Domain Adaptation Network for Cross-Scene Classification

Yakoub Bazi, *Senior Member, IEEE,* Farid Melgani[(b)], *Fellow IEEE,* Esam Othman[(a)], *Student Member, IEEE,* Haikel Alhichri, *Member, IEEE,* Naif Alajlan, *Senior Member, IEEE,* and Mansour Zuair

Abstract— **In this paper, we present a domain adaptation network to deal with classification scenarios subjected to the data-shift problem (i.e., labeled and unlabeled images acquired with different sensors and over completely different geographical areas). We rely on the power of pretrained convolutional neural networks (CNNs) to generate an initial feature representation of the labeled and unlabeled images under analysis, referred as source and target domains, respectively. Then we feed the resulting features to an extra network placed on the top of the pretrained CNN for further learning. During the fine-tuning phase, we learn the weights of this network by jointly minimizing three regularization terms, which are the: i) the cross-entropy error on the labeled source data; ii) the Maximum Mean Discrepancy (MMD) between the source and target data distributions; and iii) the geometrical structure of the target data. Furthermore, to obtain robust hidden representations we propose a mini-batch gradient-based optimization method with a dynamic sample size for the local alignment of the source and target distributions. To validate the method we use in the experiments the UC Merced dataset and a new multi-sensor dataset acquired over several regions of the Kingdom of Saudi Arabia. The experiments show that: 1) pretrained CNNs offer an interesting solution for image classification compared to state-of-the-art methods; 2) their performances can be degraded when dealing with datasets subjected to the data-shift problem; and 3) how the proposed approach represents a promising solution for effectively handling this issue.**

***Index Terms*— Cross-scene classification, multi-sensor data, distribution mismatch, pretrained CNN, domain adaptation.**

I. INTRODUCTION

Nowadays the information about the earth surface is obtained from a wide variety of earth observation platforms including satellites, aerial systems and unmanned aerial vehicles. Thanks to these diverse platforms, the remote sensing images are increasingly available with different spectral, spatial, and temporal resolutions. On the other hand, the availability of such large amount of heterogeneous data introduces new challenges and thus calls for the development of advanced methodologies for automatic processing and analysis.

Besides the commonly pixel-based and object-based classification methodologies, the scene-level analysis is currently attracting much interest from the remote sensing community [1]-[11], [45]-[46]. Unlike the former analysis methods the latter aims to classify the image based on a set of semantic categories in accordance to human interpretation. This task is particularly challenging as it requires the definition of high-level features for representing the image content. To this end, several scene-level classification methods have been proposed, such as the bag of word (BOW) model [1]–[4], sparse representation [5], compressive sensing [6], and lately deep learning [7]–[11]. This last yielded state-of-the-art results on several benchmark remote sensing datasets confirming the outstanding performances achieved in many other applications such as natural scene image classification [12], object recognition [13], face recognition [14], medical image analysis [15], speech recognition, [16] and traffic flow prediction [17].

Basically, the idea of deep learning also known as hierarchical learning is about learning a good feature representation automatically from the input data [18]–[22]. Typical deep learning architectures include deep belief networks (DBNs) [23], stacked autoencoder (SAE) [24], and convolutional neural networks (CNNs), which are actually perceived as the most effective representation learning methods for visual recognition tasks when trained on large labeled datasets [25]. However, when the number labeled samples is not large enough, specialized solutions should be developed. To this end, Luus et al. [7] proposed a multi-view strategy based on multiple view scales for the extraction of partial input sample patches from the scene. The different views were generated using data flipping and multiple partial views of a given input image. Then these views were used to train a single deep CNN. Zhang et al. [8] proposed a method called gradient boosting random convolutional network by fusing many deep CNNs. To reduce the computation complexity they introduced a modified version called random CNNs (i.e., tie some of the weights of the CNN ensemble). On the other side, other approaches based on knowledge transfer from CNNs pretrained on very large auxiliary labeled data have been also introduced recently [9]–[11]. In [9] the authors used a pretrained CNN to generate an initial feature representation of the remote sensing images under analysis.

(a) Y. Bazi, E. Othman, H. Alhichri, and N. Alajlan are with ALISR laboratory, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia (e-mail: {ybazi, hhichri, najlan}@ksu.edu.sa).

(b) Farid Melgani is with the department of Information Engineering and Computer Science, University of Trento, Via Sommarive 9, I-38123, Trento, Italy E-mail: melgani@disi.unitn.it

This CNN was pretrained on the ImageNet dataset which is composed of 1.2 million images and 1000 classes' related to natural images. After this step, they reshaped the outputs of the two last fully connected layers into 2D arrays and trained a new CNN composed of two convolutions and two fully connected layers followed by a softmax classifier. As pretrained CNN they used the Overfeat model [26], which is an improved version of the AlexNet model [27]. By contrast Esam et al. [10] used the output of the last fully connected layer of the CNN model proposed by Chatfield et al. [28] as input to a sparse autoencoder for learning a new feature representation. Then they tailored the sparse autoencoder to view the classification problem from discriminative and reconstruction perspectives. Finally, Fan et al. [11] investigated the problem of transferring features from CNN models trained also on very large auxiliary labeled dataset. In particular, they considered two different scenarios for carrying out feature extraction. For the first scenario, they used the last fully connected layer, while for the second one they extracted dense features from the convolution layer at multiple scales and then used different encoding techniques to generate the final representation of the image. For both scenarios, they used a support vector machine (SVM) classifier for training. In the experiments, they carried out extensive analysis using several pretrained CNN models.

The aforementioned developments show that pretrained CNNs are undoubtedly valuable tools for building sophisticated classification systems for the analysis of remote sensing images as noticed also in other fields [29]. Typically, the knowledge transfer from these models can be made either by fine-tuning the complete CNN network on the new labeled data (which is usually not large enough) or simply by extracting features from the fully connected or convolutional layers. Then these features are fed into additional fully connected layers or to the support vector machine (SVM) classifier for learning. However, this process may not be sufficient when dealing with images acquired under different acquisition conditions. Indeed, the occurrence of such scenarios in the case of remote sensing is inevitable as the images are continuously acquired with different sensors and over different locations of the earth surface. Thus it is expected that simple fine-tuning will fail to produce acceptable results mainly due to the data-shift problem. For such purpose, it becomes necessary to develop sophisticated solutions based on domain adaptation to increase the generalization ability of the network. It is worth recalling that domain adaptation has been investigated previously in the context of remote sensing but for pixel-based classification and the related solutions were mainly based on shallow architectures such as kernel methods [30], [31], graph matching [32], [33] and compressive sensing [34].

To address this challenging problem, this work proposes a domain adaptation network (DAN) composed of a pretrained CNN coupled with supplementary fully connected layers as shown in Figure 1. The weights of these fully connected layers are learned by simultaneously optimizing three terms related to discrimination, distributions mismatch and data geometry, respectively. In detail, the first term is the usual cross-entropy error computed on the labeled source examples. The second one is based on the Maximum Mean Discrepancy (MMD) [37] criterion, which has been previously adopted by shallow architectures to measure the distribution mismatch. Usually it is computed as the distances between the means of source and target examples in a reproducing Hilbert space spanned by a kernel. In this work, we express the MMD as the distance between the means of the source and target representations obtained by the hidden layers of the network. Finally, the third term uses graph Laplacian regularization [38] to preserve the geometrical structure of the unlabeled target data in the new representation space when reducing the mismatch between the distributions. In addition, to learn robust representations, we tailor the network to align distributions of the two domains in a local way via a mini-batch gradient-based optimization method with a dynamic sample size, which makes the method suitable for large scale problems.

It is interesting to mention that some domain adaptation networks have been proposed in the literature of computer vision [47]-[49]. However, our approach convoys the following main features:

- The source and target images are considered to be labeled and unlabeled, respectively, while the work proposed in [48] assumes also the existence of few labeled target images besides the labeled source images;
- DAN carries out domain adaptation using an extra network with multiple hidden layers placed on the top of a pretrained CNN and integrates graph Laplacian and MMD regularization terms besides the cross-entropy error. In addition it is based on an efficient dynamic minibatch optimization solution for the local the alignment of the source and target images. The method proposed in [47] uses a gradient reversal layer and trains the entire CNN network which is computationally demanding. On the other side, the method in [49] proposes a formulation for a shallow network with only one hidden layer and uses a full-batch optimization procedure for the global alignment of the distribution of the source and target images;
- In the experiments we will show that DAN is able to combat the data-shift problem and allows to generate significant improvements in terms of classification accuracy compared to the existing solutions on a cross-dataset composed of images acquired over the Kingdom of Saudi Arabia (KSA) and USA, respectively.

The remainder of the paper is organized as follows: Section II will provide a general description of the proposed approach. Experimental results will be presented in Section III, followed by conclusions and future works in section IV.

II. METHODOLOGY

A. Feature Extraction using Pretrained CNNs

Deep CNNs are composed of several layers of processing, each comprising linear as well as non-linear operators, which are learnt jointly, in an end-to-end way, to solve specific tasks [25]–[27]. Specifically, Deep CNNs are commonly made up of different types of layers, which are convolution; normalization, pooling and fully connected. The convolutional layer is the core building block of the CNN and its parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input image. The output of this layer is called feature maps. The feature maps, produced via convolving the learnable filters across the input image, are fed to a non-linear gating function such as the Rectified Linear Unit (ReLU). Then the output of this activation function can further be subjected to normalization (i.e., local response normalization) to help in generalization. Regarding the pooling layer, it takes small rectangular blocks from the convolutional layer and subsamples it to produce a single output from each block. There are several ways to perform pooling, such as taking the average or the maximum, or a learned linear combination of the values in the block. After several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. A fully connected layer takes all neurons in the previous layer and connects it to every single neuron it has. In the case of classification, a softmax layer is added at the end of this network and the weights of the CNN are learned using back-propagation.

In this work, we follow the recent approaches for exploiting pretrained CNN models by taking the output of the last fully connected layer (before the classification layer) to represent the image scenes. That is we feed each image $\mathbf{I}_i$ as input to the network and generate its corresponding CNN feature representation vector $\boldsymbol{x}_i \in \Re^d$ of dimension d:

$$\boldsymbol{x}_i = f_{L-1}^{\text{CNN}}\left(\dots f_2^{\text{CNN}}\left(f_1^{\text{CNN}}(\mathbf{I}_i)\right)\right),\ i = 1, \dots, n_s + n_t \tag{1}$$

where $f_j^{\text{CNN}}, j = 1, \dots, L-1$ represent the functions defining the different layers of CNN, L is the total number of layers, and n_s and n_t represent the number of labeled source images and unlabeled target images, respectively.

B. Adaptation with one hidden fully connected layer

For simplicity, we present the method for one hidden layer, then we show its generalization to the case of multiple hidden layers. Let us refer to $D^{(\text{s})} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{n_s}$ as the labeled source data and $y_i \in \{1,2,..,K\}$ is its corresponding class label. Similarly, let us define $D^{(\text{t})} = \{\boldsymbol{x}_j\}_{j=1}^{n_t}$ as the unlabeled target data. In the rest of the paper, we introduce the superscripts (s) and (t) whenever it is necessary to distinguish between the source and target domains. Our goal is develop a DAN approach that jointly learns a shared representation between the source and target data and minimizes the training error over the source data.

To achieve this goal, we feed these CNN feature vectors to an extra network placed on the top of the pretrained CNN as shown in Figure 1. Specifically, this network is composed of two fully-connected layers: hidden and *softmax* regression. The hidden layer takes the input $\boldsymbol{x}_i$ and maps it to another representation $\boldsymbol{h}_i^{(1)} \in \Re^{d^{(1)}}$ of dimension $d^{(1)}$ through the nonlinear activation function f as follows:

$$\boldsymbol{h}_i^{(1)} = f(\boldsymbol{W}^{(1)}\boldsymbol{x}_i) \tag{2}$$

where $\boldsymbol{W}^{(1)} \in \Re^{d^{(1)} \times d}$ is the mapping weight matrix. A typical choice of the activation function is the sigmoid function i.e, $f(v) = 1/(1+\exp(-v))$. For simplicity, we omit the bias vector in the expression as it can be incorporated as an additional column vector in the mapping matrix then in that case the feature vector should be appended by the value 1. The *softmax* regression performs multiclass classification and takes as input the resulting hidden representation $\boldsymbol{h}_i^{(1)}$ and produces an estimate of the posterior probability for each class label $k = 1,2, \dots, K$ as follows:

$$p(\hat{y}_i = k|\boldsymbol{x}_i) = \frac{\exp\left(\left(\boldsymbol{w}_k^{(2)}\right)^{\text{T}}\boldsymbol{h}_i^{(1)}\right)}{\sum_{j=1}^{K}\exp\left(\left(\boldsymbol{w}_j^{(2)}\right)^{\text{T}}\boldsymbol{h}_i^{(1)}\right)} \tag{3}$$

Where $\boldsymbol{W}^{(2)} = \left[\boldsymbol{w}_1^{(2)}\ \boldsymbol{w}_2^{(2)} \dots \boldsymbol{w}_K^{(2)}\right] \in \Re^{d^{(1)} \times K}$ are the weights of the *softmax* regression layer and the superscript $(\cdot)^{\text{T}}$ refers to the transpose operation.

To prevent the network from overfiting and increase its generalization ability, we use the recently introduced dropout technique [35]. This regularization technique aims to drop nodes of the hidden layer with their weights during the training phase. It allows generating a thinned network by temporarily removing the nodes from the original fully-connected network, along with all its incoming and outgoing connections. The choice of which nodes to drop is usually done randomly. Specifically, the dropout regularization technique acts by defining $\boldsymbol{r} \in \mathcal{R}^{d^{(1)}}$ (same dimension as the hidden representation $\boldsymbol{h}_i^{(1)}$) as a vector of independent *Bernoulli* random variables each of which has a probability ρ (usually set to 0.5) of being 1. At training time, the output of the hidden layer after dropout is given as follows:

$$\begin{cases} \boldsymbol{r} = \text{Bernoulli}(\rho) \\ \boldsymbol{h}_i^{(1)} := \boldsymbol{h}_i^{(1)} \odot \boldsymbol{r} \end{cases} \tag{4}$$

with $\odot$ denoting an element-wise product. At test time dropout is turned off and all hidden unit are used, however the weights are scaled by the retaining probability ρ.

To learn the weights $\boldsymbol{\theta} = \left\{\boldsymbol{W}^{(1)}, \mathbf{W}^{(2)}\right\}$ representing the complete network structure, we propose to simultaneously minimize: i) the training error on the labeled source data; ii) reduce the shift between the source and target data domains; and iii) maintain the geometry of the unlabeled target data. The proposed cost function is then formulated as follows:

$$J\left(\boldsymbol{\theta}, D^{(\text{s})}, D^{(\text{t})}\right) = argmin\ E_{net}\left(D^{(s)}\right) + \lambda_1 MMD\left(D^{(s)}, D^{(t)}\right) + \lambda_2 Lap\left(D^{(t)}\right) \tag{5}$$

where λ_1 and λ_2 are two regularization parameters. For simplicity, we define $0 \leq \lambda \leq 1$ as a new regularization parameter balancing between the contribution of both MMD and graph terms in the cost function such that $\lambda_1 = \lambda$ and $\lambda_2 = 1 - \lambda$.

- *Cross-entropy loss* $E_{net}\left(D^{(s)}\right)$: this term measures the error between the actual network outputs and the desired outputs of the labeled source data. As the outputs of the network are probabilistic, we propose to maximize the log-posterior probability to learn the network weights, which is equivalent to minimizing the so-called cross-entropy error:

$$E_{net}\left(D^{(s)}\right) = -\frac{1}{n}\sum_{i=1}^{n_s}\sum_{k=1}^{K} 1(y_i = k)\ln\left(\frac{\exp\left(\left(\boldsymbol{w}_k^{(2)}\right)^{\mathrm{T}}\boldsymbol{h}_k^{(1)}\right)}{\sum_{j=1}^{K}\exp\left(\left(\boldsymbol{w}_j^{(2)}\right)^{\mathrm{T}}\boldsymbol{h}_l^{(1)}\right)}\right) \quad (6)$$

where $1(\cdot)$ is an indicator function that takes 1 if the statement is true otherwise it takes 0 and the superscript T refers to matrix transpose.

- *Maximum Mean discrepancy* $MMD\left(D^{(s)}, D^{(t)}\right)$: The MMD has been widely used by shallow methods for reducing the mismatch between the source and target distributions in the kernel space [30], [36], [37]. In our context, we compute it as the difference between the means of the two domains obtained by the hidden representation layer of the network. So, this term is then given as follows:

$$MMD\left(D^{(s)}, D^{(t)}\right) = \frac{1}{2}\left\|\frac{1}{n_s}\sum_{i=1}^{n_s}\boldsymbol{W}^{(1)}\boldsymbol{x}_i - \frac{1}{n_t}\sum_{j=1}^{n_t}\boldsymbol{W}^{(1)}\boldsymbol{x}_j\right\|_2^2$$
$$= \frac{1}{2}Trace\left[\left(\boldsymbol{W}^{(1)}\boldsymbol{X}\right)\boldsymbol{\tau}\boldsymbol{\tau}^{\mathrm{T}}\left(\boldsymbol{W}^{(1)}\boldsymbol{X}\right)^{\mathrm{T}}\right]$$
$$= \frac{1}{2}Trace\left[\left(\boldsymbol{W}^{(1)}\boldsymbol{X}\right)\boldsymbol{M}\left(\boldsymbol{W}^{(1)}\boldsymbol{X}\right)^{\mathrm{T}}\right] \quad (7)$$

where $\boldsymbol{X} = [\boldsymbol{x}_1 \quad \cdots \boldsymbol{x}_{n_s} \cdots \quad \boldsymbol{x}_{n_s+n_t}] \in \mathcal{R}^{d\times(n_s+n_t)}$ represents the labeled source and unlabeled target samples and $\boldsymbol{M} = \boldsymbol{\tau}\boldsymbol{\tau}^{\mathrm{T}} \in \mathcal{R}^{(n_s+n_t)\times(n_s+n_t)}$ is called the MMD matrix and $\boldsymbol{\tau} \in \mathcal{R}^{(n_s+n_t)}$ is the target domain indicator vector:

$$\tau_i = \begin{cases} 1/n_s & \boldsymbol{x}_i \in D^{(s)} \\ -1/n_t & \boldsymbol{x}_i \in D^{(t)} \end{cases} \quad (8)$$

- *Graph regularization* $Lap\left(D^{(t)}\right)$: While reducing the shift between the source and target distribution, we risk to deform the geometrical structure of the unlabeled target data unlike the labeled source data, which are controlled via the cross-entropy error. To handle this issue, we rely on graph theory [38] to constrain the network so that it keeps the geometry of the target data in the hidden representation space. If we consider $G = (V, E)$ a graph with vertices $v \in V$ and edges $e \in E$ on the unlabeled target data $D^{(t)} = \{\boldsymbol{x}_j\}_{j=1}^{n_t}$. Each feature vector in $D^{(t)}$ is associated with a vertex v_j. An edge spanning two vertices v_j and v_k is denoted by e_{jk} and the associated weight is denoted by $\omega(e_{jk})$ or simply ω_{jk}. A common choice for obtaining these weights is the Gaussian weighting function. i.e., $\omega_{jk} = \exp\left(-\left\|\mathbf{x}_j - \mathbf{x}_k\right\|^2/\beta\right)$ and β is a free parameter (usually set to 1). The degree of a vertex v_j is $d_j = \sum_k \omega_{jk}$ for all edges e_{jk} incident on v_j. The combinatorial Laplacian matrix $\boldsymbol{\mathcal{L}} \in \Re^{n_t\times n_t}$ indexed by vertices v_j and v_k is given by:

$$\mathcal{L}_{jk} = \begin{cases} d_{jk} & \text{if } v_j = v_k, \\ -\omega_{jk} & \text{if } v_j \text{ is among the } m\text{th nearset to } v_k \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

Then, the graph regularization term can be written as follows:

$$Lap\left(D^{(t)}\right) = \frac{1}{2}\sum_{e_{jk}\in E}\omega_{jk}\left\|\boldsymbol{W}^{(1)}\boldsymbol{x}_j - \boldsymbol{W}^{(1)}\boldsymbol{x}_k\right\|_2^2 \quad (10)$$

which can be written in the following matrix form:

$$Lap\left(D^{(t)}\right) = \frac{1}{2}Trace\left(\left(\boldsymbol{W}^{(1)}\boldsymbol{X}^{(t)}\right)\boldsymbol{\mathcal{L}}\left(\boldsymbol{W}^{(1)}\boldsymbol{X}^{(t)}\right)^{\mathrm{T}}\right) \quad (11)$$

where $\boldsymbol{X}^{(t)} = [\boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_{n_t}] \in \mathcal{R}^{n_t\times d^1}$ represents the unlabeled target samples.

By summing up all above terms, the total cost function $J(\boldsymbol{\theta})$ is then given by:

$$J\left(\boldsymbol{\theta}, D^{(s)}, D^{(t)}\right) = argmin \; -\frac{1}{n}\sum_{i=1}^{n_s}\sum_{k=1}^{K} 1(y_i = k)\ln\left(\frac{\exp\left(\boldsymbol{w}_k^{(2)\mathrm{T}}\boldsymbol{h}_k^{(1)}\right)}{\sum_{j=1}^{K}\exp\left(\boldsymbol{w}_j^{(2)\mathrm{T}}\boldsymbol{h}_j^{(1)}\right)}\right)$$
$$+\frac{\lambda}{2}Trace\left(\left(\boldsymbol{W}^{(1)}\boldsymbol{X}\right)\boldsymbol{M}\left(\boldsymbol{W}^{(1)}\boldsymbol{X}\right)^{\mathrm{T}}\right)$$
$$+\frac{(1-\lambda)}{2}Trace\left(\left(\boldsymbol{W}^{(1)}\;\boldsymbol{X}^{(t)}\right)\boldsymbol{\mathcal{L}}\left(\boldsymbol{W}^{(1)}\boldsymbol{X}^{(t)}\right)^{\mathrm{T}}\right) \quad (12)$$

C. Generalization to multiple hidden fully connected layers

The above network could be extended to perform domain adaptation with multiple hidden layers. If we consider ℓ as the number of hidden layers, then the new cost function can be given as follows:

$$J\left(\boldsymbol{\theta}, D^{(s)}, D^{(t)}\right) = argmin \; -\frac{1}{n}\sum_{i=1}^{n_s}\sum_{k=1}^{K} 1(y_i = k)\ln\left(\frac{\exp\left(\boldsymbol{w}_k^{(\ell+1)\mathrm{T}}\boldsymbol{h}_i^{(\ell)}\right)}{\sum_{j=1}^{K}\exp\left(\boldsymbol{w}_j^{(\ell+1)\mathrm{T}}\boldsymbol{h}_i^{(\ell)}\right)}\right)$$
$$+\frac{\lambda}{2}\sum_{l=1}^{\ell}Trace\left(\left(\boldsymbol{H}^{(l)}\right)\boldsymbol{M}\left(\boldsymbol{H}^{(l)}\right)^{\mathrm{T}}\right)$$
$$+\frac{(1-\lambda)}{2}\sum_{l=1}^{\ell}Trace\left(\left(\boldsymbol{H}^{(l)(t)}\right)\boldsymbol{\mathcal{L}}\left(\boldsymbol{H}^{(l)(t)}\right)^{\mathrm{T}}\right) \quad (13)$$

where $\boldsymbol{H}^{(l)}$ is the hidden representation obtained for both source and target samples at layer l, whereas $\boldsymbol{H}^{(l)(t)}$ is hidden representation for the target samples. They are given as $\boldsymbol{H}^{(l)} =$

$\boldsymbol{W}^{(l)}\boldsymbol{H}^{(l-1)}$ and $\boldsymbol{H}^{(l)(t)} = \boldsymbol{W}^{(l)}\boldsymbol{H}^{(l-1)(t)}$, respectively with $\boldsymbol{H}^{(0)} = \boldsymbol{X}$ and $\boldsymbol{H}^{(0)(t)} = \boldsymbol{X}^{(t)}$. In this case, the complete weights of the network are: $\boldsymbol{\theta} = \{\boldsymbol{W}^{(1)}, \dots, \mathbf{W}^{(\ell)}, \boldsymbol{W}^{(\ell+1)}\}$ with $\boldsymbol{W}^{(\ell+1)}$ denoting to the weights of the softmax layer.

D. Optimization of the cost function $J(\boldsymbol{\theta}, D^{(s)}, D^{(t)})$

To optimize the cost function $J(\boldsymbol{\theta}, D^{(\mathrm{s})}, D^{(\mathrm{t})})$, we use a mini-batch Stochastic Gradient Descent (SGD) method, which is the common choice of deep learning approaches. This last consists of dividing the labeled set into several mini-batches of the same size and then learning is performed by updating the weights for every mini-batch. In our setting, DAN will learn on both labeled and unlabeled data, respectively. So, for every mini-batch from the labeled set we should associate another mini-batch from the target domain. In an ideal situation, each mini-batch from the target domain would contain samples belonging to the same classes as those present in the associated labeled source domain to locally reduce the mismatch between the two distributions. Under this condition, one can intuitively expect to obtain an accurate alignment of the distributions using mini-batches of small size. Since the true labels of the target samples are not known, we rely on the predictions provided by DAN during the fine-tuning process to align source and target mini-batches. But this time using small mini-batches for aligning the distributions could be inappropriate as the predictions provided in the early learning stages are of low confidence due to data-shift problems. To tackle this issue, we introduce an iterative solution based on dynamic-size mini-batches. Its basic idea is to start with a large size to increase the percentage of samples belonging to the same classes in each pair of source and target mini-batches. In other words, using large mini-batches will reduce the risk of aligning samples belonging to completely different classes as the misclassified target samples would actually belong to classes that are present in the labeled target mini-batch anyway. After this step, we expect a reduction in the distribution mismatch and at the same time an increased confidence in the target labels prediction. This result will encourage us to decrease the size of the mini-batch in the next learning stage to reduce further the shift and at the same increase the prediction confidence for the target samples. This process could be repeated many times but with mini-batches with smaller and smaller sizes. In the experiments, we will show that this trick is computationally efficient and yields significant improvement in terms of classification accuracy.

Mathematically, the gradients are typically computed using the well-known backpropagation algorithm and the weights of the network are updated as follows:

$$\boldsymbol{W}^{(l)} := \boldsymbol{W}^{(l)} - \frac{\eta}{N_b}\sum_{i=1+(k-1)S_b}^{kS_b} \frac{dJ\left(\boldsymbol{\theta}, D_{S_b}^{(s)}, D_{S_b}^{(t)}\right)}{d\boldsymbol{W}^{(l)}} \tag{14}$$

where η refers to the learning rate, S_b and N_b represent the size and the number of mini-batches, respectively. $D_{S_b}^{(s)}$ and $D_{S_b}^{(t)}$ are the labeled source and unlabeled target mini-batch pairs each of size S_b extracted from $D^{(s)}$ and $D^{(t)}$, respectively. Since the proposed cost function includes new extra terms related to MMD and graph regularization besides the standard cross-entropy error, we can rewrite the equation (14) in a more detailed from as follows:

$$\boldsymbol{W}^{(l)} := \boldsymbol{W}^{(l)} - \frac{\eta}{N_b}\sum_{i=1+(k-1)S_b}^{kS_b}\left(\frac{dE_{net}\left(\boldsymbol{\theta}, D_{S_b}^{(s)}\right)}{d\boldsymbol{W}^{(l)}} + \frac{\lambda}{2}\frac{dMMD\left(\boldsymbol{\theta}, D_{S_b}^{(s)}, D_{S_b}^{(t)}\right)}{d\boldsymbol{W}^{(l)}} + \frac{(1-\lambda)}{2}\frac{dLap\left(\boldsymbol{\theta}, D_{S_b}^{(t)}\right)}{d\boldsymbol{W}^{(l)}}\right) \tag{15}$$

It can be shown that, the update equations related to the to the MMD and graph regularizations terms are given as follows:

$$\frac{dMMD\left(\boldsymbol{\theta}, D_{S_b}^{(s)}, D_{S_b}^{(t)}\right)}{d\boldsymbol{W}^{(l)}} := \boldsymbol{H}_{S_b}^{(l)}\boldsymbol{M}_{S_b}\left(\boldsymbol{H}_{S_b}^{(l-1)}\right)^{\mathrm{T}} \tag{16}$$

$$\frac{dLap(\boldsymbol{\theta}, D_{S_b}^{(t)})}{d\boldsymbol{W}^{(l)}} = \boldsymbol{H}_{S_b}^{(l)(t)}\boldsymbol{\mathcal{L}}_{S_b}\left(\boldsymbol{H}_{S_b}^{(l-1)(t)}\right)^{\mathrm{T}} \tag{17}$$

where $(\cdot)_{S_b}$ means that the variables defined in the previous sections are evaluated for a mini-batch of size S_b.

It is interesting to notice that more advanced gradient-based update rules could be used instead of (15), but here for ease of presentation we have provided this basic implementation. In the experiments, we will use an alternative version based on the momentum method [40], [41].

In the following we provide the main steps for carrying out the adaption with the proposed method and its nominal parameters:

Algorithm: *DAN method*

Input: Labeled source images $\{\boldsymbol{I}_i, \boldsymbol{y}_i\}_{i=1}^{n_s}$ and unlabeled target images $\{\boldsymbol{I}_j\}_{j=1}^{n_t}$

Output: Target Class labels

1: Set D^2AN parameters:

- $\lambda = 0.5, \ell \geq 2$,
- *#nodes=256* per hidden layer,
- $Dynamic_{minibatch} = [100\ 80\ 60\ 40\ 20\ 10]$,
- Nearest neighbors m for building the local graph on the target data in the range $[5\ \ 11]$;

2: Obtain the CNN feature vectors: $\{\boldsymbol{x}_i\}^{n_s+n_t} = pretrained\_CNN\left(\{\boldsymbol{I}_i\}_{i=1}^{n_s+n_t}\right)$;

3: Set the initial min-batch size $S_b = Dynamic\_minibatch(1)$ (i.e., $S_b = 100$);

4: Optimize DAN using backpropagation on the labeled source data only (MMD and graph regularization terms are not considered here);

5: Feed the unlabeled target data $\{\boldsymbol{x}_j\}_{j=1}^{n_t}$ to DAN and estimate the corresponding labels;

6: $for\ j = 1: length(Dynamic\_minibatch)$

6.1: Set the number of mini-batches to: $N_b = n_s/S_b$;

6.2: Shuffle randomly the labeled source samples and organize them into N_b groups each of size S_b;

6.3: $for\ k = 1: N_b$

- Pick mini-batch k from the source data: $\{\boldsymbol{x}_i, y_i\}_{i=1+(k-1)S_b}^{kS_b}$;
- Pick randomly $\{\boldsymbol{x}_j\}_{j=1}^{S_b}$ samples from the target data having estimated labels similar to the labeled samples present in mini-batch k.
- Build a local graph on the target samples $\{\boldsymbol{x}_j\}_{j=1}^{S_b}$;
- Update the weights of DAN using backpropagation by training on the mini-batches $\{\boldsymbol{x}_i, y_i\}_{i=1+(k-1)S_b}^{kS_b}$ and $\{\boldsymbol{x}_j\}_{j=1}^{S_b}$;

$end\ for$

6.4: Feed the unlabeled target data to the updated DAN and estimate the new corresponding labels;

6.5: Set the new batch size as: $S_b = Dynamic\_minibatch(j)$;

$end\ for$

III. Experimental Results

A. Dataset Description

1) *The University of California (UC) Merced dataset*: This dataset was manually derived by Yang and Newsam [1], [2] from another data set of large aerial orthoimagery with pixel resolution of 30 cm. It was downloaded from the U.S. Geological Survey national map of the following U.S. regions: Birmingham, Boston, Buffalo, Columbus, Dallas, Harrisburg, Houston, Jacksonville, Las Vegas, Los Angeles, Miami, Napa, New York, Reno, San Diego, Santa Barbara, Seattle, Tampa, Tucson, and Ventura. It consists of 2100 RGB images of size (256×256) pixels each, categorized into 21 classes (100 images per class). The class labels are as follows: agriculture, airplane, baseball diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbor, intersection, medium-density residential, mobile home park, overpass, parking lot, river, runway, sparse residential, storage tanks, and tennis court. Sample images of this database are shown in Figure 2.

2) *KSA dataset:* This multi-sensor dataset was acquired over different cities of KSA (i.e., Riyadh, Al-Qassim, Al-Rajhi farms, Al-Hufuf and Jeddah) by three different VHR sensors including IKONOS-2, GeoEye-1 and WorldView-2 with spatial resolutions of 1 m, 0.5 m and 0.5 m, respectively. This dataset consists of 3250 RGB images of size 256×256 pixels categorized into 13 classes (250 images per class). The class labels are as follows: agriculture, beach, cargo, chaparral, dense residential, dens trees, desert, freeway, medium-density residential, parking lot, sparse residential, storage tanks, and water. Sample images of this multi-sensor dataset are shown in Figure 3.

3) *Cross-dataset:* This last dataset will be mainly used for assessing the performances of the method when the distribution of the source data is different from the target data on which the model will be applied. Typically, this cross-dataset is composed of the 8 common classes between Merced and KSA identified by visual inspection. These classes are: agriculture, beach, chaparral, dense residential, forest, freeway, parking lot, and storage tanks. Thus the number of images selected from Merced is equal to 800 (100 image per class), whereas for KSA it is equal to 2000 (250 images per class). From this cross-dataset, we build two scenarios termed as KSA→Merced and Merced→KSA referring to source→target data.

B. Experimental Setup

Pretrained CNN: For generating the convolutional features, we use the pretrained CNN model of Chatfield *et al.* [28] composed of 8 layers however any other recent pretrained CNN model could be used as well. Specifically this deep CNN uses five convolutional filers of the following dimensions: (number of filters × filter height × filter width: 96×7×7, 256×5×5, 512×3×3, 512×3×3, and 512×3×3) and three fully connected layers with number of hidden nodes: (fc1: 4096, fc2: 4096, and softmax: 1000). This model was trained on the ILSVRC-12 challenge dataset [39] composed of 1.2 million RGB images of size 224×224 pixels belonging to 1000 classes. These classes describe general images such as beaches, dogs, cats, cars, shopping carts, minivans, etc. As can be seen, this auxiliary domain is completely different from the remote sensing datasets used in the experiments. For extracting the CNN features, we resize the images of both datasets to 224×224 and then feed them to this deep CNN (without the sotmax layer) and take the output of the last fully connected fc2 which produces a feature vector of dimension $d = 4096$.

Extra Network architecture: In the experiments, we follow the practical recommendations for training deep neural networks [40], [41]. We use one hidden layer and set the #nodes to 256. For dropout, we use the standard value 0.5. For the mini-batch gradient method, we use a batch size S_b of variable size: $\{100, 80, 60, 40, 20, \text{and } 10\}$ samples per mini-batch during the adaptation phase. We fix the learning rate and momentum to the values of 1 and 0.5, respectively. In addition, we set the regularization parameters λ to 0.5, respectively (i.e., balanced weights for graph and MMD terms). For building the local graph, we fix the Gaussian weighting function to the value 1 and set the number of neighbors to 7. It is worth recalling, that we will carry out in the experiments a detailed sensitivity analysis with respect to the most important parameters of the network.

C. Results

1) *Results on Merced and KSA datasets:* In Figures 4 we provide a general view of the distributions in the 2D space of the features learned by the hidden layer of the extra network for both datasets. As can be seen, this feature visualization computed by the t-SNE method [42] indicates *a-priori* a good discrimination between the different classes. In Tables I and II, we provide the classification results obtained by the proposed DAN without and with adaptation, respectively. In the first phase, we run DAN to optimize only the cross-entropy error using a fixed mini-batch of size 100 samples. Then, in the second phase we fine-tune the network by

including the MMD and graph regularization terms (i.e. $\boldsymbol{\lambda = 0.5}$). For Merced dataset, the OA is equal to 96.75% and 94.61% for the fivefold and the twofold validations schemes without adaptation. Regarding KSA dataset, the OA is 95.08% and 94.95% for both validation schemes, respectively. As can be seen adding the adaptation terms didn't affect much the accuracies as the OA became 96.54% and 94.51%, for Merced and 95.33% and 94.36% for KSA. These results indicates that the shift between the distributions of the training and test data is not significant for both datasets. This observation is further supported by the outcomes of t-SNE feature visualization method shown in Figure 5 where the distributions looks very close. In Tables I and II, we compare also our results against several state-of-the-art methods. As can be seen these preliminary results show that: i) DAN performs clearly better than state-of-the-art methods based on handcrafted features confirming the recent findings of the literature; ii) it is competing with other deep learning approaches based on pretraining strategies; and iii) it is computationally efficient.

2) *Results on the cross-dataset:* For this dataset the situation is completely different as the source and target images were acquired over two different continents and with different sensors. Indeed, the difference in the acquisition conditions makes the data-shift problem apparent as shown in Figure 6(a) and Figure 7(a) for both scenarios. In this case, our proposed DAN without adaptation besides the existing pretraining strategies yield low classification accuracies as shown in Table III. To get the whole picture, we report in Table IV the accuracies for the different classes composing this cross dataset. For the scenario KSA→Merecd, DAN without adaptation yields an OA of 73.25%, with three classes under 70%. Regarding Merced→KSA it yields an OA of 69.95%, with 5 classes having accuracies less than 70%. By contrast, running DAN with adaptation reduces progressively the shift between the source and target distributions while maintaining a good discrimination ability between the different classes as shown in Figure 6 and Figure 7, respectively. At the end of the optimization process, it yields for KSA→Merecd an OA of 91.50% corresponding to a significant increase of 18.25% and with clear improvements for all classes. Regarding, Merced→KSA, the OA become 85.20% corresponding also to a significant improvement of 15.25%. These results show that the transfer KSA→Merecd is better than Merced→KSA. Actually, this situation was expected as the number of labeled source data in the first scenario is greater than the second one (i.e., 2000 imagers versus 800 images). In addition, the KSA dataset is more representative as it is composed from images acquired with different sensors.

3) *Analysis of the method:* As mentioned previously, to assess further the performances of the proposed DAN method, we provide in the next subsections a detailed sensitivity analysis with respect to the following parameters: the mini-batch size S_b; the regularization parameter λ; the number of #nodes in the hidden layer; and the number of hidden layers ℓ.

- *Mini-batch size effect:* To illustrate the importance of fine-tuning the alignment between the source and target data using a mini-batch with a dynamic size, we repeat the above experiments by setting S_b to a fixed value for the entire optimization process. First, we run DAN under an optimal selection scheme. That is for every mini-batch from the source data we associate samples having the same labels from the target data (true labels). The results reported in Figure 8, indicate that using a small mini-batch is the ideal choice for reducing the shift between the source and target data. By contrast, increasing the size of the mini-batch leads to a decrease in the accuracy. Regarding our proposed solution based on label estimation, we observe a different behavior. In particular, the OA exhibits a behavior similar to the Hughes effect widely encountered in the case of hyperspectral images. For KSA→Merced, the best result is obtained for $S_b = 80$ and the OA is equal to 83.75%, whereas for Merced→KSA it is obtained for $S_b = 40$ with an OA of 79.1%. These results suggest that using a small mini-batch for aligning the distributions is not a good solution as the confidence of the target labels is low in the initial stage. In addition, the utilization of the whole data for globally aligning the distributions as usually done in shallow architectures seems to be inefficient. Here arises the importance of the solution based on a dynamic mini-batch which starts with large sizes to tackle the confidence issue in the initial stages. Figure 8, shows clearly that using a DAN with a fixed mini-batch size results in improvement compared to the no-adaption case but these improvements are not competing with those obtained with the proposed dynamic mini-batch size solution.

- *Regularization parameter λ and #nodes* in the hidden layer: Figure 9 depicts the classification accuracies obtained for both scenarios by varying the regularization parameter λ in the range [0 1] and for different #nodes of the hidden layer (i.e., 64, 128, 256, 512 and 1024). These results confirm the gain of including graph regularization while reducing the shift between the distributions of source and target data. In general selecting λ in the range [0.1 0.6] leads to quit stable results. By contrast, giving more power to MMD regularization $\lambda > 0.6$ tends to decrease the improvements and leads to unstable behaviors. We recall that in the experiments, we presented the results using $\lambda = 0.5$ (equal weights for MMD and graph regularization terms). In addition the choice of #nodes =256 represents a good compromise between classification accuracy, stability and computation time.

- *Number of hidden layers ℓ:* In this experiment, we evaluate the method with respect to number of hidden layers ℓ. We vary number of hidden layers ℓ from 1 to 4 and train the network in in a greedy layer-size manner [18] on both source labeled and unlabeled target data. That is we start by training the network using only one hidden layer (i.e., $\ell = 1$) by optimizing the cost function in (12) as done in the previous experiments. Then we augment the network with an additional hidden layer and initialize its weights to small values in the range [-0.005 0.005] and retrain again the complete network by optimizing the cost function in (13). We repeat this learning process by adding one layer each time until reaching the desired number of hidden layers fixed to $\ell = 4$. To understand further the effect of the number of hidden layers, Table V shows the results obtained for different percentages of the labeled source data (i.e., 25%, 50%, 100%, and 200%, respectively). We note that the scenario 200% is generated using rotation and flipping augmentation techniques. As can

be seen, the accuracies reported in Table V confirm clearly the value of adaptation using multiple hidden layers instead of using one hidden layer. In details, for KSA→Merced, the average results obtained through all scenarios show that DAN without adaptation yields 70.72%. By contrast, the adaptation with one hidden layer results in accuracy of 85.63% (corresponding to an improvement of 14.91%). Increasing the number of hidden layers improves further the results reaching an accuracy of 91.32% using four hidden layers. The same behavior happens with the dataset Merced→KSA as DANN without adaptation provides an accuracy of 67.63% whiles it provides an accuracy of 79.87% with adaptation using one hidden layer (corresponding to an improvement of 12.24%). Then the accuracy increases to 85.21% when applying adaptation using four hidden layers. In general these results suggest that using DAN with two hidden layers represents a good compromise between accuracy and computation time.

III. CONCLUSIONS

This paper has presented a DAN method for tackling the challenging data-shift problem in remote sensing imagery. DAN has the following attractive proprieties: 1) it uses a pretrained CNN to generate an initial feature representation of both labeled source and unlabeled target images; 2) its second building block based on fully connected layers takes as input the CNN features and learn the weights by reducing the mismatch between the distributions of the source and target data while maintaining the discrimination ability of the labeled data and the geometrical structure of the target data; 4) it uses a mini-batch gradient optimization method with dynamic sample size to learn robust hidden representations; which makes it suitable for large scale problems. The experiments carried out on a non-GPU unit and on two real datasets acquired over two different geographical areas and with different sensors confirmed its computational efficiency and its ability in providing improved classification results compared to several state-of-the-art methods. In the future we plan to improve this network by including several enhancements such as: 1) introducing appropriate confidence measures for selecting samples from the unlabeled set; 2) fusing several pretrained CNN models; and 3) estimating the optimal number of hidden layers of the extra network in an automatic way.

ACKNOWLEDGEMENTS

The authors would like to extend their sincere appreciation to Deanship of Scientific Research at King Saud University for funding this Research group No. (RG-1435-050). The Authors would like to thank A. Vedaldi and K. Lenc for making available the software MatConvNet [50] used in the context of this work.

.

REFERENCES

[1] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," Proc. 18th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst., San Jose, CA, USA, 2010, pp. 270–279, 2010.

[2] Y. Yang and S. Newsam, "Spatial pyramid co-occurrence for image classification," In Proc. IEEE Int. Conf. Comput. Vis., pp. 1465–1472, 2011.

[3] S. Chen and Y. Tian, "Pyramid of spatial relatons for scene-level land use classification," IEEE Trans. Geosci. Remote Sens., 2015, vol. 53, no. 4, pp. 1947–1957, 2015.

[4] L.-J. Zhao, P. Tang, and L.-Z. Huo, "Land-Use Scene Classification Using a Concentric Circle-Structured Multiscale Bag-of-Visual-Words Model," IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 7, no. 12, pp. 4620–4631, Dec. 2014.

[5] A. M. Cheriyadat, "Unsupervised Feature Learning for Aerial Scene Classification," IEEE Trans. Geosci. Remote Sens., vol. 52, no. 1, pp. 439–451, Jan. 2014.

[6] M. L. Mekhalfi, F. Melgani, Y. Bazi, and N. Alajlan, "Land-Use Classification With Compressive Sensing Multifeature Fusion," IEEE Geosci. Remote Sens. Lett., vol. 12, no. 10, pp. 2155–2159, Oct. 2015.

[7] F. P. S. Luus, B. P. Salmon, F. Van Den Bergh, and B. T. J. Maharaj, "Multiview Deep Learning for Land-Use Classification," IEEE Geosci. Remote Sens. Lett., vol. 12, no. 12, pp. 2448–2452, 2015.

[8] F. Zhang, B. Du, and L. Zhang, "Scene Classification via a Gradient Boosting Random Convolutional Network Framework," IEEE Trans. Geosci. Remote Sens., vol. 54, no. 3, pp. 1793–1802, Mar. 2016.

[9] D. Marmanis, M. Datcu, T. Esch, U. Stilla, and S. Member, "Deep Learning Earth Obsevation Classification Using ImageNet Pretrained Networks," IEEE Geosci. Remote Sens. Lett., vol. 13, no. 1, pp. 1–5, 2015.

[10] E. Othman, Y. Bazi, N. Alajlan, H. Alhichri, and F. Melgani, "Using convolutional features and a sparse autoencoder for land-use scene classification," Int. J. Remote Sens., vol. 37, no. 10, pp. 1977–1995, 2016.

[11] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery," Remote Sens., vol. 7, no. 11, pp. 14680–14707, Nov. 2015.

[12] M. Hayat, M. Bennamoun, and S. An, "Deep Reconstruction Models for Image Set Classification," IEEE Trans. Pattern Anal. Mach. Intell., vol. PP, no. 99, pp. 1–1, 2014.

[13] J. Bai, Y. Wu, J. Zhang, and F. Chen, "Subset based deep learning for RGB-D object recognition," Neurocomputing, vol. 165, pp. 280–292, 2015.

[14] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, "Single Sample Face Recognition via Learning Deep Supervised Autoencoders," IEEE Trans. Inf. Forensics Secur., vol. 10, no. 10, pp. 2108–2118, Oct. 2015.

[15] T. Brosch and R. Tam, "Efficient training of convolutional deep belief networks in the frequency domain for application to high-resolution 2D and 3D images.," Neural Comput., vol. 27, no. 1, pp. 211–27, Jan. 2015.

[16] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," IEEE Signal Process. Mag., vol. 29, no. 6, pp. 82–97, Nov. 2012.

[17] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic Flow Prediction with Big Data: A Deep Learning Approach," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 1–9, 2014.

[18] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–828, Aug. 2013.

[19] Xue-Wen Chen and Xiaotong Lin, "Big Data Deep Learning: Challenges and Perspectives," IEEE Access, vol. 2, pp. 514–525, 2014.

[20] M. Längkvist, L. Karlsson, and A. Loutfi, “A review of unsupervised feature learning and deep learning for time-series modeling,” Pattern Recognit. Lett., vol. 42, pp. 11–24, Jun. 2014.
[21] J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85–117, Oct. 2014.
[22] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks.,” Science, vol. 313, no. 5786, pp. 504–7, Jul. 2006.
[23] G. Hinton, S. Osindero, and Y. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” Neural Comput., vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
[24] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders,” in Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1096–1103.
[25] C. Farabet, C. Couprie, L. Najman, and Y. Lecun, “Learning hierarchical features for scene labeling,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1915–29, Aug. 2013.
[26] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,” arXiv:1312.6229, Dec. 2013.
[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[28] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the Devil in the Details: Delving Deep into Convolutional Nets,” May 2014.
[29] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, “Factors of Transferability for a Generic ConvNet Representation,” pp. 1–12, 2014.
[30] J. Li, “Learn Multiple-Kernel SVMs for Domain Adaptation in Hyperspectral Data,” IEEE Geosci. Remote Sens. Lett., vol. 10, no. 5, pp. 1224–1228, Sep. 2013.
[31] E. Othman, Y. Bazi, N. Alajlan, H. AlHichri, and F. Melgani, “Three-Layer Convex Network for Domain Adaptation in Multitemporal VHR Images,” IEEE Geosci. Remote Sens. Lett., vol. 13, no. 3, pp. 1–5, 2016.
[32] D. Tuia, J. Munoz-Mari, L. Gomez-Chova, and J. Malo, “Graph Matching for Adaptation in Remote Sensing,” IEEE Trans. Geosci. Remote Sens., vol. 51, no. 1, pp. 329–341, Jan. 2013.
[33] D. Tuia, M. Volpi, M. Trolliet, and G. Camps-Valls, “Semisupervised Manifold Alignment of Multimodal Remote Sensing Images,” IEEE Trans. Geosci. Remote Sens., vol. 52, no. 12, pp. 7708–7720, Dec. 2014.
[34] M. Roy, F. Melgani, A. Ghosh, E. Blanzieri, and S. Ghosh, “Land-Cover Classification of Remotely Sensed Images Using Compressive Sensing Having Severe Scarcity of Labeled Patterns,” IEEE Geosci. Remote Sens. Lett., vol. 12, no. 6, pp. 1257–1261, Jun. 2015.
[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[36] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, “Domain adaptation via transfer component analysis.,” IEEE Trans. Neural Netw., vol. 22, no. 2, pp. 199–210, Feb. 2011.
[37] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” J. Mach. Learn. Res., vol. 13, no. 1, pp. 723–723–773–773, Jan. 2012.
[38] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples,” J. Mach. Learn. Res., vol. 7, pp. 2399–2434, Dec. 2006.
[39] R. Socher, “ImageNet: A large-scale hierarchical image database,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
[40] Y. Bengio, "Practical Recommendations for Gradient-Based Training of Deep Architectures", in: G. Montavon, G.B. Orr, K.-R. Müller (Eds.), Neural Netw. Tricks Trade, Springer Berlin Heidelberg, 2012: pp. 437–478.
[41] G. Montavon, G. B. Orr, and K.-R. Müller, “Neural Networks: Tricks of the Trade: Second Edition,”, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 421–436.
[42] L. van der Maaten, and G. E. Hinton, “Visualizing data using t-Sne,” J. Mach. Learn. Res., vol. 9, pp. 2579–2605, 2008.
[43] K. G. B. Gong, Y. Shi, and F. Sha, “Geodesic flow kernel for unsupervised domain adaptation,” in 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2066–2073.
[44] M. Chen, Z. Xu, K. Weinberger, and F. Sha, “Marginalized Denoising Autoencoders for Domain Adaptation,” Jun. 2012.
[45] X. Zhang, S. Du, and Y.-C. Wang, “Semantic classification of heterogeneous urban scenes using intrascene feature similarity and interscene semantic dependency,” IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 8, no. 5, pp. 2005–2014, May 2015.
[46] F. Hu, G.-S. Xia, Z. Wang, X. Huang, L. Zhang, and H. Sun, “Unsupervised feature learning via spectral clustering of multidimensional patches for remotely sensed scene classification,” IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 8, no. 5, pp. 2015–2030, May 2015.
[47] G. Yaroslav, and V. Lempitsky. "Unsupervised domain adaptation by backpropagation." ICML-15, vol. 37, pp. 1180.1189, July 2015.
[48] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, “Simultaneous deep transfer across domains and tasks,” Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 4068-4076, Dec. 2015.
[49] M. Ghifary, W.B. Kleijn, and M. Zhang, "Domain adaptive neural networks for object recognition," PRICAI 2014: Trends in Artificial Intelligence. Springer International Publishing, vol. 8862, pp. 898-904, May 2014.
[50] Vedaldi, A., and K. Lenc. 2014. “Matconvnet - Convolutional Neural Networks for MATLAB.” Proceedings of the ACM International Conference on Multimedia.

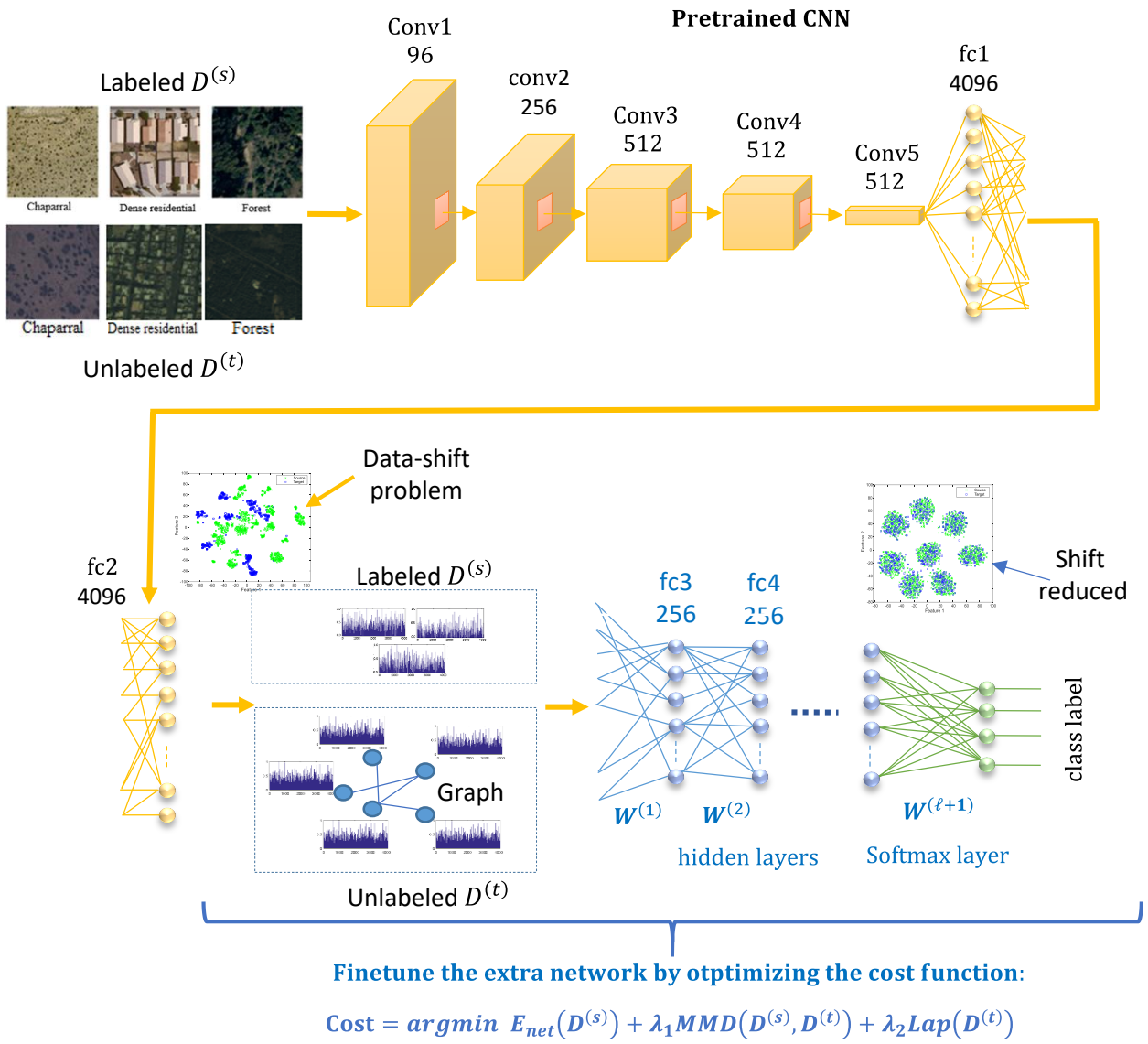


Figure 1. Flowchart of the proposed DAN approach: DAN is composed of a pretrained CNN coupled with an extra network. The weights of the network are leaned by simultaneously optimizing there criteria related to discrimination, shift-between source and target data, and geometrical structure of the target data.

Figure 2. UC Merced dataset: aerial images with spatial resolution of 0.3 m acquired over USA. The dataset contains 21 classes (with 100 images per class).

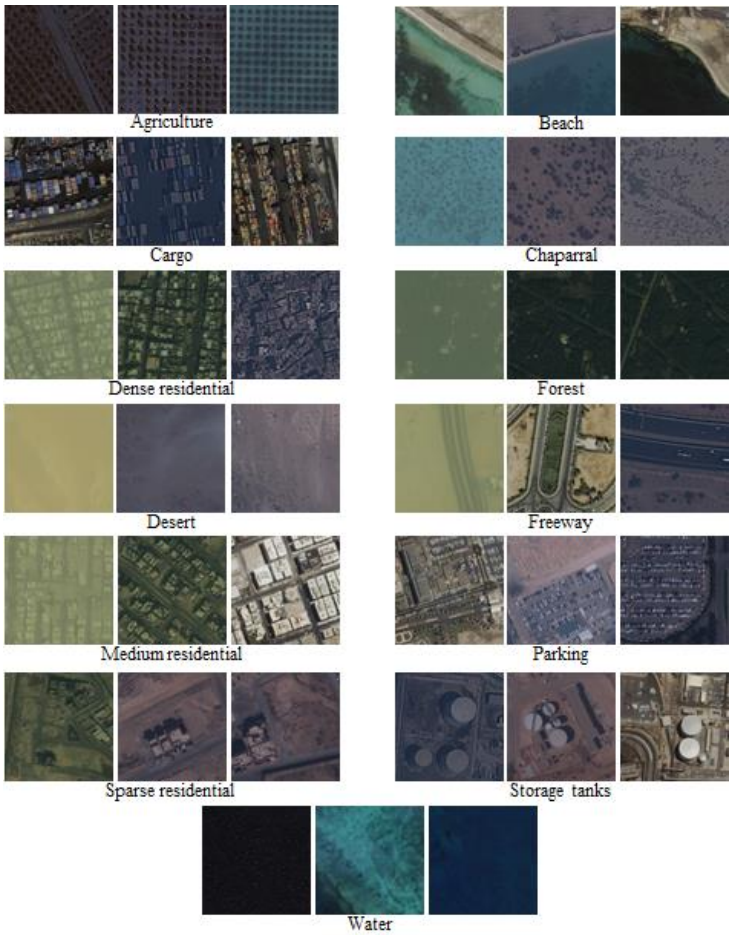
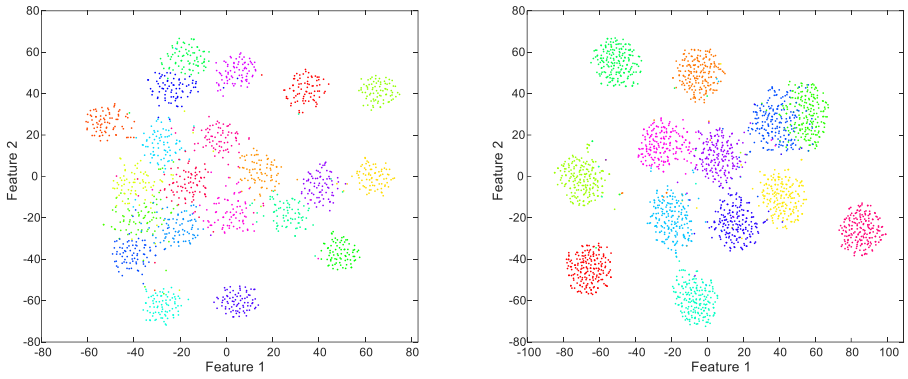
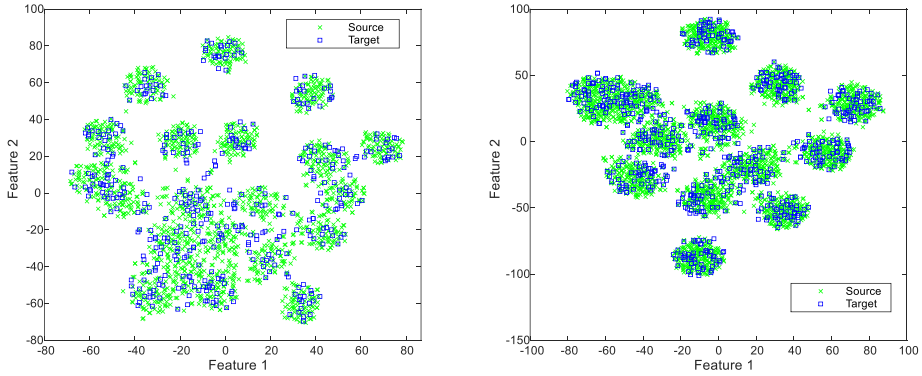


Figure 3. KSA multi-sensor dataset: acquired over KSA, with INKONOS-2, GeoEye-1, and Worldview-1 sensors with spatial resolutions of 1 m, 0.5 m and 0.5 m, respectively. The dataset contains 13 classes (250 images per class).



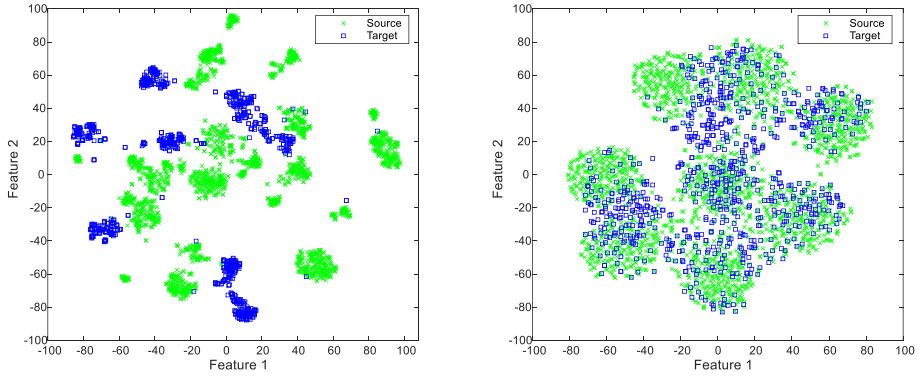
(a) (b)

Figure 4. Visualization of the features learned by the hidden layer of the extra network: (a) Merced (with 21 classes); and (b) KSA (with 13 classes).

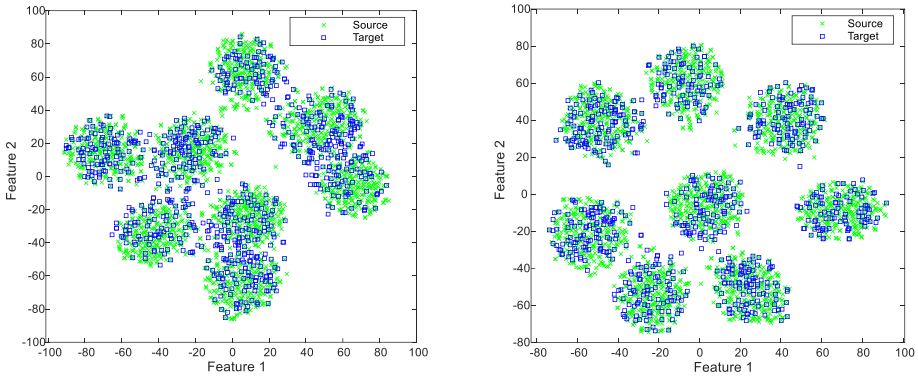


(a) (b)

Figure 5. Feature visualization for: (a) Merced; and (b) KSA datasets. Both figures show that the data- shift problem is not relevant.

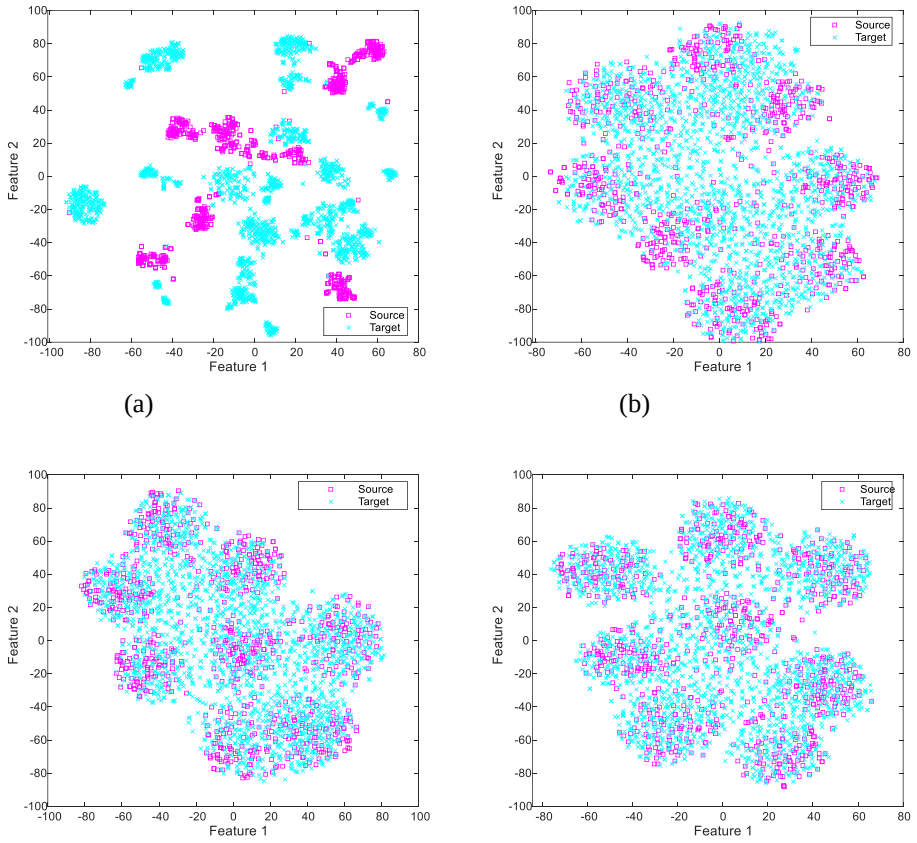


(a) (b)



(c) (d)

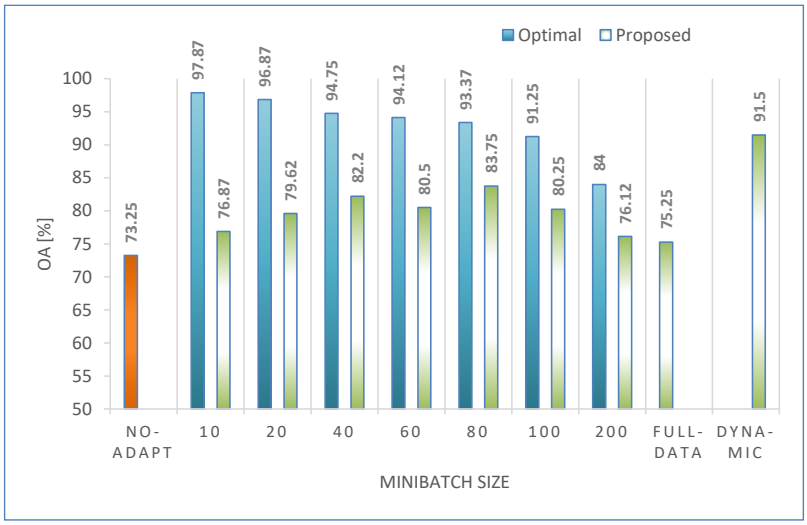
Figure 6. Mismatch reduction of the distributions during the adaptation process for the scenario KSA→Merced: (a) without adaptation (the data shift problem is significant); (b) and (c) intermediate results; and (d) Final result.



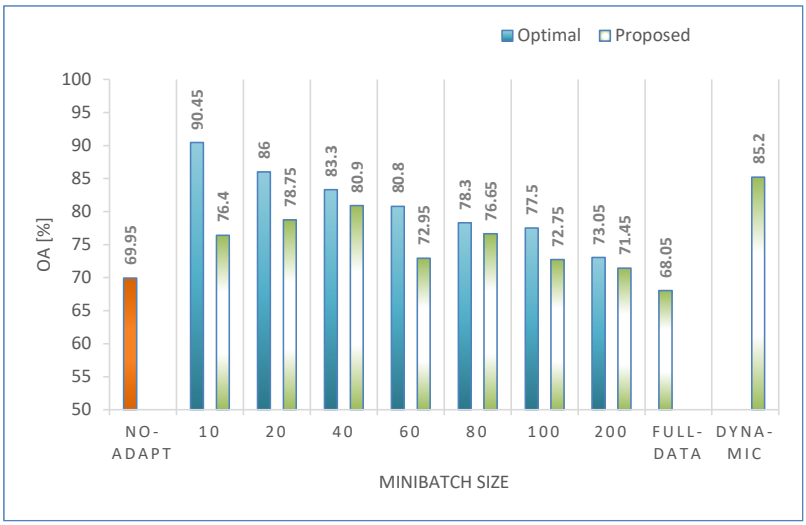
(a) (b)

(c) (d)

Figure 7. Mismatch reduction of the distributions during the adaptation process for the scenario Merced→KSA: (a) without adaptation (the data shift problem is significant); (b) and (c) intermediate results; and (d) Final result.

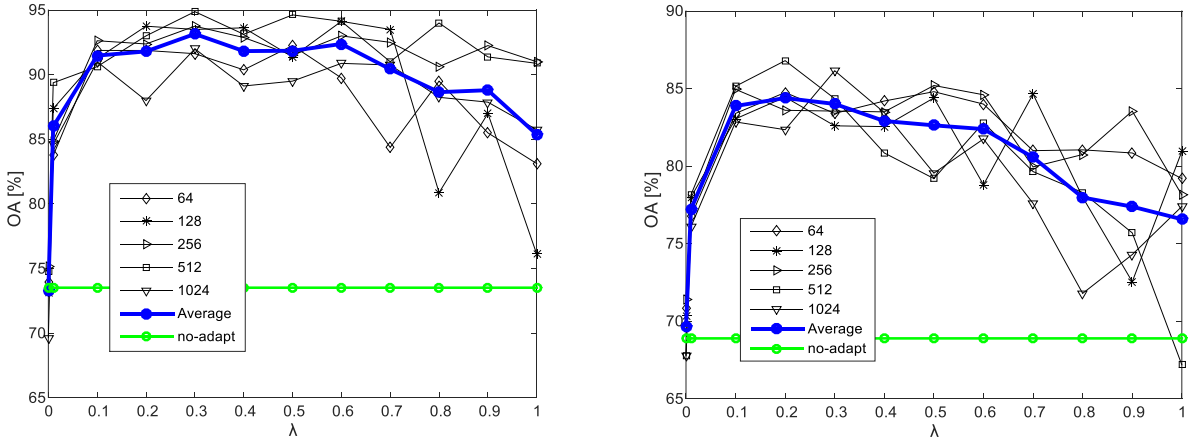


(a)



(b)

Figure 8. Sensitivity analysis with respect to the minibatch size for: (a) KSA→Merced; and (b) Merced→KSA.



(b)

Figure 9. Sensitivity analysis with respect to the regularization parameter λ and the number of *#nodes* in the hidden layer for: (a) KSA→Merced; and (b) Merced→KSA. No-adapt refers to the result obtained without including both MMD and graph regularization terms.

TABLE I
Classification results obtained for Merced dataset.

Method	OA [%]	Time [m]	Validation
Spatial BOVW [1]	81.19	---	5-fold
Sparse Coding [13]	81.70	---	
Bag of SIFT [10]	88.00	---	
Second order visual features [11]	94.30	---	
Pyramid of Spatial Relations [2]	89.10	---	
CS Multifeature Fusion [4]	94.33	20.10	
Salient Unsupervised Learning [9]	82.80	---	
Multiview deep learning [12]	93.48	---	
Gradient boosting RCN [15]	94.53	---	
Pretrained CNN +SAE [10]	97.19	5.60	
Pretrained CNN +SVM	95.09	8.60	
Pretrained CNN +CNN [9]	92.40	---	
DAN (without adaptation)	96.75±0.36	0.81	
DAN (with adaptation)	96.51±0.36	9.01	
Concentric Multiscale BOVW [3]	86.64	---	2-fold
CS Multifeature Fusion [4]	91.10	27.33	
Pretrained CNN +SAE [10]	95.10	4.72	
Pretrained CNN +SVM	92.83	4.45	
DAN (without adaptation)	94.61±0.11	0.54	
DAN (with adaptation)	94.54±0.15	6.06	

TABLE II
Classification results obtained for KSA dataset.

Method	OA [%]	Time [m]	Validation
CS Multifeature Fusion [4]	90.77	57.10	5-fold
Pretrained CNN+SAE [10]	94.92	10.20	
Pretrained CNN+SVM	94.46	12.26	
DAN (no-adaptation)	95.08±0.27	1.27	
DAN (with adaptation)	95.33±0.79	14.50	
CS Multifeature Fusion [4]	91.69	70.31	2-fold
Pretrained CNN+SAE [10]	94.77	9.40	
Pretrained CNN+SVM	94.52	6.86	
DAN (no-adaptation)	94.95±0.53	0.91	
DAN (with adaptation)	94.36±0.41	9.57	

TABLE III
Classification results obtained for KSA→Merced and Merced→KSA datasets.

Cross-dataset	Method	OA [%]	Time [m]
KSA → Merced	Pretrained CNN+SAE [10]	70.35	4.11
	Pretrained CNN+SVM [11]	63.63	5.36
	DAN (no-adaptation)	73.25	0.73
	DAN (with adaptation)	**91.50**	4.07
Merced → KSA	Pretrained CNN+SAE [10]	70.75	4.81
	Pretrained CNN+SVM [11]	66.9	5.21
	DAN (no-adaptation)	69.95	0.27
	DAN (with adaptation)	**85.20**	1.77

TABLE IV
Class-by-class accuracies obtained by DAN for KSA→Merced and Merced→KSA datasets.

Cross-dataset	DAN	Agriculture	Beach	Chaparral	Dense residential	Forest	Freeway	Parking	Storage tanks	OA [%]
KSA→ Merced	no adaptation	43	92	84	59	36	99	93	80	73.25
	with adaptation	**86**	**100**	**100**	**86**	**80**	**99**	**98**	**83**	**91.50**
Merced→ KSA	no adaptation	52	62	78	84.8	65.6	54.4	66.4	96.4	69.95
	with adaptation	**82.8**	**74.4**	**82.4**	**98.8**	**88.4**	**74**	**84.8**	**96**	**85.20**

TABLE V
Sensitivity analysis of DAN with respect to the number of hidden layers for (A) KSA→Merced and (B) Merced→KSA datasets.

(A)

		Size of labeled source images				
ℓ	*#nodes*	25%	50%	100%	200%	Average
	(without adaptation)	65.50±1.31	70.25±1.20	73.25	73.87±0.90	70.72±0.85
1	[256]	76.00±1.37	83.25±1.23	91.50	91.75±0.87	85.63±0.86
2	[256 256]	82.50±1.50	91.00±1.01	93.12	93.75±0.71	90.09±0.80
3	[256 256 256]	**84.50**±1.07	90.25±1.25	93.25	95.50±0.95	90.87±0.81
4	[256 256 256 256]	84.25±1.19	**91.87**±1.07	**93.55**	**95.62**±0.97	**91.32**±0.81

(B)

		Size of labeled source images				
ℓ	*#nodes*	25%	50%	100%	200%	Average
	(without adaptation)	61.35±1.01	67.35±0.90	69.95	71.85±0.81	67.63±0.68
1	[256]	72.45±0.95	75.55±0.75	85.20	86.35±0.67	79.89±0.68
2	[256 256]	76.30±0.84	80.90±0.67	87.25	88.40±0.71	83.21±0.55
3	[256 256 256]	**78.55**±0.82	83.95±0.59	87.45	88.45±0.45	84.60±0.46
4	[256 256 256 256]	77.60±0.97	**85.90**±0.80	**87.95**	**89.40**±0.53	**85.21**±0.57

TABLE VI
Comparison with other domain adaption methods.

Cross-dataset	**Domain Adaptation methods**	**OA [%]**	**Time [m]**
KSA→Merced	TCA [36]	71.10	3.46
	GFK [43]	72.40	1.90
	mSDA [44]	69.90	9.13
	DaNN [49]	75.25	4.80
	DAN (one hidden layer)	**91.50**	4.07
	DAN (two hidden layer)	**93.12**	7.60
	DAN (four hidden layer)	**93.55**	15.21
Merced→ KSA	TCA [36]	68.10	2.00
	GFK [43]	67.25	1.23
	mSDA [44]	71.38	7.25
	DaNN [49]	68.05	2.10
	DAN (one hidden layer)	**85.20**	1.77
	DAN (two hidden layer)	**87.25**	3.10
	DAN (four hidden layer)	**87.95**	8.50