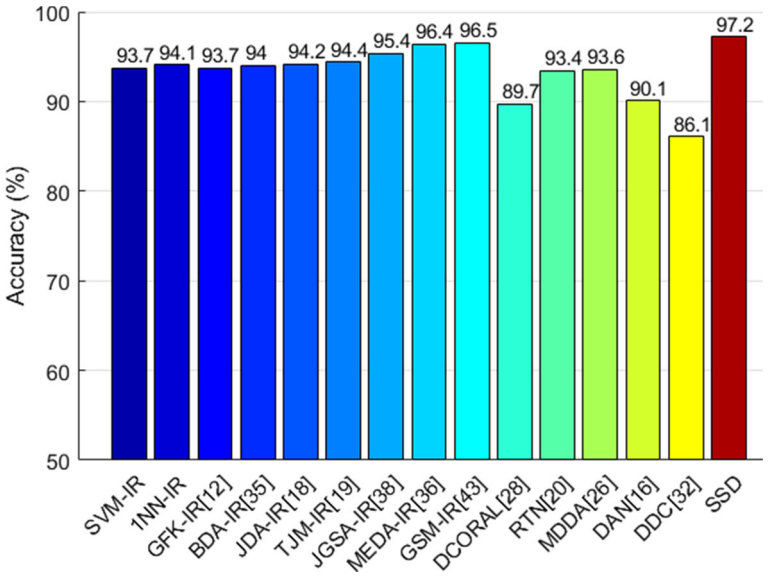


Fig. 6 Mean accuracy per approach across all tasks in Office+Caltech 10 dataset

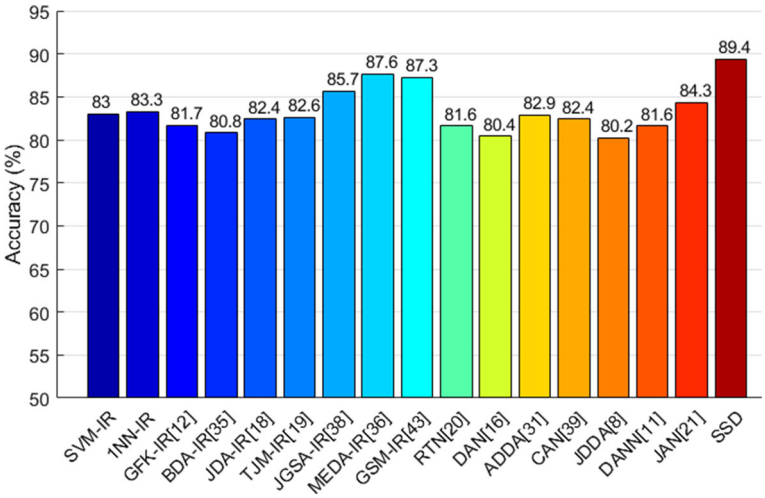


Fig. 7 The mean accuracy of all tasks in Office31 dataset

0.0037, 0.0028 and 0.00086, respectively, suggesting that our model significantly improves the image recognition accuracy. In addition, we list the reduced error rate of the proposed SSD model and the best baseline method in Table 8. The biggest reduction achieves 25%. The reduced error rate of Office-Home datasets is 1.81%, pointing out that Office-Home is a more challenging dataset since it contains more images and classes, and more domain shifts can be observed. It is also difficult to make significant improvements. However, we achieve a 15.12% improvement over the best deep learning-based model CDAN-M [17]. The high performance of our model is caused by the two reasons, as mentioned earlier: better intermediate features and better extracted backbone features.

However, one disadvantage is that the subspace sampling demon model takes more time to compute than some methods, as shown in Table 9. We can observe that our model SSD needs more time than four traditional methods (GFK, BDA, JDA, and GSM) since we need to find the optimal intermediate features. However, we still take less time than deep learning based models (DAN and JAN) since deep learning models need to re-train all images. Our SSD only focuses on extracted features without re-training on the original images. In addition, the dimensionality is another tuning parameter in our model, which also takes some time. Also, our model does not consider multiple objects in one image, and the number of classes in the source and target domains are the same as stated in Section 2.2.

An interesting phenomenon is that the performance of deep neural networks (e.g., MDDA model in Office + Caltech 10 and Office-Home datasets) are worse than some non-domain adaptation methods (e.g., SVM), when using our extracted IR features. For

Table 8 Average reduced error rate of best baseline and SSD model

Task	Best Baseline	SSD	Reduction
Office+Caltech-10	GSM [43]: 3.50	2.80	25.00%
Office-31	GSM [43]: 12.40	10.60	16.98%
Office-Home	MEDA [36]: 28.20	27.70	1.81%

Table 9 Computation time (Seconds) comparison on Office31 dataset

Task	GFK [12]	BDA [35]	JDA [18]	GSM [36]	DAN [16]	JAN [21]	SSD
A → W	0.88	50.24	51.95	33.21	730.74	890.20	656.50
A → D	0.76	41.18	41.71	27.93	493.09	540.85	411.03
W → A	0.76	49.48	51.06	15.12	830.78	950.37	663.16
W → A	0.40	5.57	5.89	8.26	493.58	620.58	401.30
D → A	0.65	42.27	42.93	9.98	505.80	628.04	413.79
D → W	0.33	5.26	5.42	6.78	492.85	530.83	399.54
Average	0.63	32.33	33.16	16.88	591.14	693.48	490.88

example, the performance of MDDA is 86.5%, which re-trains all images, while SVM is not a domain adaptation method, and achieves 93.7% using our extracted IR features. This demonstrates the extracted IR features are better than the features from those deep neural networks. However, our SSD model still outperforms the traditional methods using the same IR features, which demonstrates the value of the SSD method in domain adaptation for image recognition.

7 Conclusion

We have presented a fast subspace sampling demon method to learn intermediate subspace features from the source domain to the target domain. We then evaluate the quality of learned features. To better represent the features, we construct one distribution alignment model. The results on synthetic data reveal the superiority of the SSD approach in learning detailed features. The extensive experiments from benchmark datasets show significant improvements in classification accuracy compared to state-of-the-art methods. Therefore, we can conclude that the subspace sampling demon model is useful in bridging the gap between the source and target domain, and the best intermediate features can be selected by evaluation methods, which will lead to high performance in the domain adaptation problem.

There are some obvious next steps for follow-up work. Firstly, testing on a broader set of unsupervised domain adaptation problems will demonstrate the wider applicability of our model. Secondly, a better metric to quantify the quality of learned features will be beneficial for increasing predictive accuracy.

References

1. Belkin M, Niyogi P (2004) Semi-supervised learning on Riemannian manifolds. Mach Learn 56(1-3):209–239
2. Ben-David S, Blitzer J, Crammer K, Kulesza A, Pereira F, Vaughan JW (2010) A theory of learning from different domains. Mach Learn 79(1-2):151–175
3. Ben-David S, Blitzer J, Crammer K, Pereira F (2007) Analysis of representations for domain adaptation. In: Advances in neural information processing systems, pp 137–144
4. Ben-David S, Lu T, Luu T, Pál D. (2010) Impossibility theorems for domain adaptation. In: International conference on artificial intelligence and statistics, pp 129–136
5. Bergamo A, Torresani L (2010) Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In: Advances in neural information processing systems, pp 181–189

6. Blitzer J, Crammer K, Kulesza A, Pereira F, Wortman J (2008) Learning bounds for domain adaptation. In: Advances in neural information processing systems, pp 129–136
7. Blitzer J, Dredze M, Pereira F (2007) Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th annual meeting of the association of computational linguistics, pp 440–447
8. Chen C, Chen Z, Jiang B, Jin X (2019) Joint domain alignment and discriminative feature learning for unsupervised deep domain adaptation. In: Proceedings of AAAI
9. Chen L, Zhang H, Xiao J, Liu W, Chang SF (2018) Zero-shot visual recognition using semantics-preserving adversarial embedding network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 2
10. Fletcher PT, Lu C, Pizer SM, Joshi S (2004) Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Trans Med Imaging 23(8):995–1005
11. Ghifary M, Kleijn WB, Zhang M (2014) Domain adaptive neural networks for object recognition. In: Pacific rim international conference on artificial intelligence. Springer, New York, pp 898–904
12. Gong B, Shi Y, Sha F, Grauman K (2012) Geodesic flow kernel for unsupervised domain adaptation. In: IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 2066–2073
13. Gopalan R, Li R, Chellappa R (2011) Domain adaptation for object recognition: an unsupervised approach. In: IEEE international conference on computer vision (ICCV), IEEE, pp 999–1006
14. Jiang M, Huang W, Huang Z, Yen GG (2017) Integration of global and local metrics for domain adaptation learning via dimensionality reduction. IEEE Trans Cybern 47(1):38–51
15. Jolliffe I (2011) Principal component analysis. In: International encyclopedia of statistical science. Springer, New York, pp 1094–1096
16. Long M, Cao Y, Wang J, Jordan MI (2015) Learning transferable features with deep adaptation networks. arXiv:1502.02791
17. Long M, Cao Z, Wang J, Jordan MI (2018) Conditional adversarial domain adaptation. In: Advances in neural information processing systems, pp 1647–1657
18. Long M, Wang J, Ding G, Sun J, Yu PS (2013) Transfer feature learning with joint distribution adaptation. In: Proceedings of the IEEE international conference on computer vision, pp 2200–2207
19. Long M, Wang J, Ding G, Sun J, Yu PS (2014) Transfer joint matching for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1410–1417
20. Long M, Zhu H, Wang J, Jordan MI (2016) Unsupervised domain adaptation with residual transfer networks. In: Advances in neural information processing systems, pp 136–144
21. Long M, Zhu H, Wang J, Jordan MI (2017) Deep transfer learning with joint adaptation networks. In: Proceedings of the 34th international conference on machine learning, JMLR.org, vol 70, pp 2208–2217
22. Maaten LVD, Hinton G (2008) Visualizing data using t-sne. J Mach Learn Res 9(Nov):2579–2605
23. Mansour Y, Mohri M, Rostamizadeh A (2009) Domain adaptation:, Learning bounds and algorithms. arXiv:0902.3430
24. Pan SJ, Tsang IW, Kwok JT, Yang Q (2011) Domain adaptation via transfer component analysis. IEEE Trans Neural Netw 22(2):199–210
25. Pan SJ, Yang Q et al (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
26. Rahman MM, Fookes C, Baktashmotlagh M, Sridharan S (2020) On minimum discrepancy estimation for deep domain adaptation. In: Domain adaptation for visual understanding. Springer, New York, pp 81–94
27. Saenko K, Kulis B, Fritz M, Darrell T (2010) Adapting visual category models to new domains. In: European conference on computer vision. Springer, New York, pp 213–226
28. Sun B, Saenko K (2016) Deep coral: Correlation alignment for deep domain adaptation. In: European conference on computer vision. Springer, New York, pp 443–450
29. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI conference on artificial intelligence
30. Thirion JP (1998) Image matching as a diffusion process: an analogy with maxwell's demons. Med Image Anal 2(3):243–260
31. Tzeng E, Hoffman J, Saenko K, Darrell T (2017) Adversarial discriminative domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7167–7176
32. Tzeng E, Hoffman J, Zhang N, Saenko K, Darrell T (2014) Deep domain confusion:, Maximizing for domain invariance. arXiv:1412.3474
33. Venkateswara H, Eusebio J, Chakraborty S, Panchanathan S (2017) Deep hashing network for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5018–5027

34. Wang C, Mahadevan S (2009) Manifold alignment without correspondence. In: IJCAI, vol 2, p 3
35. Wang J, Chen Y, Hao S, Feng W, Shen Z (2017) Balanced distribution adaptation for transfer learning. In: Proceedings of the IEEE international conference on data mining (ICDM), pp 1129–1134
36. Wang J, Feng W, Chen Y, Yu H, Huang M, Yu PS (2018) Visual domain adaptation with manifold embedded distribution alignment. In: Proceedings of the 26th ACM international conference on multimedia, MM '18, pp 402–410, https://doi.org/10.1145/3240508.3240512
37. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
38. Zhang J, Li W, Ogunbona P (2017) Joint geometrical and statistical alignment for visual domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1859–1867
39. Zhang W, Ouyang W, Li W, Xu D (2018) Collaborative and adversarial network for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3801–3809
40. Zhang Y, Allem JP, Unger JB, Cruz TB (2018) Automated identification of hookahs (waterpipes) on instagram: an application in feature extraction using convolutional neural network and support vector machine classification. J Med Inter Res 20(11):e10513
41. Zhang Y, Davison BD (2019) Modified distribution alignment for domain adaptation with pre-trained inception resnet. arXiv:1904.02322
42. Zhang Y, Davison BD (2020) Impact of imagenet model selection on domain adaptation. In: Proceedings of the IEEE winter conference on applications of computer vision workshops, pp 173–182
43. Zhang Y, Xie S, Davison BD (2019) Transductive learning via improved geodesic sampling. In: Proceedings of the 30th british machine vision conference