Deep Hashing Network for Unsupervised Domain Adaptation

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, Sethuraman Panchanathan
Center for Cognitive Ubiquitous Computing, Arizona State University, Tempe, AZ, USA
{hemanthv, jeusebio, shayok.chakraborty, panch}@asu.edu

Abstract

In recent years, deep neural networks have emerged as a dominant machine learning tool for a wide variety of application domains. However, training a deep neural network requires a large amount of labeled data, which is an expensive process in terms of time, labor and human expertise. Domain adaptation or transfer learning algorithms address this challenge by leveraging labeled data in a different, but related source domain, to develop a model for the target domain. Further, the explosive growth of digital data has posed a fundamental challenge concerning its storage and retrieval. Due to its storage and retrieval efficiency, recent years have witnessed a wide application of hashing in a variety of computer vision applications. In this paper, we first introduce a new dataset, Office-Home, to evaluate domain adaptation algorithms. The dataset contains images of a variety of everyday objects from multiple domains. We then propose a novel deep learning framework that can exploit labeled source data and unlabeled target data to learn informative hash codes, to accurately classify unseen target data. To the best of our knowledge, this is the first research effort to exploit the feature learning capabilities of deep neural networks to learn representative hash codes to address the domain adaptation problem. Our extensive empirical studies on multiple transfer tasks corroborate the usefulness of the framework in learning efficient hash codes which outperform existing competitive baselines for unsupervised domain adaptation.

1. Introduction

Deep learning algorithms automatically learn a discriminating set of features and have depicted commendable performance in a variety of computer vision applications. Unfortunately, training a deep model necessitates a large volume of labeled data, which can be time consuming and expensive to acquire. However, labeled data from a different, but related domain is often available, which has motivated the development of algorithms which can leverage labeled data in a source domain to develop a machine learning model for the target domain. Learning a discriminative model in the presence of the shift between training and test distributions is known as transfer learning or domain adaptation [17]. Unsupervised domain adaptation is a challenging setting, where labeled data is available only in the source domain; no labeled data is available in the target domain. Conventional shallow transfer learning methods develop their models in two stages, feature extraction followed by domain adaptation. The features are fixed and then a model is trained to align the source and target domains [16, 20, 33, 38, 42, 43, 44]. On the other hand, deep transfer learning procedures exploit the feature learning capabilities of deep networks to learn transferable feature representations for domain adaptation and have demonstrated impressive empirical performance [17, 18, 31, 34, 46].

The explosive growth of digital data in the modern era has posed fundamental challenges regarding their storage, retrieval and computational requirements. Against this backdrop, hashing has emerged as one of the most popular and effective techniques due to its fast query speed and low memory cost [48]. Hashing techniques transform high dimensional data into compact binary codes and generate similar binary codes for similar data items. Motivated by this fact, we propose to train a deep neural network to output binary hash codes (instead of probability values), which can be used for classification. We see two advantages to estimating a hash value instead of a standard probability vector in the final layer of the network: (i) the hash values are used to develop a unique loss function for target data in the absence of labels and (ii) during prediction, the hash value of a test sample can be compared against the hash values of the training samples to arrive at a more robust category prediction.

In this paper, we first introduce a new dataset, Office-Home, which we use to evaluate our algorithm. The Office-Home dataset is an object recognition dataset which contains images from 4 domains. It has around 15,500 images organized into 65 categories. We further propose a novel deep learning framework called Domain Adaptive Hashing (DAH) to learn informative hash codes to address the problem of unsupervised domain adaptation. We propose
a unique loss function to train the deep network with the following components: (i) supervised hash loss for labeled source data, which ensures that source samples belonging to the same class have similar hash codes; (ii) unsupervised entropy loss for unlabeled target data, which imposes each target sample to align closely with exactly one of the source categories and be distinct from the other categories and (iii) a loss based on multi-kernel Maximum Mean Discrepancy (MK-MMD), which seeks to learn transferable features within the layers of the network to minimize the distribution difference between the source and target domains. Figure 1 illustrates the different layers of the DAH and the components of the loss function.

2. Related Work

There have been many approaches to address the problem of domain-shift in unsupervised domain adaptation. One straightforward approach is, to modify a classifier trained for the source data by adapting it to classify target data [1, 4] or learn a transformation matrix to linearly transform the source data, so that it is aligned with the target [27, 42]. Some other procedures re-weight the data points in the source domain, to select source data that is similar to the target, when training a domain adaptive classifier, [9, 10, 19]. A standard procedure to reduce domain discrepancy is, to project the source and target data to a common subspace, thereby aligning their principal axes [16, 44]. Reducing domain disparity through nonlinear alignment of data has been possible with Maximum Mean Discrepancy (MMD) - a measure that provides the distribution difference between two datasets in a reproducing-kernel Hilbert space [13]. Kernel-PCA based methods apply the MMD to achieve nonlinear alignment of domains [32, 33, 38]. Manifold based approaches are also popular in domain adaptation for computer vision, where the subspace of a domain is treated as a point on the manifold and transformations are learned to align two domains [20, 23]. A survey of popular domain adaptation techniques for computer vision is provided in [41] and a more generic survey of transfer learning approaches can be found in [39].

All of the above techniques can be termed as shallow learning procedures, since the models are learned using predetermined features. In recent years deep learning has become very successful at learning highly discriminative features for computer vision applications [7, 11, 14]. The closest work incorporating hashing and adaptation appears in cross-modal hashing, where deep hashing techniques embed multi-modal data and learn hash codes for two related domains, like text and images [5, 6, 29]. However, these algorithms are not unsupervised and they are mainly applied to extract common hash codes for multi-modal data for retrieval purposes. To the best of our knowledge, there has been no work in unsupervised domain adaptation using deep hashing networks. We now present the Domain Adaptive Hashing (DAH) network for unsupervised domain adaptation through deep hashing.

3. Domain Adaptive Hashing Networks

In unsupervised domain adaptation, we consider data from two domains; source and target. The source consists of labeled data, \( D_s = \{ x^s_i, y^s_i \}_{i=1}^{n_s} \) and the target has only unlabeled data \( D_t = \{ x^t_i \}_{i=1}^{n_t} \). The data points \( x^s_i \) belong to \( X \), where \( X \) is some input space. The corresponding labels are represented by \( y^s_i \in Y := \{ 1, \ldots, C \} \). The paradigm of domain adaptive learning attempts to address the problem of domain-shift in the data, where the data distributions of the source and target are different, i.e. \( P_s(X,Y) \neq P_t(X,Y) \). The domain-shift notwithstanding, our goal is to train a deep neural network classifier \( \psi(\cdot) \), that can predict the labels \( \{ y^t_i \}_{i=1}^{n_t} \), for the target data.

We implement the neural network as a deep CNN which consists of 5 convolution layers conv1 - conv5 and 3 fully connected layers fc6 - fc8 followed by a loss layer. In our model, we introduce a hashing layer hash-fc8 in place of the standard fc8 layer to learn a binary code \( h_x \), for every data point \( x_i \), where \( h_x \in \{-1, +1\}^d \). The hash-fc8 layer is driven by two loss functions, (i) supervised hash loss for the source data, (ii) unsupervised entropy loss for the target data. The supervised hash loss ensures hash values that are distinct and discriminatory, i.e. if \( x_i \) and \( x_j \) belong to the same category, their hash values \( h_x \) and \( h_j \) are similar and different otherwise. The unsupervised entropy loss aligns the target hash values with source hash values based on the similarity of their feature representations. The output of the network is represented as \( \psi(x) \), where \( \psi(x) \in \mathbb{R}^d \), which we convert to a hash code \( h = \text{sgn}(\psi(x)) \), where \( \text{sgn}(\cdot) \)
Figure 1: The Domain Adaptive Hash (DAH) network that outputs hash codes for the source and the target. The network is trained with a batch of source and target data. The convolution layers $conv1$-$conv5$ and the fully connected layers $fc6$ and $fc7$ are fine-tuned from the VGG-F network. The MK-MMD loss trains the DAH to learn feature representations which align the source and the target. The hash-$fc8$ layer is trained to output vectors of $d$ dimensions. The supervised hash loss drives the DAH to estimate a unique hash value for each object category. The unsupervised entropy loss aligns the target hash values to their corresponding source categories. Best viewed in color.

is the sign function. Once the network has been trained, the probability of $x$ being assigned a label $y$ is given by $f(x) = p(y|h)$. We train the network using $D_s$ and $D_i$ and predict the target data labels $y^t$ using $f(.)$.

In order to address the issue of domain-shift, we need to align the feature representations of the target and the source. We do that by reducing the domain discrepancy between the source and target feature representations at multiple layers of the network. In the following subsections, we discuss the design of the domain adaptive hash (DAH) network in detail.

3.1. Reducing Domain Disparity

Deep learning methods have been very successful in domain adaptation with state-of-the-art algorithms [17, 31, 34, 46] in recent years. The feature representations transition from generic to task-specific as one goes up the layers of a deep CNN [49]. The convolution layers $conv1$ to $conv5$ have been shown to be generic and so, readily transferable, whereas the fully connected layers are more task-specific and need to be adapted before they can be transferred.

In the DAH algorithm, we attempt to minimize the MK-MMD loss to reduce the domain difference between the source and target feature representations for fully connected layers, $F = \{fc6, fc7, fc8\}$. Such a loss function has been used in previous research [31, 34]. The multi-layer MK-MMD loss is given by

$$M(u_s, u_t) = \sum_{l \in F} d^2_k(u^l_s, u^l_t),$$  \hspace{1cm} (1)

where, $u^l_s = \{u^{l,i}_s\}_{i=1}^{n_s}$ and $u^l_t = \{u^{l,i}_t\}_{i=1}^{n_t}$ are the set of output representations for the source and target data at layer $l$, where $u^{l,i}_s$ is the output representation of $x^*_i$ for the $l^{th}$ layer. The final layer outputs are denoted as $u^l_s$ and $u^l_t$.

The MK-MMD measure $d^2_k(\cdot)$ is the multi-kernel maximum mean discrepancy between the source and target representations, [24]. For a nonlinear mapping $\phi(\cdot)$ associated with a reproducing kernel Hilbert space $\mathcal{H}_k$ and kernel $k(\cdot)$, where $k(x, y) = \langle \phi(x), \phi(y) \rangle$, the MMD is defined as,

$$d^2_k(u^l_s, u^l_t) = \left\| \mathbb{E} [\phi(u^l_s)] - \mathbb{E} [\phi(u^l_t)] \right\|^2_{\mathcal{H}_k}. \hspace{1cm} (2)$$

The characteristic kernel $k(\cdot)$, is determined as a convex combination of $\kappa$ PSD kernels, $\{k_m\}_{m=1}^{\kappa}$, $\mathcal{K} := \{k : k = \sum_{m=1}^{\kappa} \beta_m k_m, \sum_{m=1}^{\kappa} \beta_m = 1, \beta_m \geq 0, \forall m\}$. We set $\beta_m = 1/\kappa$ according to [34] and it works well in practice.

3.2. Supervised Hashing for Source Data

The Hamming distance for a pair of hash values $h_i$ and $h_j$ has a unique relationship with the dot product $\langle h_i, h_j \rangle$, given by: $\text{dist}_{H}(h_i, h_j) = \frac{1}{2}(d - h_i^\top h_j)$, where $d$ is the hash length. The dot product $\langle h_i, h_j \rangle$ can be treated as a similarity measure for the hash codes. Larger the value of the dot product (high similarity), smaller is the distance $\text{dist}_H$. Let $s_{ij} \in \{0, 1\}$ be the similarity between $x_i$ and $x_j$. If $x_i$ and $x_j$ belong to the same category, $s_{ij} = 1$ and 0, otherwise. The probability of similarity between $x_i$ and $x_j$ given the corresponding hash values $h_i$ and $h_j$, can be expressed as a likelihood function, given by,

$$p(s_{ij}|h_i, h_j) = \begin{cases} 
\sigma(h_i^\top h_j), & s_{ij} = 1 \\
1 - \sigma(h_i^\top h_j), & s_{ij} = 0,
\end{cases} \hspace{1cm} (3)$$

where, $\sigma(x) = \frac{1}{1 + e^{-x}}$ is the sigmoid function. As the dot product $\langle h_i, h_j \rangle$ increases, the probability of $p(s_{ij} = 1|h_i, h_j)$ also increases, i.e., $x_i$ and $x_j$ belong to the same category. As the dot product decreases, the probability $p(s_{ij} = 1|h_i, h_j)$ also decreases, i.e., $x_i$ and $x_j$ belong to different categories. We construct the $\{n_x \times n_x\}$ similarity matrix $S = \{s_{ij}\}$, for the source data with the provided labels, where $s_{ij} = 1$ if $x_i$ and $x_j$ belong to the same category and 0, otherwise. Let $H = \{h_i\}_{i=1}^{n_x}$ be the set of source data hash values. If the elements of $H$ are assumed to be i.i.d., the negative log likelihood of the similarity matrix $S$ given $H$ can be written as,

$$\min_{H} \mathcal{L}(H) = -\log p(S|H) = -\sum_{s_{ij} \in S} \left( s_{ij} h_i^\top h_j - \log(1 + \exp(h_i^\top h_j)) \right). \hspace{1cm} (4)$$

By minimizing Equation (4), we can determine hash values $H$ for the source data which are consistent with the similarity matrix $S$. The hash loss has been used in previous research for supervised hashing [30, 50].
(4) is a discrete optimization problem that is challenging to solve. We introduce a relaxation on the discrete constraint $h_i \in \{-1, +1\}^d$ by instead solving for $u_i \in \mathbb{R}^d$, where $u_s = (u_{ij})_{j=1}^{C}$ is the output of the network and $u_t = \psi(x_t)$ (the superscript denoting the domain has been dropped for ease of representation). However, the continuous relaxation gives rise to (i) approximation error, when $(h_i, h_j)$ is substituted with $(u_i, u_j)$ and, (ii) quantization error, when the resulting real codes $u_i$ are binarized [50]. We account for the approximation error by having a tanh(.) as the final activation layer of the neural network, so that the components of $u_i$ are bounded between $-1$ and $+1$. In addition, we also introduce a quantization loss $\|u_i - \text{sgn}(u_i]\|_2^2$ along the lines of [22], where sgn(.) is the sign function. The continuous optimization problem for supervised hashing can now be outlined;

$$\min_{u_s} \mathcal{L}(u_s) = -\sum_{s \in \mathcal{S}} \left( s_{ij} u_i^T u_j - \log(1 + \exp(u_i^T u_j)) \right) + \sum_{i=1}^{n_s} \|u_i - \text{sgn}(u_i]\|_2^2.$$

$$\text{(5)}$$

### 3.3. Unsupervised Hashing for Target Data

In the absence of target data labels, we use the similarity measure $(u_i, u_j)$, to guide the network to learn discriminative hash values for the target data. An ideal target output $u_i^t$, needs to be similar to many of the source outputs from the $j^{th}$ category $\{(u_k^s)_{k=1}^{K}\}$. We assume without loss of generality, $K$ source data points for every category $j$ where, $j \in \{1, \ldots, C\}$ and $u_k^s$ is the $k^{th}$ source output from category $j$. In addition, $u_i^t$ must be dissimilar to most other source outputs $u_k^s$ belonging to a different category ($j \neq l$). Enforcing similarity with all the $K$ data points makes for a more robust target data category assignment. We outline a probability measure to capture this intuition. Let $p_{ij}$ be the probability that input target data point $x_i$ is assigned to category $j$ where,

$$p_{ij} = \frac{\sum_{k=1}^{K} \exp(u_i^T u_k^s)}{\sum_{t=1}^{C} \sum_{k=1}^{K} \exp(u_i^T u_k^s)}$$

$$\text{(6)}$$

The exp(.) has been introduced for ease of differentiability and the denominator ensures $\sum_j p_{ij} = 1$. When the target data point output is similar to one category only and dissimilar to all the other categories, the probability vector $p_i = [p_{i1}, \ldots, p_{iC}]^T$ tends to be a one-hot vector. A one-hot vector can be viewed as a low entropy realization of $p_i$. We can therefore envisage all the $p_i$ to be one-hot vectors (low entropy probability vectors), where the target data point outputs are similar to source data point outputs in one and only one category. To this end we introduce a loss to capture the entropy of the target probability vectors. The entropy loss for the network outputs is given by,

$$\mathcal{H}(u_s, u_t) = -\frac{1}{n_t} \sum_{i=1}^{n_s} \sum_{j=1}^{C} p_{ij} \log(p_{ij}).$$

$$\text{(7)}$$

Minimizing the entropy loss gives us probability vectors $p_i$ that tend to be one-hot vectors, i.e., the target data point outputs are similar to source data point outputs from any one category only. Enforcing similarity with $K$ source data points from a category, guarantees that the hash values are determined based on a common similarity between multiple source category data points and the target data point.

### 3.4. Domain Adaptive Hash Network

We propose a model for deep unsupervised domain adaptation based on hashing (DAH) that incorporates unsupervised domain adaptation between the source and the target (1), supervised hashing for the source (5) and unsupervised hashing for the target (7) in a deep convolutional neural network. The DAH network is trained to minimize

$$\min_{\mathcal{U}} \mathcal{J} = \mathcal{L}(u_s) + \gamma \mathcal{M}(u_s, u_t) + \eta \mathcal{H}(u_s, u_t),$$

where, $\mathcal{U} = \{u_s \cup u_t\}$ and $(\gamma, \eta)$ control the importance of domain adaptation (1) and target entropy loss (7) respectively. The hash values $H$ are obtained from the output of the network using $H = \text{sgn}(u)$. The loss terms (5) and (7) are determined in the final layer of the network with the network output $u$. The MK-MMD loss (1) is determined between layer outputs $\{u_i, u_i^t\}$ at each of the fully connected layers $\mathcal{F} = \{fc6, fc7, fc8\}$, where we adopt the linear time estimate for the unbiased MK-MMD as described in [24] and [31]. The DAH is trained using standard backpropagation. The detailed derivation of the derivative of (8) w.r.t $\mathcal{U}$ is provided in the supplementary material.

**Network Architecture:** Owing to the paucity of images in a domain adaptation setting, we circumvent the need to train a deep CNN with millions of images by adapting the pre-trained VGG-F [8] network to the DAH. The VGG-F has been trained on the ImageNet 2012 dataset and it consists of 5 convolution layers (conv1 - conv5) and 3 fully connected layers (fc6, fc7, fc8). We introduce the hashing layer hash-fc8 that outputs vectors in $\mathbb{R}^d$ in the place of fc8. To account for the hashing approximation, we introduced a tanh() layer. However, we encounter the issue of vanishing gradients [26] when using tanh() as it saturates with large inputs. We therefore preface the tanh() with a batch normalization layer which prevents the tanh() from saturating. In effect, hash-fc8 := [fc8 $\rightarrow$ batch-norm $\rightarrow$ tanh()]. The hash-fc8 provides greater stability when fine-tuning the learning rates than the deep hashing networks [30, 50]. Figure 1 illustrates the proposed DAH network.
refers to classification accuracies using the LIBLINEAR.

depicts a sampling of outlines some minimum number of images amongst all the categories, is the classification accuracy.

Table 1: Statistics for the Office-Home dataset. Min: # is the minimum number of images amongst all the categories, Min: Size and Max: Size are the minimum and maximum image sizes across all categories and Acc. is the classification accuracy.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Min: #</th>
<th>Min: Size</th>
<th>Max: Size</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art</td>
<td>15</td>
<td>117×85 pix.</td>
<td>4384×2686 pix.</td>
<td>44.99±1.85</td>
</tr>
<tr>
<td>Clipart</td>
<td>39</td>
<td>18×18 pix.</td>
<td>2360×2360 pix.</td>
<td>55.95±1.45</td>
</tr>
<tr>
<td>Product</td>
<td>38</td>
<td>75×63 pix.</td>
<td>2650×2650 pix.</td>
<td>66.41±1.18</td>
</tr>
<tr>
<td>Real-World</td>
<td>23</td>
<td>68×80 pix.</td>
<td>6500×4900 pix.</td>
<td>59.70±1.04</td>
</tr>
</tbody>
</table>

4. The Office-Home Dataset

Supervised deep learning models require a large volume of labeled training data. Unfortunately, existing datasets for vision-based domain adaptation are limited in their size and are not suitable for validating deep learning algorithms. The standard datasets for vision based domain adaptation are, facial expression datasets CKPlus [35] and MMI [40], digit datasets SVHN [36], USPS and MNIST[28], head pose recognition datasets PIE [33], object recognition datasets COIL[33], Office [42] and Office-Caltech [20]. These datasets were created before deep-learning became popular and are insufficient for training and evaluating deep learning based domain adaptation approaches. For instance, the object-recognition dataset Office has 4110 images across 31 categories and Office-Caltech has 2533 images across 10 categories.

We release the Office-Home dataset for domain adaptation based object recognition, that can be used to evaluate deep learning algorithms for domain adaptation. The Office-Home dataset consists of 4 domains, with each domain containing images from 65 categories of everyday objects and a total of around 15,500 images. The domains include, Art: artistic depictions of objects in the form of sketches, paintings, ornamentation, etc.; Clipart: collection of clipart images; Product: images of objects without a background, akin to the Amazon category in Office dataset; Real-World: images of objects captured with a regular camera.

Figure 2: Sample images from the Office-Home dataset. The dataset consists of images of everyday objects organized into 4 domains; Art: paintings, sketches and/or artistic depictions, Clipart: clipart images, Product: images without background and Real-World: regular images captured with a camera. The figure displays examples from 16 of the 65 categories.

Public domain images were downloaded from websites like www.deviantart.com and www.flickr.com to create the Art and Real-World domains. Clipart images were gathered from multiple clipart websites. The Product domain images were exclusively collected from www.amazon.com using web-crawlers. The collected images were manually filtered on the basis of quality, size and content. The dataset has an average of around 70 images per category and a maximum of 99 images in a category. The primary challenge in creating this dataset was acquiring sufficient number of public domain images across all the 4 domains. Figure 2 depicts a sampling of 16 categories from the Office-Home dataset and Table 1 outlines some meta data for the dataset. The Acc. column in the Table 1 refers to classification accuracies using the LIBLINEAR SVM [15] classifier (5-fold cross validation) with deep features extracted using the VGG-F network. The dataset is publicly available for research.

5. Experiments

In this section we conduct extensive experiments to evaluate the DAH algorithm. Since we propose a domain adaptation technique based on hashing, we evaluate objection recognition accuracies for unsupervised domain adaptation and also study the discriminatory capability of the learned hash codes for unsupervised domain adaptive hashing. The implementation details are available at [https://github.com/hemanthdv/da-hash](https://github.com/hemanthdv/da-hash)

5.1. Datasets

Office [42]: This is currently the most popular benchmark dataset for object recognition in the domain adaptation computer vision community. The dataset consists of images of everyday objects in an office environment. It has 3 domains; Amazon (A), Dslr (D) and Webcam (W). The dataset has

1https://hemanthdv.github.io/officehome-dataset/
around 4,100 images with a majority of the images (2816 images) in the Amazon domain. We adopt the common evaluation protocol of different pairs of transfer tasks for this dataset [31, 34]. We consider 6 transfer tasks for all combinations of source and target pairs for the 3 domains.

Office-Home: We introduce this new dataset and evaluate it in a similar manner to the Office dataset. We consider 12 transfer tasks for the Art (Ar), Clipart (Cl), Product (Pr) and Real-World (Rw) domains for all combinations of source and target for the 4 domains. Considering all the different pairs of transfer enables us to evaluate the inherent bias between the domains in a comprehensive manner [45].

5.2. Implementation Details

We implement the DAH using the MatConvnet framework [47]. Since we train a pre-trained VGG-F, we fine-tune the weights of conv1-conv5, fc6 and fc7. We set their learning rates to 1/10^6 the learning rate of hash-fc8. We vary the learning rate between 10^-4 to 10^-5 over 300 epochs with a momentum 0.9 and weight decay 5 x 10^-4. We set K = 5 (number of samples from a category). Since we have 31 categories in the Office dataset, we get a source batch size of 31 x 5 = 155. For the target batch, we randomly select 155 samples. The total batch size turns out to be 310. For the Office-Home dataset, with K = 5 and 65 categories, we get a batch size of 650. We set d = 64 (hash code length) for all our experiments. Since there is imbalance in the number of like and unlike pairs in S, we set the values in similarity matrix S_{i,j} ∈ {0, 10}. Increasing the similarity weight of like-pairs improves the performance of DAH. For the entropy loss, we set η = 1. For the MK-MMD loss, we follow the heuristics mentioned in [24], to determine the parameters. We estimate γ, by validating a binary domain classifier to distinguish between source and target data points and select γ which gives largest error on a validation set. For MMD, we use a Gaussian kernel with a bandwidth σ given by the median of the pairwise distances in the training data. To incorporate the multi-kernel, we vary the bandwidth σ_m ∈ [2^{-8}σ, 2^{8}σ] with a multiplicative factor of 2. We define the target classifier f(x_i) = p(y|h_i) in terms of 6. The target data point is assigned to the class with the largest probability, with ŷ_i = max_j(p_i_j) using the hash codes for the source and the target.

5.3. Unsupervised Domain Adaptation

In this section, we study the performance of the DAH for unsupervised domain adaptation, where labeled data is available only in the source domain and no labeled data is available in the target domain. We compare the DAH with state-of-the-art domain adaptation methods: (i) Geodesic Flow Kernel (GFK) [20], (ii) Transfer Component Analysis (TCA) [38], (iii) Correlation Alignment (CORAL) [44] and (iv) Joint Distribution Adaptation (JDA) [33]. We also compare the DAH with state-of-the-art deep learning methods for domain adaptation: (v) Deep Adaptation Network (DAN) [31] and (vi) Domain Adversarial Neural Network (DANN) [17]. For all of the shallow learning methods, we extract and use deep features from the fc7 layer of the VGG-F network that was pre-trained on the ImageNet 2012 dataset. We also evaluate the effect of the entropy loss on hashing for the DAH. The DAH-e is the DAH algorithm where γ is set to zero, which implies that the target hash values are not driven to align with the source categories. We follow the standard protocol for unsupervised domain adaptation, where all the labeled source data and all the unlabeled target data is used for training.

Results and Discussion: The results are reported for the target classification in each of the transfer tasks in Tables 2 and 3, where accuracies denote the percentage of correctly classified target data samples. We present results with hash length d = 64 bits. The DAH algorithm consistently outperforms the baselines across all the domains for the Office-Home dataset. However, DANN marginally surpasses DAH for the Office dataset, prompting us to reason that domain adversarial training is more effective than DAH when the categories are fewer in number. Since domain alignment is category agnostic, it is possible that the aligned domains are not classification friendly in the presence of large number of categories. When the number of categories is large, as in Office-Home, DAH does best at extracting transferable features to achieve higher accuracies. We also note that DAH delivers better performance than DAH-e; thus, minimizing the entropy on the target data through 7 aids in improved alignment of the source and target samples, which boosts the accuracy.

Feature Analysis: We also study the feature representations of the penultimate layer (fc7) outputs using t-SNE embeddings as in [12]. Figure 3a depicts the A-distance between domain pairs using Deep (VGG-F), DAN and DAH features. Ben-David et al. [2] defined A-distance as the distance between two domains that can be viewed as the discrepancy between two domains. Although it is difficult to estimate its exact value, an approximate distance measure is given by 2(1 − 2ε), where ε is the generalization error for a binary classifier trained to distinguish between

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GFK</td>
<td>48.59</td>
<td>52.08</td>
<td>41.83</td>
<td>89.18</td>
<td>49.04</td>
<td>93.17</td>
<td>62.35</td>
</tr>
<tr>
<td>TCA</td>
<td>51.00</td>
<td>49.43</td>
<td>48.12</td>
<td>93.08</td>
<td>48.83</td>
<td>96.79</td>
<td>64.54</td>
</tr>
<tr>
<td>CORAL</td>
<td>54.42</td>
<td>51.70</td>
<td>48.26</td>
<td>95.97</td>
<td>47.27</td>
<td>98.59</td>
<td>66.04</td>
</tr>
<tr>
<td>JDA</td>
<td>59.24</td>
<td>58.62</td>
<td>51.35</td>
<td>96.86</td>
<td>52.34</td>
<td>97.79</td>
<td>69.37</td>
</tr>
<tr>
<td>DAN</td>
<td>67.04</td>
<td>67.80</td>
<td>50.36</td>
<td>95.85</td>
<td>52.33</td>
<td>99.40</td>
<td>72.13</td>
</tr>
<tr>
<td>DANN</td>
<td>72.89</td>
<td>72.70</td>
<td>56.25</td>
<td>96.48</td>
<td>53.20</td>
<td>99.40</td>
<td>75.15</td>
</tr>
<tr>
<td>DAH-e</td>
<td>66.27</td>
<td>66.16</td>
<td>55.97</td>
<td>94.59</td>
<td>53.91</td>
<td>96.99</td>
<td>72.31</td>
</tr>
<tr>
<td>DAH</td>
<td>66.47</td>
<td>68.30</td>
<td>55.54</td>
<td>96.10</td>
<td>53.02</td>
<td>98.80</td>
<td>73.04</td>
</tr>
</tbody>
</table>
the two domains. We used a LIBLINEAR SVM [15] classifier with 5-fold cross-validation to estimate $\epsilon$. Figure 3a indicates that the DAH features have the least discrepancy between the source and target compared to DAN and Deep features. This is also confirmed with the t-SNE embeddings in Figures 3b-3d. The Deep features show very little overlap between the domains and the categories depict minimal clustering. Domain overlap and clustering improves as we move to DAN and DAH features, with DAH providing the best visualizations. This corroborates the efficacy of the DAH algorithm to exploit the feature learning capabilities of deep neural networks to learn representative hash codes to address domain adaptation.

### 5.4. Unsupervised Domain Adaptive Hashing

In this section, we study the performance of our algorithm to generate compact and efficient hash codes from the data for classifying unseen test instances, when no labels are available. This problem has been addressed in the literature, with promising empirical results [7, 11, 21]. However, in a real-world setting, labels may be available from a different, but related (source) domain; a strategy to utilize the labeled data from the source domain to learn representative hash codes for the target domain is therefore of immense practical importance. Our work is the first to identify and address this problem. We consider the following scenarios to address this real-world challenge: (i) No labels are available for a given dataset and the hash codes need to be learned in a completely unsupervised manner. We evaluate against baseline unsupervised hashing methods (ITQ) [22] and (KMeans) [25] and also state-of-the-art methods for unsupervised hashing (BA) [7] and (BDNN) [11]. (ii) Labeled data is available from a different, but related source domain. A hashing model is trained on the labeled source data and is used to learn hash codes for the target data. We refer to this method as NoDA, as no domain adaptation is performed. We used the deep pairwise-supervised hashing (DPSH) algorithm [30] to train a deep network with the source data and applied the network to generate hash codes for the target data. (iii) Labeled data is available from a different, but related source domain and we use our DAH formulation to learn hash codes for the target domain by reducing domain disparity. (iv) Labeled data is available in the target domain. This method falls under supervised hashing (SuH) (as it uses labeled data in the target domain to learn hash codes in the same domain) and denotes the upper bound on the performance. It is included to compare the performance of unsupervised hashing algorithms relative to the supervised algorithm. We used the DPHS algorithm [30] to train a deep network on the target data and used it to generate hash codes on a validation subset.

### Results and Discussion

We applied the precision-recall curves and the mean average precision (mAP) measures to evaluate the efficacy of the hashing methods, similar to previous research [7, 11, 21]. The results are depicted in Figure 5.
Figure 3: Feature analysis of fc7 layer. (a) A-distances for Deep, DAN and DAH, (b), (c) and (d) t-SNE embeddings for 10 categories from Art (•) and Clipart (+) domains. Best viewed in color.

Figure 4: Precision-Recall curves @64 bits for the Office-Home dataset. Comparison of hashing without domain adaptation (NoDA), shallow unsupervised hashing (ITQ, KMeans), state-of-the-art deep unsupervised hashing (BA, BDNN), unsupervised domain adaptive hashing (DAH) and supervised hashing (SuH). Best viewed in color.

6. Conclusions

In this paper, we have proposed a novel domain adaptive hashing (DAH) framework which exploits the feature learning capabilities of deep neural networks to learn efficient hash codes for unsupervised domain adaptation. The DAH framework solves two important practical problems: category assignment with weak supervision or insufficient labels (through domain adaptation) and the estimation of hash codes in an unsupervised setting (hash codes for target data). Thus, two practical challenges are addressed through a single integrated framework. This research is the first of its kind to integrate hash code learning with unsupervised domain adaptation. We also introduced a new dataset, Office-Home, which can be used to further research in domain adaptation.

Acknowledgements: This material is based upon work supported by the National Science Foundation (NSF) under Grant No:1116360. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.
References


