DASH-N: Joint Hierarchical Domain Adaptation and Feature Learning

Hien V. Nguyen, *Member, IEEE*, Huy Tho Ho, *Student Member, IEEE*, Vishal M. Patel, *Member, IEEE*, and Rama Chellappa, *Fellow, IEEE*

***Abstract*— Complex visual data contain discriminative structures that are difficult to be fully captured by any single feature descriptor. While recent work on domain adaptation focuses on adapting a single hand-crafted feature, it is important to perform adaptation of a hierarchy of features to exploit the richness of visual data. We propose a novel framework for domain adaptation using a sparse and hierarchical network (DASH-N). Our method jointly learns a hierarchy of features together with transformations that rectify the mismatch between different domains. The building block of DASH-N is the latent sparse representation. It employs a dimensionality reduction step that can prevent the data dimension from increasing too fast as one traverses deeper into the hierarchy. The experimental results show that our method compares favorably with the competing state-of-the-art methods. In addition, it is shown that a multi-layer DASH-N performs better than a single-layer DASH-N.**

***Index Terms*— Domain adaptation, hierarchical sparse representation, dictionary learning, object recognition.**

I. Introduction

IN MANY practical computer vision applications, we are often confronted with the situation where the data that we use to train classification/regression algorithm has a different distribution or representation with that presented during testing. The ubiquity of this problem is well-known to machine learning and computer vision researchers. This challenge is commonly referred to as *covariate shift* [1], or *class imbalance* [2]. For instance, indoor images are quite different from outdoor images, just as videos captured with a high definition camera are from those collected using a webcam. This detrimental effect is often a dominant factor contributing to the poor performances of many computer vision algorithms. As an example of the effect of distribution mismatch, Ben-David *et al.* [3] show that, under certain assumption, the bound on the test error linearly increases with the ℓ_1 divergence between training and testing distributions. Even worse, data from the test domain are often scarce and expensive to obtain. This makes it impractical to re-train an algorithm from scratch since a learning algorithm would generalize poorly when an insufficient amount of data is presented [4]. Regardless of the cause, any distributional change that may occur after training can degrade the performance of the system when it comes to testing. Domain adaptation also known as domain-transfer learning attempts to minimize this degradation.

The problems caused by domain changes have received substantial attention in recent years. The problem can be informally stated as follows. Given a source domain whose representation or distribution can be different from that of the target domain, how to effectively utilize the model trained on the source data to achieve a good performance on the target data. It is also often assumed that the source domain has sufficient labelled training samples while there are only a few (both labelled and unlabelled) samples available in the target domain. It has been shown in [5]–[9] that domain adaptation techniques can significantly improve the performance of computer vision tasks such as visual object detection and recognition.

Most of the algorithms for adapting a recognition system to a new visual domain share a common architecture containing two main stages. First, features are extracted *separately* for source and target using *hand-crafted* feature descriptors, followed by the second stage where transformations are learned in order to rectify the discrepancy between the two domains. This architecture has several drawbacks. Without any knowledge about the target domain, the feature extraction performed on the source data can ignore information important to the target data. In addition, the process of designing features, such as SIFT [10] or SURF [11], is tedious and time-consuming. It requires a deep understanding and a careful examination of the underlying physics that governs the generation of data. Such requirements might be impractical given that the data from the target domain are often very scarce.

Another issue is that discriminative information can be embedded in multiple levels of the features hierarchy. High-level features are sometimes more useful than low-level ones. In fact, this is one of the main motivations behind the development of hierarchical networks (e.g. [12], [13]) so that more complex abstraction from a visual object can be captured. The traditional framework of domain adaptation employs

Manuscript received December 31, 2014; revised May 21, 2015; accepted September 1, 2015. Date of publication September 22, 2015; date of current version October 13, 2015. This work was supported by XEROX. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. David Frakes. *(Hien V. Nguyen and Huy Tho Ho contributed equally to this work.)*

H. V. Nguyen is with Siemens Corporate Technology, Princeton, NJ 08540 USA (e-mail: hien@umiacs.umd.edu).

H. T. Ho and R. Chellappa are with the Center for Automation Research, Department of Electrical Engineering, University of Maryland Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 USA (e-mail: hhuytho@gmail.com; rama@umiacs.umd.edu).

V. M. Patel is with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08901 USA (e-mail: vishal.m.patel@rutgers.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2015.2479405

a shallow architecture containing a single layer. This ignores the possibility of transferring at multiple levels of the feature hierarchy. In this paper, we show that jointly learning a hierarchy of features could significantly improve the accuracy of cross-domain classification. We compare and contrast our approach with recent work on using deep networks in domain adaptation. Our method uses the latent sparse representation. The network incorporates a dimensionality reduction stage to prevent the feature dimension from increasing too fast as one traverses deeper into the hierarchy. The contributions of our paper are summarized below.

Contributions: In order to address the limitations of existing approaches, we propose a novel approach for domain adaptation that possesses the following advantages:

- Adaptation is performed on multiple levels of the feature hierarchy in order to maximize the knowledge transfer. The hierarchical structure allows the transfer of useful information that might not be well captured by existing domain adaptation techniques.
- Adaptation is done jointly with feature learning. Our method learns a hierarchy of sparse codes and uses them to describe a visual object instead of relying on any low-level feature.
- Unlike existing hierarchical networks, our network is more computationally efficient with a mechanism to prevent the data dimension from increasing too fast as the number of layer increases.

We performed extensive experiments to show that our approach performs better than many current state-of-the-art domain adaptation methods. This is interesting since in our method, training is entirely generative followed by a linear support vector machine while several other methods employ discriminative training together with non-linear kernels. Furthermore, we introduce a new set of data for benchmarking the performance of our algorithm. The new dataset has two domains containing half-toned and edge images, respectively. In order to facilitate future research in the area, a Matlab implementation of our method will be made available.

A. Organization of the Paper

This paper is organized as follows. Related works on domain adaptation are discussed in Section II. The main formulation of DASH-N is given in Section III, followed by the optimization procedure in Section V. Experimental results of domain adaptation for object recognition are presented in Section VI. Finally, Section VII concludes the paper with a brief summary and discussion.

II. Related Works

In this section, we review some related works on domain adaptation and hierarchical feature learning.

A. Domain Adaptation

While domain adaptation was first investigated in speech and natural language processing [14]–[16], it has been studied extensively in other areas such as machine learning [3], [17] and computer vision, especially in the context of visual object recognition [5]–[9], [18]. Domain adaptation for visual recognition was introduced by Saenko *et al.* [5] in a semi-supervised setting. They employed metric learning to learn the domain shift using partially labeled data from the target domain. This work was extended by Kulis *et al.* [6] to handle asymmetric domain transformations. Gopalan *et al.* [7] addressed the problem of unsupervised domain adaptation, where samples from the target domain are unlabeled, by using an incremental approach based on Grassmann manifolds. By formulating a geodesic flow kernel, Gong *et al.* [8] and Zheng *et al.* [19] independently extended the idea of interpolation to integrate an infinite number of subspaces on the geodesic flow from the source domain to the target domain. Chen *et al.* [20] presented a co-training based method that slowly adapted a training set from the source to the target domain. An information-theoretic method for unsupervised domain adaptation was proposed by Shi and Sha [21] that attempted to find a common feature space, where the source and target data distributions are similar and the misclassification error is minimized.

Sparse representation and dictionary-based methods for domain adaptation [22]–[24] are also gaining a lot of traction. In particular, [22] modeled dictionaries across different domains with a parametric mapping function, while [24] enforced different domains to have a common sparse representation on some latent domain. Another class of techniques [25], [26] performed domain adaptation by directly learning a target classifier using classifiers trained on the source domain(s).

A major drawback of some of the existing approaches is that the domain shifting transformation is considered only at a single layer and may not capture adequately the shift between the source and target domains. It is worth noting that although [27] also named their method hierarchical domain adaptation, the paper is not related to ours. They made use of hierarchical Bayesian prior, while we employ a multi-layer network of sparse representation.

There are also some closely related machine learning problems that have been studied extensively, including transfer learning or multi-task learning [28], self-taught learning [29], semi-supervised learning [30] and multiview analysis [31]. A review of domain adaptation methods from machine learning and the natural language processing communities can be found in [32]. A survey on the related field of transfer learning can be found in [17].

B. Hierarchical Feature Learning

Designing features for visual objects is a time-consuming and challenging task that requires a deep understanding of domain knowledge. It is also non-trivial to adapt these manually designed features to new types of data such as hyperspectral or range-scan images. For this reason, learning features from the raw data has become increasingly popular with demonstrated competitive performances on practical computer vision tasks [12], [13], [33]. In order to capture the richness of data, a multi-layer or hierarchical network is employed to learn a spectrum of features, layer by layer.

The design of multi-layer networks has been an active research topic in computer vision. One of the early works includes [34], which used a multistage system to extract salient features in the image at different spatial scales. By learning higher-level feature representations from unlabelled data, deep belief networks (DBN) [12] and its variants, such as convolutional DBNs [13] and deep autoencoders [35], have been shown to be effective when applied to classification problems. Motivated by recent works on deep learning, multi-layer sparse coding networks [33], [36], [37] have been proposed to build feature hierarchies layer by layer using sparse codes and spatial pooling. Each layer in these networks contains a coding step and a pooling step. A dictionary is learned at each coding step which then serves as a codebook for obtaining sparse codes from image patches or pooled features. Spatial pooling schemes, most notably max-pooling, group the sparse codes from adjacent blocks into common entities. This operation makes the resulting features more invariant to certain changes caused by translation and rotation. The pooled sparse codes from one layer serve as the input to the next layer.

Although the high dimension of the feature vectors obtained from a hierarchical network may provide some improvements in classification tasks [12], [13], [33], it may also lead to high redundancy and thus, reduce the efficiency of these algorithms. As a result, it is desirable to have a built-in mechanism in the hierarchy to reduce the dimension of the feature vectors while keeping their discriminative power. Hierarchical feature learning has also been used in domain adaptation such as in [38], [39].

Deep learning has recently made significant improvement to cross-domain classification [38]–[42]. One of the early works [38] uses stacked auto-encoder (SDA) to learn high-level features in an unsupervised manner. They show that deep features improve sentiment classification accuracy on a dataset of 22 different domains. A fast variant of auto-encoder [39] was developed to make SDA training two orders of magnitudes faster than that of the traditional counterpart. Another architecture [43] was based on CNN to learn generic features from a large dataset containing millions of images of over 1000 classes. This work was able to bring down the state-of-the-art error on ImageNet dataset from 26.1% to 15.3%. In addition, the learned features have been shown to generalize well across different domains [40], [41], making them suitable for domain adaptation. The effectiveness of the learned features can be attributed to the large number of training images which probably contains significant information from all interested domains. Another work [42] also makes use of CNN to learn features but with an interesting twist inspired by the work of [7]. In particular, they create a path of interpolated representations by slowly varying the sampling proportion of source and target data. Each representation along the path is generated by applying CNN on the resulting dataset.

Additional data greatly benefit the performance of domain adaptation algorithm. For example, [41] showed that the cross-domain classification accuracies on several popular datasets [5], [44] can be improved by employing a large number of training images from other sources. The improvement can be attributed to the high-capacity learning framework of deep network like CNN. It might also be because the learner has seen sufficient information for different domains from the large number of additional training images. It is understandable that this approach requires a lot of data to perform well. However, data collection is difficult in many practical applications. For instance, medical data are rarely available in abundance like RGB images. For this reason, our paper will focus on comparing with those methods that do not use additional data from external sources.

III. Background

Since our formulation is based on sparse coding and dictionary learning, in this section, we briefly give a background on these topics.

A. Dictionary Learning

Given a set of training samples $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{d \times n}$, the problem of learning a dictionary together with the sparse codes is typically posed as the minimization of the following cost function over ($\mathbf{D}$, $\mathbf{X}$):

$$\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \beta\Psi(\mathbf{X}) \quad \text{s.t. } \|\mathbf{d}_i\|_2 = 1, \quad \forall i \in [1, K] \tag{1}$$

where $\|\mathbf{Y}\|_F$ denotes the Frobenius norm defined as $\|\mathbf{Y}\|_F = \sqrt{\sum_{i,j} |Y_{i,j}|^2}$, $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K] \in \mathbb{R}^{d \times K}$ is the sought dictionary, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{K \times n}$ is the horizontal concatenation of sparse codes, β is a non-negative constant and Ψ promotes sparsity. Various methods have been proposed in the literature for solving such optimization problem. In the case when ℓ_0 norm is enforced, the K-SVD [45] algorithm can be used to train a dictionary. One can also promote sparsity by enforcing the ℓ_1 norm on $\mathbf{X}$. In this case, one can use the algorithm proposed in [46] to solve the above problem. See [45] and [46] for more details.

B. Latent Sparse Representation

From the observation that signals often lie on a low-dimensional manifold, several authors have proposed to perform dictionary learning and sparse coding in a latent space [47]–[49]. We call it *latent sparse representation* to distinguish from the formulation in (1). This is done by minimizing the following cost function over ($\mathbf{P}$, $\mathbf{D}$, $\mathbf{X}$):

$$\begin{aligned} &\mathcal{L}(\mathbf{Y}, \mathbf{P}, \mathbf{D}, \mathbf{X}, \alpha, \beta) \\ &\qquad = \|\mathbf{P}\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \alpha\|\mathbf{Y} - \mathbf{P}^T\mathbf{P}\mathbf{Y}\|_F^2 + \beta\|\mathbf{X}\|_1 \\ &\text{s.t. } \mathbf{P}\mathbf{P}^T = \mathbf{I} \quad \text{and } \|\mathbf{d}_i\|_2 = 1, \quad \forall i \in [1, K], \end{aligned} \tag{2}$$

where $\mathbf{P} \in \mathbb{R}^{p \times d}$ is a linear transformation that brings the data to a low-dimensional feature space ($p < d$). Note that the dictionary is now in the low-dimensional space $\mathbf{D} \in \mathbb{R}^{p \times K}$. The first term of the cost function promotes sparsity of signals in the reduced space. The second term is the amount of energy discarded by the transformation $\mathbf{P}$, or the difference between low-dimensional approximations and the original signals. The minimization of the second term encourages the learned transformation to preserve the useful information present in the original signals. Besides the computational advantage,

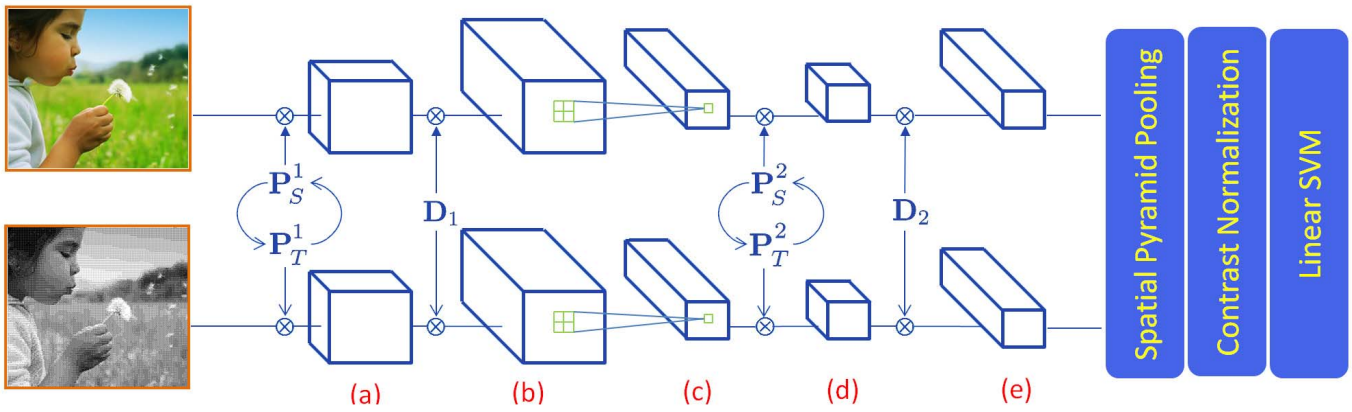


Fig. 1. An illustration of DASH-N algorithm. The source domain is RGB images and the target domain is halftone images. First, images are divided into small overlapping patches. These patches are vectorized while maintaining their spatial arrangements. (a) Performing contrast-normalization and dimensionality reduction using $\mathbf{P}_S$ for source images and $\mathbf{P}_T$ for target images. The circular feedbacks between $\mathbf{P}_S$ and $\mathbf{P}_T$ indicate that these two transformations are learned jointly. (b) Otaining sparse codes using the common dictionary $\mathbf{D}_1$. (c) Performing max pooling. The process then repeats for layer 2 (d & e), except that the input is the sparse codes from layer 1 instead of pixel intensities. At the final stage, spatial pyramid with max pooling are used to create image descriptors. Classification is done using linear support vector machine.

[48] shows that this optimization can recover the underlying sparse representation better than the traditional dictionary learning methods. This formulation is attractive since it allows the transformation of the data into another domain to better handle different sources of variation such as illumination and geometric articulation.

IV. Hierarchical Domain Adaptation

We propose a method to perform hierarchical domain adaptation jointly with feature learning. Figure 1 shows an overview of the proposed method. The network contains multiple layers, each of which contains 3 sub-layers as illustrated in Figure 1. The first sub-layer performs contrast-normalization and dimensionality reduction on the input data. Sparse coding is carried out in the second sub-layer. In the final sub-layer, adjacent features are max-pooled together to produce a new features. Output from one layer becomes the input to the next layer. We note that the hierarchical sparse coding technique was used to extract powerful features for object classification [33]. However, the work in [33] is essentially different from our work in terms of network architecture, learning algorithm, and application. For the simplicity of notation, we consider a single source domain. The extension of DASH-N to multiple source domains is straight forward and is discussed in the Appendix.

Let $\mathbf{Y}_S \in \mathbb{R}^{d_S \times n_S}$ and $\mathbf{Y}_T \in \mathbb{R}^{d_T \times n_T}$ be the input data at each layer from source domain and target domain, respectively. Note that there are n_S, d_S-dimensional samples in the source domain and n_T, d_T-dimensional samples in the target domain. Given $\mathbf{Y}_S$ and $\mathbf{Y}_T$, in each layer of DASH-N, we learn a *joint* latent sparse representation by minimizing the following cost function with respect to $(\mathbf{P}_S, \mathbf{P}_T, \mathbf{D}, \mathbf{X}_S, \mathbf{X}_T)$:

$$\mathcal{L}(\mathbf{Y}_S, \mathbf{P}_S, \mathbf{D}, \mathbf{X}_S, \alpha, \beta) + \lambda \mathcal{L}(\mathbf{Y}_T, \mathbf{P}_T, \mathbf{D}, \mathbf{X}_T, \alpha, \beta) \quad (3)$$

$$\text{s.t. } \mathbf{P}_S \mathbf{P}_S^T = \mathbf{P}_T \mathbf{P}_T^T = \mathbf{I}, \quad \|\mathbf{d}_i\|_2 = 1, \quad \forall i \in [1, K], \quad (4)$$

where (α, β, λ) are the non-negative constants, $\mathbf{D} \in \mathbb{R}^{p \times K}$ is the common dictionary, $\mathbf{P}_S \in \mathbb{R}^{p \times d_S}$ and $\mathbf{P}_T \in \mathbb{R}^{p \times d_T}$ are the transformations to the latent domain, $\mathbf{X}_S \in \mathbb{R}^{K \times n_S}$ and $\mathbf{X}_T \in \mathbb{R}^{K \times n_T}$ are the sparse codes of the source and the target domains, respectively. As can be seen from the above formulation, two domains are forced to share a common dictionary in the latent domain. Together with the sparsity constraint, the common $\mathbf{D}$ provides a coupling effect that promotes the discovery of common structure between the two domains. For simplicity, in what follows, we provide a detailed discussion on a two-layer DASH-N network. Extension of DASH-N to multiple layers is straight forward.

A. Layer 1

We perform dense sampling on each training image to get a set of overlapping patches. These patches are then contrast-normalized. If $\mathbf{f}$ is a vector corresponding to a patch, then the contrast-normalization can be performed as in [37]

$$\hat{\mathbf{f}} = \frac{\mathbf{f}}{\sqrt{\|\mathbf{f}\|^2 + \epsilon}}, \quad (5)$$

where ϵ is some parameter. We set the value of ϵ equal to 0.1 as it is found to work well in our experiments. In order to make the computation more efficient, only a random subset of patches from each image is used for learning the latent sparse representation. We found that setting this number to 150 for images of maximum size of 150×150 provides a good trade-off between accuracy and computational efficiency. After learning the dictionary $\mathbf{D}_1$ and the transformations $(\mathbf{P}_S^1, \mathbf{P}_T^1)$, the sparse codes $(\mathbf{X}_S^1, \mathbf{X}_T^1)$ are computed for all sampled patches by solving the following optimization problem

$$\min_{\mathbf{X}_*^1} \|\mathbf{P}_*^1 \mathbf{Y}_*^1 - \mathbf{D}_1 \mathbf{X}_*^1\|_2^2 + \beta_1 \|\mathbf{X}_*^1\|_1, \quad (6)$$

where $*$ indicates that the above problem can either correspond to source data or target data. Each column of $\mathbf{Y}_*^1$ is the vectorized pixel values inside a patch. A fast implementation of the LARS algorithm is used for solving this optimization problem [46].

Spatial max pooling is used to aggregate the sparse codes over each 4×4 neighborhood as this pooling method is particularly well-suited for the separation of sparse features [50].

B. Layer 2

In this layer, we perform similar computations except that the input is the sparse codes from layer 1 instead of image pixels. The features obtained from the previous layer are aggregated by concatenation over each 4×4 neighborhood and contrast-normalized. This results in a new representation that is more robust to occlusion and illumination. Similar to layer 1, we also randomly sample 150 normalized feature vectors $\hat{\mathbf{f}}$ from each image for training. ℓ_1 optimization is again employed to compute the sparse codes of the normalized features $\hat{\mathbf{f}}$.

After layer 2, the sparse codes are then aggregated using max pooling in a multi-level patch decomposition (i.e. spatial pyramid max pooling). At level 0 of the spatial pyramid, a single feature vector is obtained by performing max pooling over the whole image. At level 1, the image is divided into four quadrants and max pooling is applied to each quadrant, yielding 4 feature vectors. Similarly, for level 2, we obtain 9 feature vectors, and so on. In this paper, max pooling using a three level spatial pyramid is used. As a result, the final feature vector returned by the second layer for each image is a result of concatenating 14 feature vectors from the spatial pyramid.

V. Optimization Procedure

In this section, we describe how the cost function in (3) is minimized. First, let us define

$$\mathbf{K}_S = \mathbf{Y}_S^T\mathbf{Y}_S, \quad \mathbf{K}_T = \mathbf{Y}_T^T\mathbf{Y}_T, \quad \mathbf{K} = \begin{pmatrix} \mathbf{K}_S & \mathbf{0} \\ \mathbf{0} & \sqrt{\lambda}\mathbf{K}_T \end{pmatrix} \tag{7}$$

to be the Gram matrix of source, target, and their block diagonal concatenation, respectively. It can be shown that (see the Appendix) the optimal solution of (3) takes the following form

$$\mathbf{D} = [\mathbf{A}_S^T\mathbf{K}_S, \sqrt{\lambda}\mathbf{A}_T^T\mathbf{K}_T]\mathbf{B} \tag{8}$$

$$\mathbf{P}_S = (\mathbf{Y}_S\mathbf{A}_S)^T, \quad \mathbf{P}_T = (\mathbf{Y}_T\mathbf{A}_T)^T, \tag{9}$$

for some $\mathbf{A}_S \in \mathbb{R}^{n_S \times p}$, $\mathbf{A}_T \in \mathbb{R}^{n_T \times p}$ and $\mathbf{B} \in \mathbb{R}^{(n_S+n_T)\times K}$. Notice that rows of each transformation live in the column subspace of the data from its own domain. In contrast, columns of the dictionary are jointly created by the data of both source and target.

A. Solving for ($\mathbf{A}_S, \mathbf{A}_T$)

The orthogonal constraint in (4) can be re-written using (9) as

$$\mathbf{A}_S^T\mathbf{K}_S\mathbf{A}_S = \mathbf{I}, \quad \mathbf{A}_T^T\mathbf{K}_T\mathbf{A}_T = \mathbf{I}. \tag{10}$$

By substituting (9), (8) into (3) and making use of the orthogonal constraint in (10), the formulation can be simplified as follows (see derivation in the Appendix)

$$\min_{\mathbf{G}} \mathbf{tr}(\mathbf{G}^T\mathbf{H}\mathbf{G}) \quad \text{s.t. } \mathbf{G}_S^T\mathbf{G}_S = \mathbf{G}_T^T\mathbf{G}_T^T = \mathbf{I}, \tag{11}$$

where $\mathbf{H}$ is defined as

$$\mathbf{H} = \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}^T\mathbf{K}((\mathbf{I} - \mathbf{B}\mathbf{X})(\mathbf{I} - \mathbf{B}\mathbf{X})^T - \alpha\mathbf{I})\mathbf{K}\mathbf{V}\mathbf{\Lambda}^{\frac{1}{2}}, \tag{12}$$

$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_S & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_T \end{pmatrix}, \quad \mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_S & \mathbf{0} \\ \mathbf{0} & \sqrt{\lambda}\mathbf{\Lambda}_T \end{pmatrix}, \tag{13}$$

$$\mathbf{K}_S = \mathbf{V}_S\mathbf{\Lambda}_S\mathbf{V}_S^T, \quad \mathbf{K}_T = \mathbf{V}_T\mathbf{\Lambda}_T\mathbf{V}_T^T. \tag{14}$$

Here (14) is given by the eigen-decompositions of the Gram matrices. Finally, $\mathbf{G}$ is defined as

$$\mathbf{G} = [\mathbf{G}_S, \sqrt{\lambda}\mathbf{G}_T], \tag{15}$$

$$\mathbf{G}_S = \mathbf{\Lambda}_S^{\frac{1}{2}}\mathbf{V}_S^T\mathbf{A}_S, \quad \mathbf{G}_T = \mathbf{\Lambda}_T^{\frac{1}{2}}\mathbf{V}_T^T\mathbf{A}_T. \tag{16}$$

The optimization in (11) is non-convex due to the orthogonality constraints. However, $\mathbf{G}$ can be learned efficiently using the algorithm proposed by [51]. Given $\mathbf{G}$, the solution of $(\mathbf{A}_S, \mathbf{A}_T)$ is simply given by

$$\mathbf{A}_S = \mathbf{V}_S\mathbf{\Lambda}_S^{-\frac{1}{2}}\mathbf{G}_S, \quad \mathbf{A}_T = \mathbf{V}_T\mathbf{\Lambda}_T^{-\frac{1}{2}}\mathbf{G}_T. \tag{17}$$

We note that the optimization step involves the eigen-decompositions of large Gram matrices whose dimensions equal to the number of training samples ($\approx 10^5$ in our experiments). This is computationally infeasible. We propose a remedy for this. The source is taken for the illustration purpose and the computation for the target is similar. First, we compute the eigen-decomposition of the following matrix

$$\mathbf{C}_S = \mathbf{Y}_S\mathbf{Y}_S^T = \mathbf{U}_S\mathbf{\Lambda}'_S\mathbf{U}_S^T \in \mathbb{R}^{d_S \times d_S}. \tag{18}$$

Then, the d_S dominant eigenvectors of $\mathbf{K}_S$ can be recovered as

$$\mathbf{V}_S = \mathbf{Y}_S^T\mathbf{U}_S\mathbf{\Lambda}'^{-\frac{1}{2}}_S. \tag{19}$$

The relationship in (19) between $\mathbf{V}_S$ and $\mathbf{U}_S$ can be easily verified using an SVD-decomposition of $\mathbf{Y}_S$.

The signal dimension d_S is much smaller than the number of training samples n_S in our experiments (e.g. 10^3 versus 10^5). The eigen-decomposition of $\mathbf{C}_S$ is therefore much more efficient than that of $\mathbf{K}_S$. Finally, non-zero eigenvalues in $\mathbf{\Lambda_S}$ are given by the diagonal coefficients of $\mathbf{\Lambda}'_S$.

B. Solving for ($\mathbf{B}, \mathbf{X}$)

If we fix $(\mathbf{A}_S, \mathbf{A}_T)$, then learning $(\mathbf{B}, \mathbf{X}_S, \mathbf{X}_T)$ can be done using any dictionary learning algorithm. In order to see this, let us define

$$\mathbf{Z} = [\mathbf{A}_S\mathbf{K}_S, \sqrt{\lambda}\mathbf{A}_T\mathbf{K}_T], \tag{20}$$

$$\mathbf{X} = [\mathbf{X}_S, \sqrt{\lambda}\mathbf{X}_T]. \tag{21}$$

The cost function can be re-written in a familiar form as follows

$$\|\mathbf{Z} - \mathbf{D}\mathbf{X}\|_F^2 + \beta(\|\mathbf{X}_S\|_1 + \lambda\|\mathbf{X}_T\|_1). \tag{22}$$

We use LASSO to solve for the sparse codes $\mathbf{X}$ and the efficient online dictionary learning algorithm [46] to solve for $\mathbf{D}$. The solution of $\mathbf{B}$ can be recovered, using the relationship in (8), simply by

$$\mathbf{B} = \mathbf{Z}^\dagger\mathbf{D},$$

where † denotes the MoorePenrose pseudo-inverse.

Fig. 2. Example images from the LAPTOP-101 class in different domains. First row: original images, second row: halftone images, third row: edge images. (a) Amazon. (b) Caltech. (c) DSLR. (d) Webcam.

It is straight forward to extend the above formulation to handle the case of multiple source domains. Details of derivation for this case are included in the Appendix.

VI. Experiments

The proposed algorithm is evaluated in the context of object recognition using a recent domain adaptation dataset [5], containing 31 classes, with the addition of images from the Caltech-256 dataset [44]. There are 10 common classes between the two datasets (BACKPACK, TOURING-BIKE, CALCULATOR, HEADPHONES, COMPUTER-KEYBOARD, LAPTOP-101, COMPUTER-MONITOR, COMPUTER-MOUSE, COFFEE-MUG, and VIDEO-PROJECTOR) which contain a total of 2533 images. Domain shifts are caused by variations in factors such as pose, lighting, resolution, etc., between images in different domains. Figure 2 shows example images from the LAPTOP-101 class with respect to different domains. We compare our method with state-of-the-art adaptation algorithms such as [5], [7], [8], and [24]. Baseline results obtained using the hierarchical feature learning in [33] by learning the dictionaries separately for the source and target domains without performing domain adaptation are also included. Furthermore, in order to better assess the ability to adapt to a wide range of domains, experimental results are also reported on new images obtained by performing halftoning [52] and edge detection [53] algorithms on images from the datasets in [5] and [45].

A. Experiment Setup

We follow the experimental set-ups of [8]. The results using 10 as well as 31 common classes are reported. In both cases, experiments are performed in 20 random trials for each pair of source and target domains. If the source domain is Amazon or Caltech, 20 samples are used in the training. Otherwise, only 8 training samples are used for DLSR and Webcam. The number of target training samples is always set equal to 3. The remaining images from the target domain in each split are used for testing.

B. Parameter Settings

In our experiments, all images are resized to be no larger than 150×150 with preserved ratio and converted to grayscale. The patch size is set equal to 5×5. The parameter λ is set equal to 4 in order to account for less training samples from the target than that from the source, and α is set equal to 1.5 for all experiments. We also found that using $\beta_{train} = 0.3$ for training and $\beta_{test} = 0.15$ for testing yields the best performance. The same values for these parameters are used for both the first and second layer. A smaller sparsity constant often makes the decoding more stable, thus, leads to more consistent sparse codes. This is similar to the finding in [33]. The number of dictionary atoms is set equal to 200 and 1500 in the first and second layer, respectively. The dimension of the latent domain is set equal to 20 and 750 in the first and second layer, respectively. It is worth noting that the input feature to layer 2 has the dimension of 3200. This results from aggregating sparse codes obtained from the first layer over a 4×4 spatial cell ($4\times4\times200$). By projecting them onto a latent domain of dimension of 750, the computations become more tractable. A three level spatial pyramid, partitioned into 1×1, 2×2, and 3×3, is used to perform the max pooling in the final layer. Linear SVM [54] with the regularization parameter of 10 is employed for classification. It is worth noting that we do not use any part of the testing data in tuning the algorithmic parameters. The sparsity constants such as α, β_{train} and β_{test} are set to the standard values used by many popular sparse learning softwares such as SPAMS [46] and ScSPM [55]. For parameters such as patch size, dictionary size, latent space dimensions and the linear SVM regularization parameter, the findings in [33], [45], and [48] are employed to create a small subset of values and cross-validation on the training data is performed to obtain the optimal settings.

C. Computation Time

It takes an average of 35 minutes to perform the dictionary learning and feature extraction of all training samples using our Matlab implementation on a computer with a 3.8 GHz Intel i7 processor. It takes less than 2 seconds to compute the feature for a test image of size 150×150 using both layers of the hierarchy.

D. Object Recognition

1) 10 Common Classes: The recognition results of different algorithms on 8 pairs of source-target domains are shown in Table I. It can be seen that DASH-N outperforms all compared methods in 7 out of 8 pairs of source-target domains. For pairs such as Caltech-Amazon, Webcam-Amazon, or DSLR-Amazon, we achieve more than 20% improvements over the next best algorithm without feature learning used in the comparison (from 49.5% to 71.6%, 49.4% to 70.4%, and 48.9% to 68.9%, respectively). It is worth noting that while we employ a generative approach for learning the feature, our

TABLE I

RECOGNITION RATES OF DIFFERENT APPROACHES ON FOUR DOMAINS (C: CALTECH, A: AMAZON, D: DSLR, W: WEBCAM). 10 COMMON CLASSES ARE USED. RED COLOR DENOTES THE BEST RECOGNITION RATES. BLUE COLOR DENOTES THE SECOND BEST RECOGNITION RATES

Method	C → A	C → D	A → C	A → W	W → C	W → A	D → A	D → W
Metric [5]	33.7 ± 0.8	35.0 ± 1.1	27.3 ± 0.7	36.0 ± 1.0	21.7 ± 0.5	32.3 ± 0.8	30.3 ± 0.8	55.6 ± 0.7
SGF [7]	40.2 ± 0.7	36.6 ± 0.8	37.7 ± 0.5	37.9 ± 0.7	29.2 ± 0.7	38.2 ± 0.6	39.2 ± 0.7	69.5 ± 0.9
GFK (PLS+PCA) [8]	46.1 ± 0.6	55.0 ± 0.9	39.6 ± 0.4	56.9 ± 1.0	32.8 ± 0.1	46.2 ± 0.6	46.2 ± 0.6	**80.2 ± 0.4**
SDDL [24]	49.5 ± 2.6	76.7 ± 3.9	27.4 ± 2.4	72.0 ± 4.8	29.7 ± 1.9	49.4 ± 2.1	48.9 ± 3.8	72.6 ± 2.1
HMP [34]	67.7 ± 2.3	70.2 ± 5.1	51.7 ± 4.3	70.0 ± 4.2	46.8 ± 2.1	61.5 ± 3.8	64.7 ± 2.0	76.0 ± 4.0
DASH-N (1st layer)	60.3 ± 2.7	79.6 ± 3.1	52.2 ± 2.1	74.1 ± 4.6	45.31 ± 3.7	68.7 ± 2.9	65.9 ± 2.1	76.3 ± 2.3
DASH-N (1st+2nd layers)	**71.6 ± 2.2**	**81.4 ± 3.5**	**54.9 ± 1.8**	**75.5 ± 4.2**	**50.2 ± 3.3**	**70.4 ± 3.2**	**68.9 ± 2.9**	77.1 ± 2.8

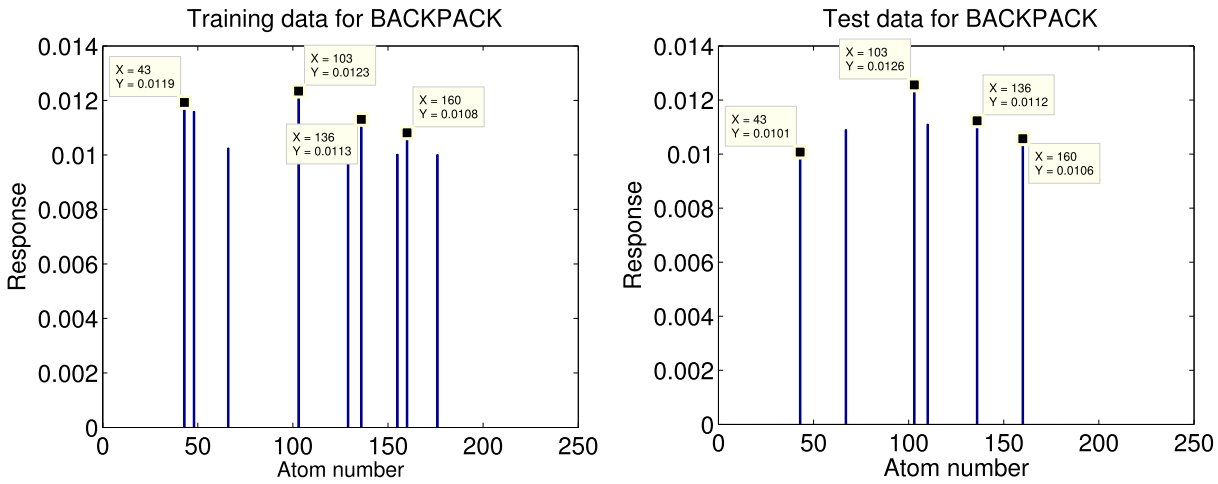


Fig. 3. Dictionary responses of training (left) and testing (right) data for the BACKPACK class for the pair DSLR-Webcam domains in the first layer.

method consistently achieves better performance than [24], which uses discriminative training together with non-linear kernels. It is also clear from the table that the multi-layer DASH-N outperforms the single-layer DASH-N. In the case of adapting from Caltech to Amazon, the performance gain by using a combination of features obtained from both layers rather than just features from the first layer is more than 10% (from 60.3% to 71.6%).

The results obtained by using Hierarchical Matching Pursuit (HMP) [33], without performing domain adaptation, are also included in the comparison in order to better evaluate the improvements provided by the proposed approach. In order to extract features using HMP, a dictionary is learned separately per the source and target domains in each layer. Sparse codes for data from the source and target domains are then computed using the corresponding dictionary. Similar to our approach, the classification is performed using the concatenated features obtained a two-layer network. The HMP parameters are selected using cross-validation on the training data. It can be seen from Table I that, although HMP does not perform as well as the proposed method, it achieves reasonably good performance on the dataset. In many cases, it even outperforms other domain adaptation methods used in the comparison. This demonstrates the effectiveness of learning feature representation. However, it is also clear from the table that by learning a common representation at each layer of the hierarchy, our algorithm is able to capture the domain shift better. As a result, it consistently achieves better classification rates compare to HMP in all scenarios.

In order to illustrate the encoding of features using the learned dictionary in the first layer, Figure 3 shows the responses of the training and testing data for the BACKPACK class with respect to each atom of the dictionary in the first layer for the pair DSLR-Webcam domains. The sparse codes for all the patches of the training and testing images belong to the class are computed. The absolutes of these sparse vectors are summed together and normalized to unit length. Small components of the normalized sparse codes are thresholded to better show the correspondences between the training and testing data. It can be seen from the figure that the sparse codes for the training and testing data for the BACKPACK class both have high responses in four different dictionary atoms (43, 103, 136 and 160).

2) 31 Classes and Multiple Sources: We also compare the recognition results for all 31 classes between our approach and other methods in both cases of single (Table II) and multiple source domains (Table III). It can be seen from Tables II and III that our results, even using only features extracted from the first layer, are consistently better than that of other algorithms in all the domain settings, except the results obtained by using features extracted from deep networks [41], [42]. This proves the effectiveness of the feature learning process using latent sparse representation. It is worth noting that the deep convolutional network used in [41] is

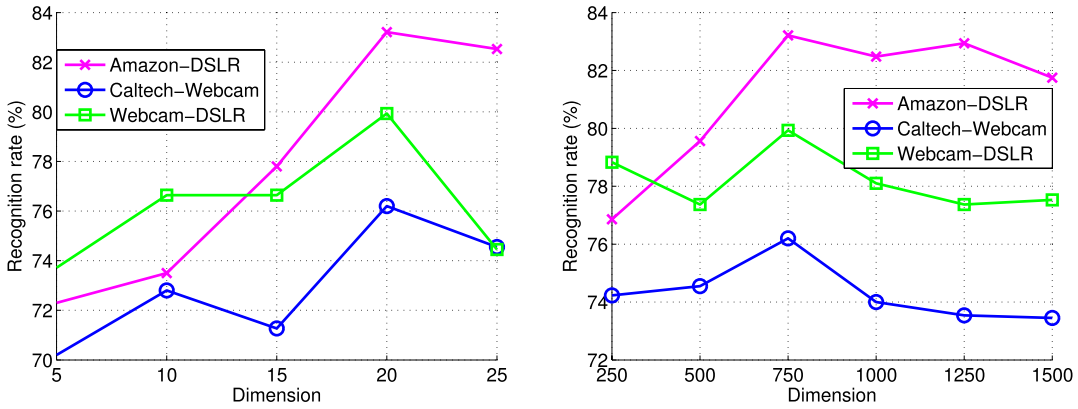


Fig. 4. Recognition rates with respect to different dimensions of the latent domain in the first and second layer. (a) First layer. (b) Second layer.

TABLE II
SINGLE-SOURCE RECOGNITION RATES ON ALL 31 CLASSES

Method	A → W	D → W	W → D
Metric [5]	44	31	27
RDALR [9]	50.7 ± 0.8	36.9 ± 19.9	32.9 ± 1.2
SGF [7]	57 ± 3.5	36 ± 1.1	37 ± 2.3
GFK (PLS+PCA) [8]	46.4 ± 0.5	61.3 ± 0.4	66.3 ± 0.4
SDDL [24]	50.1 ± 2.5	51.2 ± 2.1	50.6 ± 2.6
HMP [34]	55.7 ± 2.5	50.5 ± 2.7	56.8 ± 2.6
SVM + $DeCAF_7$ [42]	79.1 ± 2.1	-	92.9 ± 2.0
DILD (DL-S^2-FT) [43]	44.87	75.21	84.94
DASH-N (1st layer)	59.9 ± 2.7	65.8 ± 1.3	69.6 ± 2.1
DASH-N (1st+2nd layers)	$\mathbf{60.6 \pm 3.5}$	$\mathbf{67.9 \pm 1.1}$	$\mathbf{71.1 \pm 1.7}$

TABLE III
MULTIPLE-SOURCE RECOGNITION RATES ON ALL 31 CLASSES

Method	{D, A} → W	{A, W} → D	{W, D} → A
A-SVM [57]	30.4 ± 0.6	25.3 ± 1.1	17.3 ± 0.9
RDALR [9]	36.9 ± 1.1	31.2 ± 1.3	20.9 ± 0.9
SGF [7]	52 ± 2.5	39 ± 1.1	28 ± 0.8
FDDL [58]	41.0 ± 2.4	38.4 ± 3.4	19.30 ± 1.2
SDDL [24]	57.8 ± 2.4	56.7 ± 2.3	24.1 ± 1.6
HMP [34]	47.2 ± 1.9	51.3 ± 1.4	37.3 ± 1.4
DASH-N (1st layer)	61.7 ± 2.5	64.1 ± 3.5	39.6 ± 1.3
DASH-N (1st+2nd layers)	$\mathbf{64.5 \pm 2.3}$	$\mathbf{68.6 \pm 3.7}$	$\mathbf{41.8 \pm 1.1}$

trained in a fully supervised fashion for more than a week using a very large external dataset, called ImageNet [58], which has millions of images and thousands of classes. In contrast to [41], our method is only trained using a limited number of samples for less than half an hour. While the observation in [41] is interesting, it is also important to deal with the scenarios where there is no abundance of external data like in medical domain. Our experimental results indicate that DASH-N provides the best performances among the approaches not using external data. The extension of DASH-N for using external dataset is under investigation.

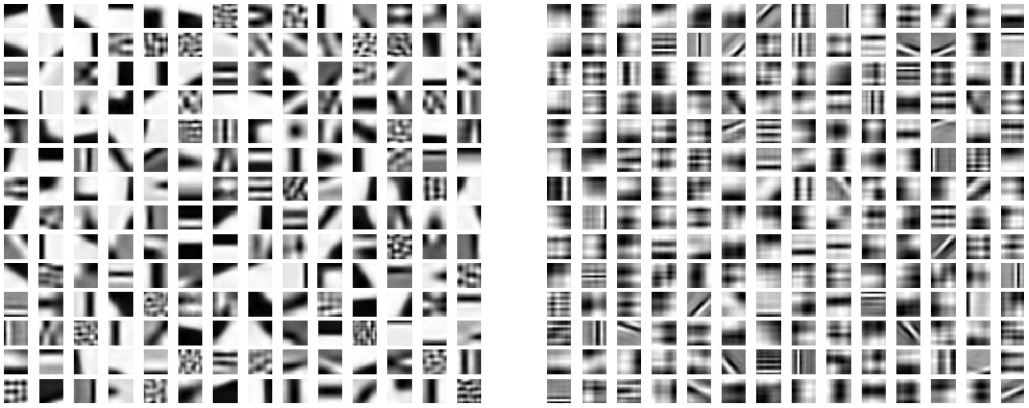
The performance of our algorithm further increases when features learned from both layers of the hierarchy are combined. Especially, in the case of adapting from Webcam and DSLR to Amazon, we achieve an improvement of more than 15% compared to the result of SDDL [24] (from 24.1% to 41.8%). We also want to point to a recent work on domain adaptation using CNN to learn a set of interpolated representations from one domain to another [42]. Their results confirm our findings that hierarchical features make adaptation easier. Our method achieves a better performance on $\{A \rightarrow W\}$ pair (60.6% versus 44.87%) while performing worse on $\{D \rightarrow W\}$ (67.9% versus 75.21%) and $\{W \rightarrow D\}$ (71.1% versus 84.94%) [42].

3) Dimensions of Latent Domains: Dimensions of latent domains are some of the important parameters affecting the performance of DASH-N. Figure 4a shows the recognition rates with respect to different dimensions of the latent domain in the first layer for three pairs of source-target domains (Amazon-DSLR, Caltech-Amazon and Webcam-DSLR), while keeping the dimension of latent domain in the second layer to 750. As the patch size is set at 5×5, we vary the dimension of the first layer dictionary from 5 to 25. It can be seen from the figure that if the latent domain dimension is too low, the accuracy decreases. The optimal dimension is achieved at 20.

Similarly, the recognition rates with respect to different dimensions of the second layer latent domain are shown in Figure 4b while the first layer latent dimension is kept at 20. It can be seen from Figure 4b that the curves for all three pairs of source-target domains peak at the dimension 750. Once again, we observe that the performance decreases if the dimension of the latent domain is too low. More interestingly, as we can observe for the pair Caltech-Webcam and Webcam-DSLR, setting the dimension of the latent domain too high is as detrimental as setting it too low. In all of our experiments, we set the dimension of the latent domain using the cross validation technique.

E. Halftoned and Edge Images

In order to evaluate the ability of DASH-N in adapting to a wide range of domains, we also perform experiments on object recognition from the original image domain to two new domains generated by applying half-toning and edge extraction algorithms to the original images. Half-toning images, which



(a) (b)

Fig. 5. The reconstructed dictionaries at layer 1. (a) Source: Amazon original images. (b) Target: Amazon edge images.

TABLE IV

RECOGNITION RATES OF DIFFERENT APPROACHES ON THE HALF-TONING AND EDGE DATASETS. 10 COMMON CLASSES ARE USED. (a) HALF-TONING. (b) EDGE

(a)

Method	C → A	C → D	A → C	A → W	W → C	W → A	D → A	D → W
kNN	50.1 ± 5.1	41.9 ± 5.5	29.8 ± 4.3	42.9 ± 2.8	28.9 ± 2.5	48.3 ± 2.1	48.6 ± 1.1	41.4 ± 4.1
Metric [5]	41.1 ± 6.4	38.8 ± 5.6	31.9 ± 5.4	49.4 ± 3.4	32.8 ± 3.2	49.9 ± 3.3	43.8 ± 2.6	49.3 ± 2.6
SDDL [24]	52.2 ± 3.9	66.7 ± 5.5	34.1 ± 3.5	69.2 ± 4.2	34.6 ± 2.8	51.2 ± 3.4	54.1 ± 2.7	71.6 ± 5.3
HMP [34]	65.0 ± 5.5	68.7 ± 3.7	44.7 ± 3.1	67.9 ± 3.2	40.3 ± 2.3	59.6 ± 3.9	62.0 ± 4.1	74.7 ± 2.9
DASH-N (1st layer)	68.1 ± 3.1	76.6 ± 3.5	49.7 ± 2.8	85.4 ± 3.1	42.1 ± 3.7	62.7 ± 3.2	66.4 ± 2.1	79.3 ± 2.5
DASH-N (1st+2nd layer)	$\mathbf{70.2 \pm 2.7}$	$\mathbf{79.6 \pm 4.3}$	$\mathbf{52.4 \pm 2.3}$	$\mathbf{86.2 \pm 4.1}$	$\mathbf{43.3 \pm 3.9}$	$\mathbf{66.1 \pm 3.7}$	$\mathbf{67.2 \pm 3.5}$	$\mathbf{80.7 \pm 2.1}$

(b)

Method	C → A	C → D	A → C	A → W	W → C	W → A	D → A	D → W
kNN	50.8 ± 1.8	50.4 ± 1.4	32.8 ± 2.9	47.5 ± 4.2	30.4 ± 3.3	51.9 ± 3.1	48.9 ± 1.8	50.2 ± 2.1
Metric [5]	42.8 ± 2.7	43.8 ± 2.5	35.2 ± 2.1	53.6 ± 1.4	36.8 ± 1.8	53.2 ± 3.1	40.8 ± 3.9	54.5 ± 2.7
SDDL [24]	52.9 ± 5.2	63.8 ± 6.3	32.4 ± 3.2	62.5 ± 5.7	33.5 ± 2.9	55.2 ± 2.8	55.4 ± 3.3	65.3 ± 4.7
HMP [34]	65.1 ± 2.5	61.4 ± 4.9	43.7 ± 4.7	64.2 ± 2.6	37.7 ± 4.9	62.3 ± 6.8	59.4 ± 4.7	70.9 ± 2.7
DASH-N (1st layer)	68.3 ± 4.2	72.9 ± 3.3	33.1 ± 2.1	66.2 ± 4.2	42.6 ± 3.4	59.9 ± 2.7	62.7 ± 2.3	61.5 ± 2.6
DASH-N (1st+2nd layer)	$\mathbf{74.2 \pm 3.9}$	$\mathbf{75.7 \pm 2.6}$	$\mathbf{44.3 \pm 2.5}$	$\mathbf{74.2 \pm 4.5}$	$\mathbf{46.7 \pm 3.1}$	$\mathbf{68.9 \pm 2.2}$	$\mathbf{67.7 \pm 3.4}$	$\mathbf{74.5 \pm 3.2}$

imitate the effect of jet-printing technology in the past, are generated using the dithering algorithm in [52]. Edge images are obtained by applying the Canny edge detector [53] with the threshold set to 0.07.

Figure 5 is the visualization of the reconstructed dictionaries atoms at layer 1 when adapting the original images (source) to edge images (target). Reconstructed dictionaries are obtained by $\hat{\mathbf{D}}_*^1 = (\mathbf{P}_*^1)^\dagger \mathbf{D}_1$, where † denotes the MoorePenrose pseudo-inverse. We observe that the dictionary atoms of original images contain rather fat and smooth regions. In contrast, dictionary atoms of edge images have many thin and highly varying patterns that are more suitable for capturing edges. Table IV shows the performance of different algorithms when adapting to these new domains. It can be seen from the table that DASH-N outperforms other methods used in the comparison in both cases of half-toning and edge images. This proves the ability of our approach to adapt well to new domains. As discussed in previous sections, although even the first layer of DASH-N already achieves very good results on different settings, the performance consistently improves with the addition of the second layer. In many cases, the improvement in the recognition rates can be significant such as in the case of $C \rightarrow A$ in Table I or $A \rightarrow W$ and $D \rightarrow W$ in Table IV which is more than 10%. Both the source code and two new datasets will be released for research purposes.

F. Complexity Analysis

Let w and h be the width and height of an input image, respectively. Recall that d is the dimension of input sample. $K^{(\ell)}$ is the dictionary size at ℓ-layer. q is the number of pixels considered in the pooling operation. $p^{(\ell)}$ is the dimension of the reduced space after the projection at ℓ-layer. The computation needed for evaluating an input image is given as follows

$$\begin{aligned}
&\mathcal{O}\big(w\times h\times d\times p^{(1)} + w\times h\times p^{(1)}\times K^{(1)}\times T^{(1)} \\
&\quad + \frac{w\times h}{q}\times K^{(1)}\times p^{(2)} + \frac{w\times h}{q}\times p^{(2)}\times K^{(2)}\times T^{(2)} \\
&\quad + \frac{w\times h}{q^2}\times K^{(2)}\times p^{(3)} + \frac{w\times h}{q^2}\times p^{(3)}\times K^{(3)}\times T^{(3)}\big) \\
&= \mathcal{O}\Big(\sum_{\ell=1}^{3} \frac{w\times h\times p^{(\ell)}}{q^{(\ell-1)}}\big(K^{(\ell-1)} + K^{(\ell)}\times T^{(\ell)}\big)\Big),
\end{aligned}$$

where we use the convention that $K^{(0)}$ is the equal to the dimension of the input patch d. We also assume that the sparse

coding for each sample could be performed with $d \times K \times T$ computation using OMP method or fast variants of Lasso.

VII. Conclusion

We have presented a hierarchical method for performing domain adaptation using multi-layer representations of images. In the proposed approach, the features and domain shifts are learned jointly in each layer of the hierarchy in order to obtain a better representation of data from different domains. Unlike other hierarchical approaches, our method prevents the dimension of feature vectors from increasing too fast as the number of layers increase. Experimental results show that the proposed approach significantly outperforms other domain adaptation algorithms considered in the comparison.

Several future directions of inquiry are possible considering our new approach to domain adaptation and feature learning. It would also be of interest to incorporate non-linear learning frameworks to DASH-N.

Appendix

In this section, we provide the derivations for the forms of $\mathbf{D}$ in (8) and $\mathbf{P}_i$ in (9) as well as extend the optimization to the case of multiple source domains.

A. Form of $\boldsymbol{D}$

We consider a general case where there are m different domains. Let $\{\mathbf{Y}_i, \mathbf{P}_i, \mathbf{X}_i, n_i\}_{i=1}^m$ be the training data, the transformation, the sparse coefficients, and the number of samples for the i-th domain, respectively. Let $\mathbf{D}$ denote the common dictionary in the latent domain. The objective function that we want to minimize is

$$\sum_{i=1}^{m} \lambda_i \mathcal{L}(\mathbf{Y}_i, \mathbf{P}_i, \mathbf{D}, \mathbf{X}_i, \alpha, \beta) = \sum_i \lambda_i \Big(\|\mathbf{P}_i \mathbf{Y}_i - \mathbf{D}\mathbf{X}_i\|_F^2 + \alpha \|\mathbf{Y}_i - \mathbf{P}_i^T \mathbf{P}_i \mathbf{Y}_i\|_F^2 + \beta \|\mathbf{X}_i\|_1 \Big). \tag{23}$$

For the convenience of notation, we first define

$$\mathbf{Z} = [\sqrt{\lambda_1}\mathbf{P}_1\mathbf{Y}_1, \ldots, \sqrt{\lambda_m}\mathbf{P}_m\mathbf{Y}_m]. \tag{24}$$

One can write $\mathbf{D}$ in the following form

$$\mathbf{D} = \mathbf{D}_{||} + \mathbf{D}_{\perp}, \quad \text{where } \mathbf{D}_{||} = \mathbf{Z}\mathbf{B} \quad \text{and} \quad \mathbf{D}_{\perp}^T \mathbf{Z} = \mathbf{0}, \tag{25}$$

for some $\mathbf{B} \in \mathbb{R}^{(\sum_i n_i) \times K}$. In other words, columns of $\mathbf{D}_{||}$ and $\mathbf{D}_{\perp}$ are in and orthogonal to the column subspace of $\mathbf{Z}$, respectively. Let $\mathbf{S} \in \mathbb{R}^{K \times K}$ be a diagonal matrix with non-negative coefficients such that columns of $\hat{\mathbf{D}}_{||} = \mathbf{D}_{||}\mathbf{S}$ have unit-norm. Since the columns of $\mathbf{D}$ have unit-norm and $\mathbf{D} = \mathbf{D}_{||} + \mathbf{D}_{\perp}$, the columns of $\mathbf{D}_{||}$ must have norms of no larger than 1. Therefore, in order for the columns of $\hat{\mathbf{D}}_{||}$ to have norm 1, the diagonal coefficients in $\mathbf{S}$ must be no less than 1. This gives us the following corollary

$$\|\mathbf{X}_i\|_1 = \|\mathbf{S}\mathbf{S}^{-1}\mathbf{X}_i\|_1 = \|\mathbf{S}\hat{\mathbf{X}}_i\|_1 \geq \|\hat{\mathbf{X}}_i\|_1, \tag{26}$$

where $\hat{\mathbf{X}}_i = \mathbf{S}^{-1}\mathbf{X}_i$. In addition, we also have

$$\begin{aligned} \|\mathbf{P}_i\mathbf{Y}_i - \mathbf{D}\mathbf{X}_i\|_F^2 &= \|\mathbf{P}_i\mathbf{Y}_i - \mathbf{D}_{||}\mathbf{X}_i\|_F^2 + \|\mathbf{D}_{\perp}\mathbf{X}_i\|_F^2 \\ &\geq \|\mathbf{P}_i\mathbf{Y}_i - \mathbf{D}_{||}\mathbf{X}_i\|_F^2 \\ &= \|\mathbf{P}_i\mathbf{Y}_i - (\mathbf{D}_{||}\mathbf{S})(\mathbf{S}^{-1}\mathbf{X}_i)\|_F^2 \\ &= \|\mathbf{P}_i\mathbf{Y}_i - \hat{\mathbf{D}}_{||}\hat{\mathbf{X}}_i\|_F^2. \end{aligned} \tag{27}$$

Using the two inequalities in (26) and (27), we can show that

$$\begin{aligned} &\sum_{i=1}^{m} \lambda_i \mathcal{L}(\mathbf{Y}_i, \mathbf{P}_i, \mathbf{D}, \mathbf{X}_i, \alpha, \beta) \\ &\geq \sum_i \lambda_i \Big(\|\mathbf{P}_i\mathbf{Y}_i - \hat{\mathbf{D}}_{||}\hat{\mathbf{X}}_i\|_F^2 + \alpha \|\mathbf{Y}_i - \mathbf{P}_i^T\mathbf{P}_i\mathbf{Y}_i\|_F^2 + \beta \|\hat{\mathbf{X}}_i\|_1 \Big) \\ &= \sum_{i=1}^{m} \lambda_i \mathcal{L}(\mathbf{Y}_i, \mathbf{P}_i, \hat{\mathbf{D}}_{||}, \hat{\mathbf{X}}_i, \alpha, \beta). \end{aligned} \tag{28}$$

This means that given any feasible solution $(\mathbf{D}, \mathbf{X}_i)$, we can find another feasible solution $(\hat{\mathbf{D}}_{||}, \hat{\mathbf{X}}_i)$, whose dictionary atoms normalized to unit-norm, that does not increase the cost function. Therefore, an optimal solution for $\mathbf{D}$ must be in form of $\hat{\mathbf{D}}_{||}$, which can be generally written as

$$\mathbf{D} = \mathbf{Z}\mathbf{B} = [\sqrt{\lambda_1}\mathbf{P}_1\mathbf{Y}_1, \ldots, \sqrt{\lambda_m}\mathbf{P}_m\mathbf{Y}_m]\mathbf{B}. \tag{29}$$

B. Form of $\boldsymbol{P}_i$

We perform the orthogonal decomposition of $\mathbf{P}_i$ as follows [59]

$$\mathbf{P}_i = \mathbf{P}_{i\perp} + \mathbf{P}_{i||}, \quad \text{where } \mathbf{P}_{i\perp}\mathbf{Y}_i = \mathbf{0} \quad \text{and} \quad \mathbf{P}_{i||} = (\mathbf{Y}_i\mathbf{A}_i)^T. \tag{30}$$

In other words, rows of $\mathbf{P}_{i||}$ and $\mathbf{P}_{i\perp}$ are in and orthogonal to the column subspace of $\mathbf{Y}_i$, respectively. For convenience of notation, let us define

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_1 & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \mathbf{P}_m \end{pmatrix} \tag{31}$$

$$\mathbf{Y} = \begin{pmatrix} \sqrt{\lambda_1}\mathbf{Y}_1 \\ \vdots \\ \sqrt{\lambda_m}\mathbf{Y}_m \end{pmatrix} \tag{32}$$

$$\mathbf{X} = (\sqrt{\lambda_1}\mathbf{X}_1, \ldots, \sqrt{\lambda_m}\mathbf{X}_m). \tag{33}$$

After simple algebraic manipulations, the cost function can be re-written as

$$\begin{aligned} &\sum_{i=1}^{m} \lambda_i \mathcal{L}(\mathbf{Y}_i, \mathbf{P}_i, \mathbf{D}, \mathbf{X}_i, \alpha, \beta) \\ &= \sum_i \lambda_i \Big(\|\mathbf{P}_i\mathbf{Y}_i - \mid \mathbf{D}\mathbf{X}_i\|_F^2 + \alpha \|\mathbf{Y}_i - \mathbf{P}_i^T\mathbf{P}_i\mathbf{Y}_i\|_F^2 + \beta \|\mathbf{X}_i\|_1 \Big) \\ &= \Big(\|\mathbf{P}_{||}\mathbf{Y}(\mathbf{I} - \mathbf{B}\mathbf{X})\|_F^2 + \alpha \mathrm{tr}(\mathbf{Y}^T\mathbf{Y} - \mathbf{P}_{||}\mathbf{Y}\mathbf{Y}^T\mathbf{P}_{||}^T) \\ &\quad + \beta \sum_{i=1}^{m} \lambda_i \|\mathbf{X}_i\|_1 \Big). \end{aligned} \tag{34}$$

Removing all the terms independent of $\mathbf{P}$, we have

$$\text{tr}\Big(\mathbf{P}_{||}\mathbf{Y}((\mathbf{I}-\mathbf{BX})(\mathbf{I}-\mathbf{BX})^T-\alpha\mathbf{I})\mathbf{Y}^T\mathbf{P}_{||}^T\Big).$$

The objective function is independent of $\mathbf{P}_\perp$. Moreover, an optimal solution of $\mathbf{P}_{||}$ is given by the eigenvectors of

$$\mathbf{Y}((\mathbf{I}-\mathbf{BX})(\mathbf{I}-\mathbf{BX})^T-\alpha\mathbf{I})\mathbf{Y}^T.$$

This means $\mathbf{P}_{||}\mathbf{P}_{||}^T=\mathbf{I}$. However, $\mathbf{P}_{||}\mathbf{P}_{||}^T=\mathbf{I}-\mathbf{P}_\perp\mathbf{P}_\perp^T$, therefore, $\mathbf{P}_\perp=\mathbf{0}$. We conclude that an optimal solution of $\mathbf{P}_i$ must have the following form

$$\mathbf{P}_i=(\mathbf{Y}_i\mathbf{A}_i)^T,\quad \forall i\in[1,m]. \tag{35}$$

C. Optimization for Multiple Source Domains

The first term of (23) is

$$\sum_i \lambda_i\|\mathbf{P}_i\mathbf{Y}_i-\mathbf{D}\mathbf{X}_i\|_F^2=\|\mathbf{Z}(\mathbf{I}-\mathbf{BX})\|_F^2, \tag{36}$$

where $\mathbf{X}=[\sqrt{\lambda}_1\mathbf{X}_1,\ldots,\sqrt{\lambda}_m\mathbf{X}_m]$. The second term of (23) can be written as

$$\begin{aligned}\alpha\sum_i\lambda_i\|\mathbf{Y}_i-\mathbf{P}_i^T\mathbf{P}_i\mathbf{Y}_i\|_F^2&=\alpha\sum_i\text{tr}\Big(\mathbf{K}_i-\mathbf{Y}_i^T\mathbf{P}_i^T\mathbf{P}_i\mathbf{Y}_i\Big)\\&=\alpha\ \text{tr}\Big(\sum_i(\mathbf{K}_i)-\mathbf{Z}^T\mathbf{Z}\Big).\end{aligned} \tag{37}$$

After discarding all constant terms, the objective function in (23) is equivalent to

$$\|\mathbf{Z}(\mathbf{I}-\mathbf{BX})\|_F^2-\alpha\ \text{tr}(\mathbf{Z}^T\mathbf{Z})+\beta\sum_i\lambda_i\|\mathbf{X}_i\|_1. \tag{38}$$

Solving for $\boldsymbol{A}_i$: First, we perform the eigen-decomposition $\mathbf{K}_i=\mathbf{V}_i\mathbf{\Lambda}_i\mathbf{V}_i^T$. Then the eigen-decomposition of $\mathbf{K}$ is given by

$$\mathbf{K}=\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{39}$$

where,

$$\mathbf{V}=\begin{pmatrix}\mathbf{V}_1&\ldots&\mathbf{0}\\\vdots&\ddots&\vdots\\\mathbf{0}&\ldots&\mathbf{V}_m\end{pmatrix} \tag{40}$$

$$\mathbf{\Lambda}=\begin{pmatrix}\sqrt{\lambda}_1\mathbf{\Lambda}_1&\ldots&\mathbf{0}\\\vdots&\ddots&\vdots\\\mathbf{0}&\ldots&\sqrt{\lambda}_m\mathbf{\Lambda}_m\end{pmatrix}. \tag{41}$$

Moreover, let us define $\mathbf{G}_i=\mathbf{\Lambda}_i^{\frac{1}{2}}\mathbf{V}_i^T\mathbf{A}_i$. Then, the constraints become

$$\mathbf{P}_i\mathbf{P}_i^T=\mathbf{A}_i^T\mathbf{K}_i\mathbf{A}_i=\mathbf{G}_i^T\mathbf{G}_i=\mathbf{I}.$$

In order to solve for $\mathbf{A}_i$, we assume that $(\mathbf{B},\mathbf{X}_i)$ are fixed. After removing all the terms independent of $\mathbf{A}_i$, and using (24) together with (39), the objective in (38) is equivalent to

$$\begin{aligned}&\|\mathbf{Z}(\mathbf{I}-\mathbf{BX})\|_F^2-\alpha\ \text{tr}(\mathbf{Z}^T\mathbf{Z})\\&=\text{tr}\Big(\mathbf{Z}((\mathbf{I}-\mathbf{BX})(\mathbf{I}-\mathbf{BX})^T-\alpha\mathbf{I})\mathbf{Z}^T\Big)\\&=\text{tr}\Big(\mathbf{A}^T\mathbf{K}((\mathbf{I}-\mathbf{BX})(\mathbf{I}-\mathbf{BX})^T-\alpha\mathbf{I})\mathbf{K}\mathbf{A}\Big)\\&=\text{tr}\Big((\mathbf{A}^T\mathbf{V}\mathbf{\Lambda}^{\frac{1}{2}})(\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}^T((\mathbf{I}-\mathbf{BX})(\mathbf{I}-\mathbf{BX})^T-\alpha\mathbf{I})\mathbf{V}\mathbf{\Lambda}^{\frac{1}{2}})\\&\qquad\times(\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}^T\mathbf{A})\Big)\\&=\text{tr}(\mathbf{G}^T\mathbf{H}\mathbf{G}),\end{aligned} \tag{42}$$

where,

$$\begin{aligned}\mathbf{G}&=\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}^T\mathbf{A}\\&=\begin{pmatrix}\mathbf{V}_1&\ldots&\mathbf{0}\\\vdots&\ddots&\vdots\\\mathbf{0}&\ldots&\mathbf{V}_m\end{pmatrix}\begin{pmatrix}\sqrt{\lambda}_1\mathbf{\Lambda}_1^T&\ldots&\mathbf{0}\\\vdots&\ddots&\vdots\\\mathbf{0}&\ldots&\sqrt{\lambda}_m\mathbf{\Lambda}_m^T\end{pmatrix}\\&\quad\begin{pmatrix}\mathbf{A}_1\\\vdots\\\mathbf{A}_m\end{pmatrix}\\&=[\sqrt{\lambda}_1\mathbf{G}_1,\ldots,\sqrt{\lambda}_m\mathbf{G}_m].\end{aligned} \tag{43}$$

Then the solution of $\mathbf{A}_i$ can be obtained by first minimizing

$$\min_{\mathbf{G}}\ \text{tr}(\mathbf{G}^T\mathbf{H}\mathbf{G})\quad\text{s.t. }\mathbf{G}_i^T\mathbf{G}_i=\mathbf{I}. \tag{44}$$

This can be solved efficiently in the same way as for the case of two domains using the algorithm proposed by [51]. The solution for $\mathbf{A}_i$ of each domain is recovered simply by

$$\mathbf{A}_i=\mathbf{V}_i\mathbf{\Lambda}_i^{-\frac{1}{2}}\mathbf{G}_i. \tag{45}$$

Solving for $(\boldsymbol{B},\boldsymbol{X}_i)$: We now assume that $\mathbf{A}_i$ are fixed. After discarding the terms independent of $(\mathbf{B},\mathbf{X}_i)$ in (38), the objective function is re-written as

$$\|\mathbf{Z}-\mathbf{D}\mathbf{X}\|_F^2+\beta\sum_i\lambda_i\|\mathbf{X}_i\|_1. \tag{46}$$

This is in the familiar form of dictionary learning problem. We use the online dictionary learning algorithm proposed by [46] to learn $(\mathbf{D},\mathbf{X}_i)$. The sparse coding is done using the LASSO algorithm. After obtaining $\mathbf{D}$, the solution of $\mathbf{B}$ is obtained by

$$\mathbf{B}=\mathbf{Z}^\dagger\mathbf{D}. \tag{47}$$

REFERENCES

[1] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. Statist. Planning Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[2] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–450, 2002.

[3] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, no. 1, pp. 151–175, 2010.

[4] C. J. Stone, "Optimal global rates of convergence for nonparametric regression," *Ann. Statist.*, vol. 10, no. 4, pp. 1040–1053, 1982.

[5] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. 11th ECCV*, 2010, pp. 213–226.

[6] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *Proc. IEEE Conf. CVPR*, Jun. 2011, pp. 1785–1792.

[7] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Proc. IEEE ICCV*, Nov. 2011, pp. 999–1006.

[8] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. CVPR*, Jun. 2012, pp. 2066–2073.

[9] I.-H. Jhuo, D. Liu, D. T. Lee, and S.-F. Chang, "Robust visual domain adaptation with low-rank reconstruction," in *Proc. IEEE Conf. CVPR*, Jun. 2012, pp. 2168–2175.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. 26th Annu. ICML*, 2009, pp. 609–616.

[14] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Comput. Speech Lang.*, vol. 9, no. 2, pp. 171–185, 1995.

[15] H. Daumé, III, and D. Marcu, "Domain adaptation for statistical classifiers," *J. Artif. Intell. Res.*, vol. 26, no. 1, pp. 101–126, 2006.

[16] H. Daumé, III, "Frustratingly easy domain adaptation," in *Proc. ACL*, 2007, pp. 256–263.

[17] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.

[18] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 53–69, May 2015.

[19] J. Zheng, M.-Y. Liu, R. Chellappa, and P. J. Phillips, "A Grassmann manifold-based domain adaptation approach," in *Proc. 21st ICPR*, Nov. 2012, pp. 2095–2099.

[20] M. Chen, K. Q. Weinberger, and J. Blitzer, "Co-training for domain adaptation," in *Proc. Adv. NIPS*, 2011, pp. 2456–2464.

[21] Y. Shi and F. Sha, "Information-theoretical learning of discriminative clusters for unsupervised domain adaptation," in *Proc. 29th ICML*, 2012, pp. 1079–1086.

[22] Q. Qiu, V. M. Patel, P. Turaga, and R. Chellappa, "Domain adaptive dictionary learning," in *Proc. 12th ECCV*, 2012, pp. 631–645.

[23] J. Ni, Q. Qiu, and R. Chellappa, "Subspace interpolation via dictionary learning for unsupervised domain adaptation," in *Proc. IEEE Conf. CVPR*, Jun. 2013, pp. 692–699.

[24] S. Shekhar, V. M. Patel, H. V. Nguyen, and R. Chellappa, "Generalized domain-adaptive dictionaries," in *Proc. IEEE Conf. CVPR*, Jun. 2013, pp. 361–368.

[25] L. Duan, I. W. Tsang, D. Xu, and T.-S. Chua, "Domain adaptation from multiple sources via auxiliary classifiers," in *Proc. 26th Annu. ICML*, 2009, pp. 289–296.

[26] L. Duan, I. W. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 465–479, Mar. 2012.

[27] J. R. Finkel and C. D. Manning, "Hierarchical Bayesian domain adaptation," in *Proc. NAACL*, 2009, pp. 602–610.

[28] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[29] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 759–766.

[30] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[31] A. Sharma, A. Kumar, H. Daumé, III, and D. W. Jacobs, "Generalized multiview analysis: A discriminative latent space," in *Proc. IEEE Conf. CVPR*, Jun. 2012, pp. 2160–2167.

[32] J. Jiang, "Domain adaptation in natural language processing," Ph.D. dissertation, Univ. Illinois Urbana-Champaign, Champaign, IL, USA, 2008.

[33] L. Bo, X. Ren, and D. Fox, "Hierarchical matching pursuit for image classification: Architecture and fast algorithms," in *Proc. Adv. NIPS*, 2011, pp. 2115–2123.

[34] B. S. Manjunath and R. Chellappa, "A unified approach to boundary perception: Edges, textures, and illusory contours," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 96–108, Jan. 1993.

[35] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th ICML*, 2008, pp. 1096–1103.

[36] K. Yu, Y. Lin, and J. Lafferty, "Learning image representations from the pixel level via hierarchical sparse coding," in *Proc. IEEE Conf. CVPR*, Jun. 2013, pp. 1713–1720.

[37] L. Bo, X. Ren, and D. Fox, "Multipath sparse coding using hierarchical matching pursuit," in *Proc. IEEE Conf. CVPR*, Jun. 2013, pp. 660–667.

[38] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proc. 28th ICML*, 2011, pp. 513–520.

[39] M. Chen, Z. Xu, K. Q. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," in *Proc. 29th ICML*, 2012, pp. 1–8.

[40] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and T. Darrell. (Dec. 2013). "One-shot adaptation of supervised deep convolutional models." [Online]. Available: http://arxiv.org/abs/1312.6204

[41] J. Donahue *et al.* (Oct. 2013). "DeCAF: A deep convolutional activation feature for generic visual recognition." [Online]. Available: http://arxiv.org/abs/1310.1531

[42] S. Chopra, S. Balakrishnan, and R. Gopalan, "DLID: Deep learning for domain adaptation by interpolating between domains," in *Proc. ICML Workshop Challenges Represent. Learn.*, 2013, pp. 1–8.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[44] G. Griffin, A. Holub, and P. Perona. (2007). "Caltech-256 object category dataset," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. 7694. [Online]. Available: http://authors.library.caltech.edu/7694

[45] M. Elad, *Sparse and Redundant Representations*. New York, NY, USA: Springer-Verlag, 2010.

[46] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. 26th ICML*, 2009, pp. 689–696.

[47] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 791–804, Apr. 2012.

[48] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Sparse embedding: A framework for sparsity promoting dimensionality reduction," in *Proc. 12th ECCV*, 2012, pp. 414–427.

[49] V. M. Patel, H. V. Nguyen, and R. Vidal, "Latent space sparse subspace clustering," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 225–232.

[50] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. 27th ICML*, 2010, pp. 1–8.

[51] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Math. Program.*, vol. 142, no. 1, pp. 397–434, 2013.

[52] V. Monga, N. Damera-Venkata, H. Rehman, and B. L. Evans. (2005). *Halftoning Toolbox for MATLAB*. [Online]. Available: http://users.ece.utexas.edu/~bevans/projects/halftoning/toolbox/

[53] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Nov. 1986.

[54] V. N. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.

[55] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. CVPR*, Jun. 2009, pp. 1794–1801.

[56] J. Yang, R. Yan, and A. G. Hauptmann, "Cross-domain video concept detection using adaptive SVMs," in *Proc. 15th Int. Conf. MM*, 2007, pp. 188–197.

[57] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Fisher discrimination dictionary learning for sparse representation," in *Proc. IEEE ICCV*, Nov. 2011, pp. 543–550.

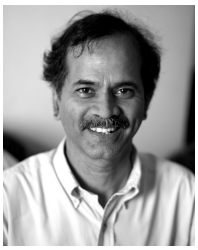
[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.

[59] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2012.

Hien V. Nguyen (M'08) received the bachelor's degree in electrical and computer engineering from the National University of Singapore, and the Ph.D. degree in electrical and computer engineering from the University of Maryland at College Park. He is currently a Research Scientist with Siemens Corporate Technology, Princeton. His current research interests are in machine learning and pattern recognition algorithms for medical imaging and healthcare applications.

Huy Tho Ho (S'07) received the B.Eng. (Hons.) degree in computer systems engineering and the M.App.Sc. degree in electrical and electronic engineering from the University of Adelaide, Australia, in 2007 and 2009, respectively, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Maryland (UMD), College Park, in 2013 and 2014, respectively. His research interests include computer vision, machine learning, and statistical pattern recognition. He received the Adelaide Achiever Scholarship International for his undergraduate study at the University of Adelaide, and the Clark School Distinguished Graduate Fellowship at UMD.

Vishal M. Patel (M'01) received the B.S. (Hons.) degrees in electrical engineering and applied mathematics and the M.S. degree in applied mathematics from North Carolina State University, Raleigh, NC, USA, in 2004 and 2005, respectively, and the Ph.D. degree in electrical engineering from the University of Maryland, College Park, MD, USA, in 2010. He was a member of the Research Faculty with the University of Marylands Institute for Advanced Computer Studies, College Park, MD, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Rutgers University. His current research interests include signal processing, computer vision, and pattern recognition with applications in biometrics and imaging. He is a member of Eta Kappa Nu, Pi Mu Epsilon, and Phi Beta Kappa. He was a recipient of the ORAU Post-Doctoral Fellowship in 2010.



Rama Chellappa (F'92) is currently a Minta Martin Professor of Engineering and the Chair of the ECE Department with the University of Maryland (UMD). He is a fellow of IAPR, OSA, AAAS, ACM, and AAAI and holds four patents. He received the K.S. Fu Prize from the International Association of Pattern Recognition. He is a recipient of the Society, Technical Achievement, and Meritorious Service Awards from the IEEE Signal Processing Society and four IBM Faculty Development Awards. He also received the Technical Achievement and Meritorious Service Awards from the IEEE Computer Society. At UMD, he received college and university level recognitions for research, teaching, innovation, and mentoring of undergraduate students. In 2010, he was recognized as an Outstanding ECE by Purdue University. He served as the Editor-in-Chief of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He is a Golden Core Member of the IEEE Computer Society, and served as a Distinguished Lecturer of the IEEE Signal Processing Society and the President of IEEE Biometrics Council.