Deep Domain Adaptation Hashing with Adversarial Learning

Fuchen Long[†], Ting Yao[‡], Qi Dai[‡], Xinmei Tian[†], Jiebo Luo[§] and Tao Mei[‡]

[†]University of Science and Technology of China, Hefei, China
[‡]Microsoft Research, Beijing, China
[§]University of Rochester, Rochester, NY USA
{longfc,tingyao}.ustc@gmail.com;qid@microsoft.com;xinmei@ustc.edu.cn;
jluo@cs.rochester.edu;tmei@live.com

ABSTRACT

The recent advances in deep neural networks have demonstrated high capability in a wide variety of scenarios. Nevertheless, fine-tuning deep models in a new domain still requires a significant amount of labeled data despite expensive labeling efforts. A valid question is how to leverage the source knowledge plus unlabeled or only sparsely labeled target data for learning a new model in target domain. The core problem is to bring the source and target distributions closer in the feature space. In the paper, we facilitate this issue in an adversarial learning framework, in which a domain discriminator is devised to handle domain shift. Particularly, we explore the learning in the context of hashing problem, which has been studied extensively due to its great efficiency in gigantic data. Specifically, a novel Deep Domain Adaptation Hashing with Adversarial learning (DeDAHA) architecture is presented, which mainly consists of three components: a deep convolutional neural networks (CNN) for learning basic image/frame representation followed by an adversary stream on one hand to optimize the domain discriminator, and on the other, to interact with each domain-specific hashing stream for encoding image representation to hash codes. The whole architecture is trained end-to-end by jointly optimizing two types of losses, i.e., triplet ranking loss to preserve the relative similarity ordering in the input triplets and adversarial loss to maximally fool the domain discriminator with the learnt source and target feature distributions. Extensive experiments are conducted on three domain transfer tasks, including cross-domain digits retrieval, image to image and image to video transfers, on several benchmarks. Our DeDAHA framework achieves superior results when compared to the state-of-the-art techniques.

CCS CONCEPTS

• **Information systems → Similarity measures**; **Learning to rank**; *Top-k retrieval in databases*;

KEYWORDS

Hashing; Similarity Learning; Domain Adaptation; Adversarial Learning; CNN

ACM Reference Format:
Fuchen Long[†], Ting Yao[‡], Qi Dai[‡], Xinmei Tian[†], Jiebo Luo[§] and Tao Mei[‡]. 2018. Deep Domain Adaptation Hashing with Adversarial Learning. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3209999

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3209999

1 INTRODUCTION

Convolutional Neural Networks (CNN) have convincingly been regarded as a powerful class of models for learning visual representation, which are generically workable across different tasks and visual domains [38]. To date in the literature, there are various datasets (e.g., ImageNet [22]) that include expert labeled training data available for developing CNN from AlexNet [10] to more recent ResNet [4] and its variants [35]. Nevertheless, given a new dataset, the typical solution is still to perform "intensive manual labeling" and then fine-tune the pre-learnt CNN with the examples collected for the new dataset. Many previous experiences have also shown that by doing so can generate satisfactory performance, though the learnt CNN may not be that transferable when being applied in another dataset due to a phenomenon known as "domain shift." As a result, there have been several domain adaptation techniques being proposed for alleviating this challenge by learning deep transformations to map both domains into a common representation distribution. A general practice to optimize the mappings is to minimize the measure of domain shift such as correlation distances [25] or maximum mean discrepancy [28].

The inspiration of recent attempts on domain adaptation are from the advances of adversarial learning, which is to model domain distribution via an adversarial objective with respect to a domain discriminator. The spirit behind is from generative adversarial learning [6], that trains two models, i.e., a generative model and a discriminative model, by pitting them against each other. This process corresponds to a minimax two-player game, in which a generative model is to capture the data distribution and a discriminative model aims to estimate the probability that a sample is from the real training data rather than the generative model. The two models are trained simultaneously and the learning of the generative model is to fool the discriminative model into making mistakes. In the context of domain adaptation, this adversarial principle is then equivalent to guiding the representation learning in both domains,

making the difference between source and target representation distributions indistinguishable through the domain discriminator. We follow this elegant recipe and capitalize on adversarial learning for domain adaptation. Furthermore, with the tremendous increase of multimedia data on the Web, the need to search for millions of visual examples in a high-dimensional feature space motivates the surge of research in large-scale visual search. In between, Hashing techniques [32] are probably the "hottest" topic that produces a compact binary representation for each example and measures visual similarity by computing the Hamming distance between each two hash codes, leading to a very efficient search. Therefore, we are particularly investigating the problem of domain adaptation on binary representation learning in this work.

By consolidating the idea of adversarial learning into domain adaptation for enhancing binary representation generation, we present a novel Deep Domain Adaptation Hashing with Adversarial learning (DeDAHA) architecture, as shown in Figure 1. Specifically, we form a set of image (video frame) triplets as the input and each tuple contains one query image, one semantically similar image and one dissimilar image. A CNN is employed to produce visual representations of each image (video frame), followed by an adversary stream to differentiate the representation distributions of source and target domains. Meanwhile, a hash stream is devised to encode hash codes in each domain and benefited from interacting with adversary stream to aggregate domain-invariant and domain-specific knowledge for enhancing hash learning. More importantly, we untie weight sharing of CNN in each domain to explicitly model the domain shift rather than enforcing the target close to source as much as possible. The whole architecture of DeDAHA is trained end-to-end by optimizing two kinds of losses, i.e., triplet ranking loss to characterize relative similarity ordering in the triplets from each domain and adversarial loss to make the domain discriminator unable to differentiate between the source and target distributions. As such, our DeDAHA endows the target model with more power of exploring source distribution and thus ensures good generalization ability. Moreover, the generated hash codes could better reflect semantic relations between images (video frames). It is also worth noting that our framework could be easily extended to unsupervised scenario, i.e., there is no labeled data in target domain, which will be explored and elaborated in Section 3.5.

The main contribution of this work is the proposal of DeDAHA framework to learn domain adaptive binary representation in a domain adversarial manner. The solution also leads to the elegant views of how adversarial training should be leveraged for domain adaptation when there are only a few and even no labeled data in target domain and how the interactions across streams could be taken into account to boost hash learning, which are problems not yet fully understood. The remaining sections are organized as follows. Section 2 describes the related works on both hashing and domain adaptation. Section 3 presents our proposed DeDAHA model and the extensions on unsupervised setting. Section 4 provides empirical evaluations, followed by the conclusions in Section 5.

2 RELATED WORK

We summarize recent works related to our proposed approach into two categories: hash learning and deep domain adaptation. The former emphasizes learning to encode data points into compact binary codes for efficient similarity search, while the latter focuses on the domain shift problem in deep learning framework.

Hashing for Search. The research in learning to hash proceeds along two dimensions: handcrafted feature based hashing and deep learning based hashing. The former direction can be briefly grouped into three categories: unsupervised hashing, supervised hashing, and semi-supervised hashing. Unsupervised hashing tries to capture the underlying structure information among unlabeled data [3]. For example, Spectral Hashing [33] learns compact binary codes by preserving the similarity between samples. In contrast, we refer to the problem as supervised hashing when all label information is available for use. The representative in this category is Kernel-based Supervised Hashing (KSH) [14], which applies kernelization to formulate hash functions and minimizes the loss function over hash codes. In addition, to utilize both the labeled and unlabeled training data, semi-supervised hashing methods [17, 31] are studied.

Benefiting from recent advances in image representation using deep CNN, a few deep architecture based hashing methods have been proposed for image retrieval. Semantic Hashing [23] is the one of the early works to exploit deep learning techniques for hashing. It applies the stacked Restricted Boltzmann Machine (RBM) to learn hash codes for visual search. Xia *et al.* propose a two-stage hash learning method called Convolutional Neural Networks Hashing (CNNH) [34], where the feature learning stage and hash codes generation stage are separated. Furthermore, Network In Network Hashing (NINH) exploits triplet ranking loss to model the similarity relationships between images in [11]. Later in [36], Yao *et al.* devise a two-stream framework, which combines hash coding and classification for preserving not only relative similarity between images but also semantic structures on images. More recently, [18] enlarges training data with synthetic images generated by generative adversarial networks (GANs) for better hash learning.

In short, while all the aforementioned deep hashing methods achieve impressive results on specific datasets, it is difficult to directly apply the trained models to other data due to the domain shift. To solve this problem, our work mainly focuses on how to adapt hashing model trained in source domain with rich labeled data to a target domain with scarce or even no labeled data.

Deep Domain Adaptation. Domain adaptation [13] aims to transfer knowledge from a labeled source domain to a target domain where labeled data is sparse or completely unavailable. Early approaches of domain adaptation focus on building feature representations that are invariant across domains. This was accomplished either by feature embedding [37] or selection mechanisms [5]. Though deep neural networks have successfully demonstrated high capability in learning visual representations [4, 10], the powerful features still largely rely on specific training dataset and may lead to poor performance when applying to others [38]. To utilize the effective pre-trained models, many deep domain adaptation methods have been proposed and proceeded along a few dimensions. One line of methods utilize Maximum Mean Discrepancy (MMD) as the metric to measure the shift between domains. Deep Domain Confusion (DDC) [28] applies MMD as well as the regular classification loss on the source to learn representations that are both discriminative and domain invariant. Deep Adaptation Network (DAN) [15] extends this idea by embedding all task specific

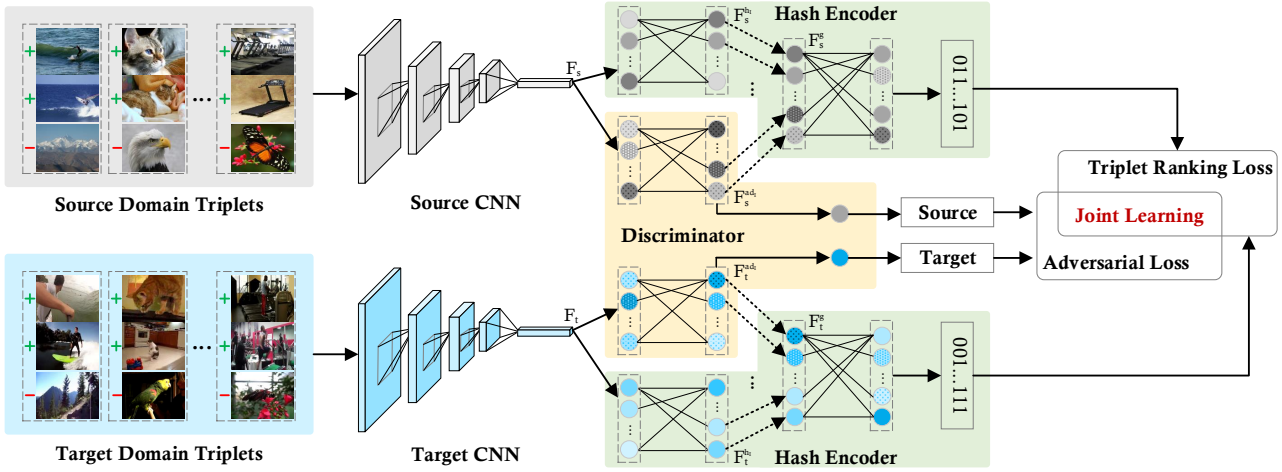


Figure 1: The framework of Deep Domain Adaptation Hashing with Adversarial learning (DeDAHA). The triplets from source and target domains are fed into two unshared weights CNNs to learn visual representations in each domain, followed by two streams, i.e., adversary and hash streams. Adversary stream makes the distributions of source and target domains indistinguishable through a domain discriminator. Hash stream encodes hash codes in each domain and benefits from interacting with adversary stream to aggregate domain-invariant and domain-specific knowledge for enhancing hash learning (dashed arrow). Better viewed in original color pdf.

layers in a reproducing kernel Hilbert space. In Deep Correlation Alignment (CORAL) [25], MMD is exploited to match the mean and covariance of two distributions.

Another direction aims at learning domain indistinguishable representations to bridge domain gap. Tzeng *et al.* [26] utilize a binary domain classifier and devise the domain confusion loss to encourage the predicted domain labels to be uniformly distributed. Inspired by adversarial learning in [6], the gradient reversal algorithm proposed in [2] treats domain invariance as a binary classification problem. It minimizes the loss of class classifier, and maximizes the loss of domain classifier. With standard adversarial training, Adversarial Discriminative Domain Adaptation (ADDA) [27] unties weight sharing of CNN models in source and target domains. It allows to learn domain specific feature embedding and improves the classification accuracy in target domain. In the context of domain adaptation for semantic segmentation [39], the adversarial principle is then equivalent to guiding the representation learning across domains, making the difference between source and target representation distributions indistinguishable through the domain discriminator on each image region. Moreover, Venkateswara *et al.* [30] address the domain adaptation hashing problem by a deep hashing framework with Multi-Kernel Maximum Mean Discrepancy (MK-MMD) loss. Our work in this paper contributes by studying not only domain adaptation through adversarial learning, but also how to aggregate domain-specific and domain-invariant knowledge through stream interaction in a deep architecture.

3 DEEP DOMAIN ADAPTATION HASHING WITH ADVERSARIAL LEARNING

In this section we present the proposed Deep Domain Adaptation Hashing with Adversarial learning framework (DeDAHA). Figure 1 illustrates the overview of our framework for domain adaptation hashing. It consists of three key components: the unshared weights CNN, an adversary stream with a domain discriminator, and one hash learning stream in each domain. CNN models are utilized to extract basic feature representations. In adversary stream, a domain discriminator D is designed to classify which domain the data comes from. Then the adversarial loss is exploited to maximally fool the discriminator with the learned source and target feature distributions. In hash stream, the intermediate domain-specific representations are augmented by concatenating representations from adversary stream, which incorporate the domain-invariant knowledge learned through discriminator. The triplet ranking loss is further utilized to preserve semantic similarity. Particularly, several training strategies are devised to obtain good domain-shift models. First, unlike traditional domain adaptation which aims to learn domain-invariant representations with unique CNN model, we untie weight sharing of CNN in each domain to learn more discriminative features. Second, we attempt to seek a common feature distribution for both source and target domains to adapt, instead of enforcing the target close to source as much as possible. In addition, we also extend our DeDAHA framework to fit the extreme case when there is no labeled data available in target domain.

3.1 Notation

Given n samples $X_t = \{x_t^i | i = 1, ..., n\}$ in target domain with labels $L_t = \{l_t^i | i = 1, ..., n\}$, and m samples $X_s = \{x_s^j | j = 1, ..., m\}$ in source domain with labels $L_s = \{l_s^j | j = 1, ..., m\}$, $n \ll m$, the goal of domain adaptation hashing is to learn a target hash mapping $\mathcal{H}_t : x_t \rightarrow \{0, 1\}^K$ adapted from source hash mapping model $\mathcal{H}_s$, such that a target domain sample x_t could be well encoded into K-bit binary code $\mathcal{H}_t(x_t)$. Our approach builds upon two streams, i.e., the adversary stream and hash stream, both of which take the outputs of CNN model as inputs.

3.2 Adversary Stream

The adversarial learning [6] has provided a series of elegant learning schemes for many tasks, such as image and text generation.

Particularly, in the problem of domain adaptation, this adversarial principle is equivalent to guiding the learning of both source and target feature representations F_s and F_t, making their feature distributions indistinguishable through a domain discriminator D. The whole framework corresponds to a two-player minimax game. The CNN feature model F, which can be treated as the generator in GANs, is to learn indistinguishable representations across source and target domains. The discriminator D, which consists of a series of fully-connected (FC) layers, tries best to differentiate between them. [26] is one recent framework to model the domain adaptation in an adversarial manner. It exploits the domain confusion loss to minimize the discrepancy between source and target distributions, and simultaneously trains a domain classifier to identify them. In particular, it shares the parameters of source and target CNN models to learn domain-invariant representations. However, when the gap between source and target domains is large, it is unreasonable to force the target distribution $F_t(X_t)$ identical to the source distribution $F_s(X_s)$, making such framework sub-optimal in many practical scenarios. Consequently, we untie weight sharing of CNN in each domain, and formulate adversarial loss as

$$\begin{aligned}\min_{D} \mathcal{L}_{\mathrm{ad}_D}(X_s, X_t, F_s, F_t) &= -E_{x_t \sim X_t}[log(D(F_t(x_t)))] \\ &\quad - E_{x_s \sim X_s}[log(1 - D(F_s(x_s)))], \\ \min_{F_s, F_t} \mathcal{L}_{\mathrm{ad}_F}(X_s, X_t, D) &= -E_{x_s \sim X_s}[log(D(F_s(x_s)))] \\ &\quad - E_{x_t \sim X_t}[log(1 - D(F_t(x_t)))],\end{aligned} \tag{1}$$

where $\mathcal{L}_{\mathrm{ad}_D}$ and $\mathcal{L}_{\mathrm{ad}_F}$ are exploited to optimize the discriminator and the feature model, respectively. Such mechanism clearly models domain shift instead of enforcing representations with identical model, and is thus helpful to learn the domain-invariant knowledge. In addition, the source feature model F_s is not fixed during training. It tries to seek a common feature space that is appropriate for both domains, instead of enforcing target towards source. To this end, optimized with the adversarial objective in Eq.(1), a more effective domain shift model could be learned.

The adversary stream explicitly utilizes the discriminator for domain adaptation learning. In addition, the learned intermediate domain-invariant representations are further exploited in the hash stream, which will be described in the following section.

3.3 Hash Stream

The hash stream is devised to map the feature representation to binary codes. It consists of several FC layers, denoted as hash encoder. In the traditional hash learning, the binary encoding of each sample is always treated independently in point-wise hashing methods [21], regardless of the similarity relationships between samples. Furthermore, the relative similarity relations like "for query x, it should be more similar to sample x^+ than to x^-" are reflected in the class labels in view that x and x^+ belong to the same class while x^- comes from another category. To fully take advantage of such relationships, our hash stream is learned in a triplet-wise manner for both source and target domains, which aims to preserve the relative similarity of the input triplets. Given a triplet (x, x^+, x^-) where x^+ is more similar to the query exmaple x than x^-, our goal is to preserve such triplet ranking information in the Hamming space. Therefore, the hash mapping $\mathcal{H}(\cdot)$ is expected to satisfy that the Hamming distance between $\mathcal{H}(x)$ and $\mathcal{H}(x^+)$ is smaller than the distance between $\mathcal{H}(x)$ and $\mathcal{H}(x^-)$. Accordingly, the triplet ranking loss is defined as the following triplet-based hinge loss

$$\begin{aligned}&\mathcal{L}_{triplet}(\mathcal{H}(x), \mathcal{H}(x^+), \mathcal{H}(x^-)) \\ &= \max(0, 1 - \left\|\mathcal{H}(x) - \mathcal{H}(x^-)\right\|_H + \left\|\mathcal{H}(x) - \mathcal{H}(x^+)\right\|_H), \\ &\quad s.t. \quad \mathcal{H}(x), \mathcal{H}(x^+), \mathcal{H}(x^-) \in \{0, 1\}^K,\end{aligned} \tag{2}$$

where $\|\cdot\|_H$ is the Hamming distance. Note that the problem in Eq. (2) is in general NP-hard because of the discrete constraint $\mathcal{H}(x) \in \{0, 1\}^K$. Hence, a common continuous relaxation that changes integer constraint to the range constraint is adopted, which replaces $\mathcal{H}(x)$ with $\hat{\mathcal{H}}(x) \in [0, 1]^K$. The Hamming distance is also replaced by the squared Euclidean distance. Finally, the hashing loss function is formulated as

$$\begin{aligned}&\hat{\mathcal{L}}_{triplet}(\hat{\mathcal{H}}(x), \hat{\mathcal{H}}(x^+), \hat{\mathcal{H}}(x^-)) \\ &= \max(0, 1 - \left\|\hat{\mathcal{H}}(x) - \hat{\mathcal{H}}(x^-)\right\|_2^2 + \left\|\hat{\mathcal{H}}(x) - \hat{\mathcal{H}}(x^+)\right\|_2^2), \\ &\quad s.t. \quad \hat{\mathcal{H}}(x), \hat{\mathcal{H}}(x^+), \hat{\mathcal{H}}(x^-) \in [0, 1]^K.\end{aligned} \tag{3}$$

This loss is utilized for hashing in both source and target domains.

In the problem of domain adaptation hashing discussed in this work, we aim to transfer the knowledge well learned in source domain to target domain. Thus, the above approximate hash codes $\hat{\mathcal{H}}(x)$ are expected to involve knowledge from both domains. We propose to aggregate the intermediate representations, $\{F^{h_I}, F^{ad_I}\} \in \mathbb{R}^d$, in hash and adversary streams to facilitate the hash learning, which implicitly encodes the domain-invariant knowledge from adversary stream. Formally, the aggregated features F^g are given by

$$F^g = G(F^{h_I}, F^{ad_I}), \tag{4}$$

where G denotes the aggregation operation. In our approach, we adopt the concatenation aggregation function. We also explore other aggregation schemes, e.g., element-wise sum, in section 4.6 for comparison, to validate the effectiveness of concatenation. Consequently, $\hat{\mathcal{H}}(x)$ is generated from the concatenation aggregation feature $F^g \in \mathbb{R}^{2d}$ with a linear mapping:

$$\hat{\mathcal{H}}(x) = \mathrm{sigmoid}(WF^g + b), \tag{5}$$

where $W \in \mathbb{R}^{K \times 2d}$ and $b \in \mathbb{R}^K$ represents the corresponding weight and bias, respectively. Finally, the hash code $\mathcal{H}(x)$ is obtained by quantizing the $\hat{\mathcal{H}}(x)$ to $\{0, 1\}^K$.

3.4 Optimization

Our proposed domain adaptation hashing method DeDAHA is trained by jointly optimizing the above adversary and hash streams. The overall training objective integrates the adversarial loss in Eq. (1) and triplet ranking loss in Eq. (3). In adversary stream, the objective function $\mathcal{L}_{\mathrm{ad}}$ consists of the following two parts

$$\min_{D} \mathcal{L}_{\mathrm{ad}_D}(X_s, X_t, F_s, F_t) \text{ and } \min_{F_s, F_t} \mathcal{L}_{\mathrm{ad}_F}(X_s, X_t, D). \tag{6}$$

In hash stream, triplet ranking loss is utilized in both source and target domains. The loss function $\mathcal{L}_h$ is formulated as

$$\min \mathcal{L}_h = \sum_{\mathcal{T}_s} \hat{\mathcal{L}}_{triplet}(x_s, x_s^+, x_s^-) + \sum_{\mathcal{T}_t} \hat{\mathcal{L}}_{triplet}(x_t, x_t^+, x_t^-), \tag{7}$$

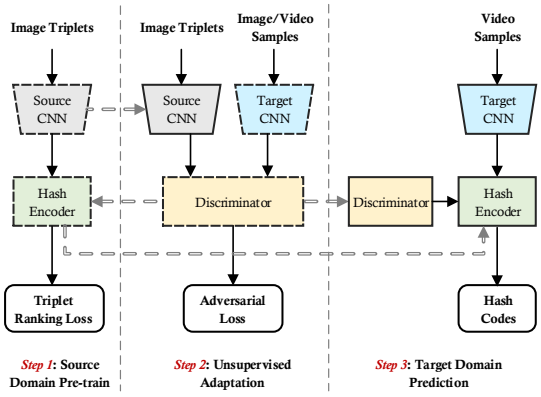


Figure 2: Extended DeDAHA (DeDAHA$^-$) framework. It includes three steps. Source domain pre-train step learns pre-trained model, which is utilized for target model initialization. Then we jointly optimize the domain discriminator and fine-tune the source hash encoder, with interaction between them. During target prediction, the hash codes are generated by mapping aggregated features of two streams to binary codes. Dashed line boundary implies that it will be optimized in this step and solid line boundary implies fixed.

where $\mathcal{T}_s$ and $\mathcal{T}_t$ is the triplet set in source and target domain, respectively. The whole framework is jointly optimized in an end-to-end manner. The final loss function $\mathcal{L}$ is then calculated by

$$\mathcal{L} = \alpha \mathcal{L}_{\text{ad}} + \mathcal{L}_h, \tag{8}$$

where the parameter α is to balance the adversary and hash stream.

By combining adversary and hash streams, the source and target feature distributions, $F_s(X_s)$ and $F_t(X_t)$, will be closer and become indistinguishable by the discriminator D. Meanwhile, the relative visual similarity in each domain is preserved as well.

3.5 Extensions

When adapting existing model to a new domain, the extreme case is that the labeled data in the new domain is completely unavailable. In this section, we extend our framework to this case, i.e., only labeled data in source domain L_s is available. The extended version of our domain adaptation hashing with adversarial learning framework, named DeDAHA$^-$, is illustrated in Figure 2.

In the first step of training, we pre-train the hash stream in source domain by using the source labeled data $\mathcal{X}_s$ with triplet ranking loss. Since there is no adversary stream at this stage, only the weights corresponding to hash stream are valid in W, which are specifically denoted as $W_{s1} \in \mathbb{R}^{K \times d}$. In the second step, we initialize the target CNN with the learned source CNN in the first step, and simultaneously optimize the adversary stream by incorporating unlabeled target data and fine-tune the source hash encoder with the concatenated intermediate representations. The weights $W \in \mathbb{R}^{K \times 2d}$ here are initialized with the concatenation of W_{s1} and a zero matrix of $\mathbf{0} \in \mathbb{R}^{K \times d}$. It is worth noting that we fix the source CNN model during adversarial training and fine-tuning in this step, which helps avoid model degeneration problem caused by the unlabeled data in target domain. In the final step of hash codes prediction in target domain, the learnt target CNN model is utilized for basic feature extraction. By aggregating intermediate representations from adversary stream, the learnt hash encoder for source domain is adapted to generate hash codes for target domain.

Table 1: The statistics of FCVID-S and ImageNet-F. The value denotes the number of videos/images in each category.

ID	Category	FCVID-S	ImageNet-F	ID	Category	FCVID-S	ImageNet-F
1	amusement park	568	1101	19	golfing	331	361
2	badminton	707	801	20	gorilla	572	1378
3	baseball	293	1156	21	hamster	223	832
4	beach	268	1073	22	laptop	372	1058
5	bee	271	1346	23	mountain	201	436
6	billiard	370	732	24	panda	236	1590
7	bird	294	1373	25	rabbit	525	934
8	bowling	295	585	26	river	252	670
9	butterfly	332	1286	27	roller skating	304	1106
10	camel	807	977	28	reflex camera	253	755
11	cat	328	868	29	skiing	390	687
12	cow	455	1017	30	snake	712	1086
13	delicious food	159	1124	31	surfing	229	1125
14	dog	533	878	32	tennis	269	856
15	dolphin	230	542	33	treadmill	350	730
16	elephant	790	825	34	turtle	222	1211
17	forest	109	459	*	**Total**	**12699**	**31684**
18	giraffe	419	726				

Our extended domain adaptation hashing with adversarial learning thus follows the below optimization

$$\begin{aligned} &\min \sum_{\mathcal{T}_s} \hat{\mathcal{L}}_{triplet}(x_s, x_s^+, x_s^-), \\ &\min_D \mathcal{L}_{\text{ad}_D}(X_s, X_t, F_s, F_t) = -E_{x_t \sim X_t}[log(D(F_t(x_t)))] \\ &\qquad\qquad - E_{x_s \sim X_s}[log(1 - D(F_s(x_s)))], \\ &\min_{F_t} \mathcal{L}_{\text{ad}_F}(X_s, X_t, D) = -E_{x_t \sim X_t}[log(1 - D(F_t(x_t)))]. \end{aligned} \tag{9}$$

Note that the adversarial learning of DeDAHA$^-$ is slightly different from that of DeDAHA. The DeDAHA$^-$ optimizes the target distribution $F_t(X_t)$ until it is indistinguishable from the fixed source distribution $F_s(X_s)$, while DeDAHA optimizes both $F_t(X_t)$ and $F_s(X_s)$ towards a common feature space.

3.6 Retrieval in Target Domain

After the optimization of DeDAHA or DeDAHA$^-$, we employ target hashing model to generate K-bit hash codes. To obtain the binary codes h_t, a quantization operation $h_t = sign\ (\hat{\mathcal{H}}(x_t) - 0.5)$ is exploited. $sign\ (v)$ is the sign function on vector v, where $sign\ (v_i) = 1$ if $v_i > 0$, otherwise $sign\ (v_i) = 0$. Given a query in target domain, the retrieval list is produced by sorting the Hamming distances of hash codes between query and data points in search pool.

4 EXPERIMENTS

We conduct extensive evaluations of our proposal across five different domain shifts, including four image to image domain transfer and one image to video transfer. Domain adaptation across three Digits datasets, i.e., MNIST [12], USPS and SVHN [16] are comprehensively studied. Moreover, we also explore two real world image datasets: CIFAR-10 [9] and a subset of ImageNet [1] containing the categories which share common definition with those in CIFAR-10. For the image to video transfer, the subsets with common categories of FCVID [8] and ImageNet are selected for evaluation.

4.1 Datasets

The MNIST and USPS image datasets are both handwritten Digits datasets. The MNIST dataset consists of $70k$ images (28×28 pixels) and the USPS dataset contains $9.3k$ images (16×16 pixels). Unlike the two, the SVHN dataset is a real-world Digits dataset obtained from house numbers in Google street view images and contains over $600k$ images (32×32 pixels) in total.

The CIFAR-10 dataset consists of 60k real world tiny images (32 × 32 pixels) from 10 categories, with 6k images per category. We sample 1k images for each category in ImageNet and construct a subset, namely ImageNet-C, as the target domain of CIFAR-10.

The FCVID dataset is a large video dataset which contains 91, 223 videos from 239 categories. In between FCVID and ImageNet datasets, there are 34 categories which share common definitions and thus we build the subset of FCVID and ImageNet, named FCVID-S and ImageNet-F, respectively. Table 1 details the statistics of the two subsets and ImageNet-F → FCVID-S is used for image to video transfer.

4.2 Experimental Settings

Domain Adaptation between Digits Datasets. Following [27], we consider three transfer directions: MNIST → USPS, USPS → MNIST and SVHN → MNIST, for domain adaptation between Digits datasets. We sample 500 images per class for MNIST and SVHN, and 300 images per class for USPS as the labeled data in source domain. Moreover, different settings are adopted for DeDAHA and DeDAHA$^-$ when these datasets are regarded as target domain. For DeDAHA, only a very small number of images (i.e., 3/5/10/15/20 images per class) are sampled as the labeled data in target domain. While for DeDAHA$^-$, 500 images per class for MNIST and 300 images per class for USPS are sampled as unlabeled data in target domain. In the testing stage, we randomly select 1k images (100 images per class) as the test query set, and the rest images are utilized as candidates to be retrieved. In addition, the CNN architecture is a simple modified version of LeNet [12], which is also exploited in [27]. The adversary stream includes three Fully-Connected (FC) layers, with the structure of FC500-ReLU-FC500-ReLU-FC1. The outputs from the second FC layer are utilized for aggregation in hash stream, which implicitly enhances the hash function learning. The hash stream consists of two FC layers with 500 and K (bit number) neurons respectively, where the output features in the first FC layer are utilized for feature aggregation.

Image to Image Adaptation. The second experiment was conducted on real image to image adaptation, i.e., CIFAR-10 → ImageNet-C transfer. We sample 500 images per class for CIFAR-10 as the labeled data in source domain. Similarly, we also have different settings for DeDAHA and DeDAHA$^-$ when performing domain adaptation. For DeDAHA, the setting is the same with above Digits adaptation, i.e., 3/5/10/15/20 images per class are sampled in ImageNet-C as the labeled data. While for DeDAHA$^-$, all the images in ImageNet-C are exploited as the unlabeled data in target domain. The setting of retrieval in testing stage is the same with adaptation between Digits datasets. We utilize the 19-layer VGGNet [24] as our basic CNN structure. The architectures of adversary and hash streams are the same with those in digits adaptation, as well as the feature aggregation scheme.

Image to Video adaptation. The third experiment was conducted on the most challenging image to video transfer. We evaluate this scenario on the transfer from ImageNet-F to FCVID-S. We sample about 500 images per class from ImageNet-F as the labeled data in source domain. The domain adaptation settings of DeDAHA and DeDAHA$^-$ are the same with image to image adaptation. In the testing stage, 1,700 videos (50 videos per class) from 34 categories are randomly sampled to construct the query set, while the rest videos are all put into the retrieval pool. The basic network architecture is the 19-layer VGGNet [24]. Different from the previous two adaptations, the architecture of adversary stream here is FC256-ReLU-FC256-ReLU-FC1, while the hash stream contains three FC layers that have 256, 256 and K neurons respectively. The features from second FC layers of both adversary and hash streams are aggregated. In the training on video dataset, 150 video clips are randomly sampled in each original video. Then, five frames are uniformly sampled in each clip. Finally, we averagely pool the CNN features of five frames as the representation of each clip. The optimization is performed on clip level. In testing, we uniformly sample 25 frames in each video and input each frame into the architecture of hash stream to produce the feature vector of each frame. The final hash codes of each video are generated by averagely pooling the feature vectors of all the 25 frames plus the quantization operation.

Parameters configuration. All of our proposed methods are implemented on Caffe [7] framework. We follow the suggested settings in [20] for deep adversarial training and optimize our framework by utilizing mini-batch stochastic gradient descent with 0.9 momentum. To balance the hash stream and adversary stream in DeDAHA, the scale parameter α is determined by using a validation set and is finally set to 0.1. The batch size is 32. The initial learning rate is 0.0001 and we decrease it to 10% after 5k iterations on Digits and real image to image adaptations, and 2k iterations on image to video adaptation. The total iteration number is 15k.

4.3 Protocols and Baseline Methods

We follow four evaluation protocols, i.e., mean average precision (MAP), hash lookup within Hamming radius 2, precision-recall curve and precision curves w.r.t. different numbers of top returned samples, which are widely used in [3, 11, 14, 34]. We compare the following approaches for performance evaluation: (1) Source-domain Hashing (SH) directly exploits the model trained on source domain to predict hash codes of the examples in target domain. (2) Target-domain Hashing (TH) trains the model only on the sparsely labeled examples in target domain. (3) Mix-domain Hashing (MH) optimizes the model with mixed labeled data from both source and target domain. In each triplet, the three examples are from the same domain. (4) Fine-Tune Hashing (FTH) learns the model first on the training data in source domain and then fine-tunes the model with the labeled data from target domain. (5) Domain Confusing Hashing (DCH) combines deep hashing model with domain confusion loss [26]. A domain classifier is trained to minimize the discrepancy between domains. (6) Maximum Mean Discrepancy Hashing (MMDH) leverages deep hashing model with maximum mean discrepancy (MMD) loss [28]. One MMD layer is used for domain-invariant learning. (7) Multi-Kernel Maximum Mean Discrepancy Hashing (MK-MMDH) [30] utilizes three multi-kernel MMD (MK-MMD) layers for domain adaptation in deep hashing network. (8) DeDAHA and DeDAHA$^-$ are our proposed approaches.

For fair comparison, the hash stream in all methods is the same. Raw images are fed as inputs for Digits and CIFAR-10 datasets. For ImageNet-C, we resize all images to 224×224 pixels. For ImageNet-F and FCVID-S, we follow the standard settings in video classification [19], and resize each image/frame to 320 × 240 pixels and then crop a 224 × 224 patch as the input.

Table 2: Mean Average Precision (MAP) of domain adaptation hashing between Digits datasets. The performances by leveraging different number of labeled examples from target domain are reported on three transfer directions.

labeled samples per class (in target domain)	Method	USPS → MNIST (MAP)				MNIST → USPS (MAP)				SVHN → MNIST (MAP)			
		12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
0 labeled sample/class	SH	0.388	0.375	0.407	0.387	0.512	0.527	0.579	0.540	0.343	0.442	0.460	0.499
	DeDAHA$^-$	0.558	0.611	0.605	0.561	0.621	0.611	0.666	0.663	0.470	0.533	0.546	0.525
3 labeled samples/class	TH	0.429	0.418	0.428	0.447	0.430	0.431	0.466	0.424	0.429	0.418	0.428	0.447
	MH	0.514	0.475	0.475	0.502	0.556	0.570	0.603	0.580	0.450	0.453	0.464	0.451
	FTH	0.453	0.496	0.555	0.511	0.477	0.506	0.497	0.489	0.467	0.511	0.510	0.449
	DCH	0.470	0.497	0.547	0.531	0.474	0.486	0.498	0.485	0.438	0.515	0.556	0.508
	MMDH	0.489	0.490	0.535	0.513	0.477	0.480	0.491	0.499	0.454	0.510	0.532	0.511
	MK-MMDH	0.554	0.566	0.561	0.543	0.543	0.563	0.584	0.559	0.535	0.524	0.548	0.545
	DeDAHA	**0.594**	**0.652**	**0.624**	**0.584**	**0.698**	**0.699**	**0.724**	**0.695**	**0.632**	**0.630**	**0.624**	**0.641**
5 labeled samples/class	TH	0.443	0.480	0.501	0.524	0.597	0.614	0.621	0.623	0.443	0.480	0.501	0.524
	MH	0.541	0.615	0.586	0.534	0.602	0.653	0.674	0.633	0.473	0.482	0.512	0.511
	FTH	0.567	0.622	0.587	0.598	0.638	0.634	0.662	0.635	0.480	0.543	0.557	0.533
	DCH	0.524	0.605	0.592	0.593	0.654	0.655	0.656	0.661	0.562	0.589	0.641	0.641
	MMDH	0.512	0.593	0.601	0.581	0.622	0.640	0.631	0.655	0.531	0.553	0.588	0.571
	MK-MMDH	0.589	0.625	0.635	0.639	0.631	0.668	0.651	0.696	0.588	0.589	0.647	0.650
	DeDAHA	**0.687**	**0.671**	**0.668**	**0.689**	**0.723**	**0.755**	**0.745**	**0.744**	**0.665**	**0.673**	**0.668**	**0.679**
10 labeled samples/class	TH	0.572	0.611	0.641	0.657	0.629	0.684	0.698	0.708	0.572	0.611	0.641	0.657
	MH	0.660	0.657	0.664	0.662	0.683	0.676	0.677	0.670	0.491	0.493	0.521	0.513
	FTH	0.602	0.688	0.711	0.686	0.662	0.669	0.674	0.664	0.526	0.555	0.562	0.567
	DCH	0.691	0.694	0.720	0.718	0.743	0.723	0.741	0.755	0.660	0.663	0.653	0.646
	MMDH	0.688	0.678	0.712	0.703	0.733	0.712	0.730	0.741	0.640	0.651	0.632	0.631
	MK-MMDH	0.701	0.694	0.741	0.740	0.745	0.728	0.751	0.764	0.678	0.675	0.663	0.661
	DeDAHA	**0.744**	**0.752**	**0.791**	**0.784**	**0.763**	**0.777**	**0.775**	**0.779**	**0.721**	**0.716**	**0.711**	**0.708**
15 labeled samples/class	TH	0.631	0.669	0.661	0.669	0.681	0.721	0.721	0.747	0.631	0.669	0.661	0.669
	MH	0.696	0.699	0.686	0.694	0.698	0.714	0.732	0.743	0.533	0.537	0.531	0.520
	FTH	0.708	0.740	0.754	0.763	0.682	0.684	0.691	0.694	0.540	0.576	0.591	0.581
	DCH	0.706	0.744	0.758	0.754	0.757	0.756	0.760	0.766	0.676	0.685	0.702	0.700
	MMDH	0.701	0.741	0.744	0.741	0.744	0.742	0.743	0.759	0.666	0.675	0.693	0.684
	MK-MMDH	0.734	0.767	0.766	0.773	0.758	0.760	0.790	0.781	0.686	0.688	0.704	0.715
	DeDAHA	**0.767**	**0.790**	**0.799**	**0.795**	**0.814**	**0.831**	**0.822**	**0.815**	**0.745**	**0.744**	**0.743**	**0.753**
20 labeled samples/class	TH	0.659	0.673	0.679	0.693	0.776	0.773	0.792	0.786	0.659	0.673	0.679	0.693
	MH	0.724	0.751	0.739	0.760	0.750	0.742	0.755	0.748	0.560	0.576	0.562	0.565
	FTH	0.717	0.759	0.760	0.787	0.693	0.713	0.716	0.716	0.570	0.636	0.634	0.611
	DCH	0.717	0.764	0.779	0.787	0.794	0.818	0.801	0.818	0.689	0.702	0.703	0.712
	MMDH	0.709	0.761	0.771	0.785	0.783	0.791	0.794	0.801	0.671	0.688	0.699	0.701
	MK-MMDH	0.743	0.797	0.789	0.809	0.792	0.833	0.828	0.825	0.694	0.713	0.717	0.716
	DeDAHA	**0.785**	**0.804**	**0.831**	**0.825**	**0.826**	**0.850**	**0.851**	**0.864**	**0.772**	**0.781**	**0.779**	**0.771**

4.4 Domain Adaptation between Digit Datasets

Table 2 shows the MAP performance comparisons on three transfer directions. Overall, the results across different number of hash bits and three adaptations indicate that our DeDAHA consistently outperforms others. In particular, the MAP of DeDAHA with 48 bits makes the absolute improvement over the best competitor MK-MMDH by 4.1%, 13.6% and 9.6% when exploiting only 3 target labeled examples on the adaptation of USPS → MNIST, MNIST → USPS and SVHN → MNIST, respectively. As expected, the performances of all the supervised transfer hashing are constantly boosted up with the increase of labeled examples from target domain. DeDAHA$^-$ leveraging unlabeled target data exhibits significantly better performance than SH which capitalizes on only source data. The result basically indicates the advantage of exploring the shift between source and target distribution for adaptation.

Directly reapplying a model trained in source domain (SH) could be much worse than re-developing a new model (TH) with more than 5 labeled examples per class from target domain, demonstrating the domain gap. By simply mixing source and target data in the training or fine-tuning the source model with target data, MH and FTH lead to better performances than TH when there are very few training examples (3 or 5 per class in our case) available in target domain. DCH improves MMDH, but the performance is still lower than MK-MMDH. The results indicate that improvements can be generally expected when extending the measure of domain disparity from MMD to adversarial learning or MK-MMD. Though both DCH and DeDAHA model domain shift in an adversarial manner, they are fundamentally different in the way that DCH is as a result of viewing the two domains identically and sharing the network parameters, while DeDAHA is by untying weight sharing and allowing the method to learn parameters for each domain individually. As indicated by our results, untied weight sharing leads to effective adaptation. This somewhat reveals the weakness of network sharing between two domains, which is to enforce the representation domain-invariant. DeDAHA, in comparison, is benefited from the mechanism of independently network learning, which explicitly models the domain difference. More importantly, hash learning in DeDAHA is enhanced by augmenting domain-specific representation and quantization with domain-invariant knowledge through the interaction across streams.

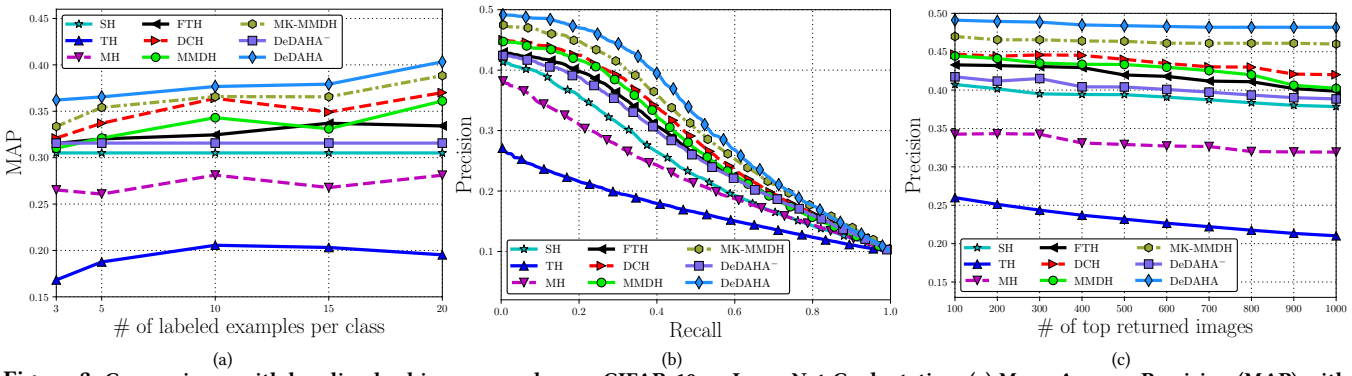


Figure 3: Comparisons with baseline hashing approaches on CIFAR-10 → ImageNet-C adaptation. (a) Mean Average Precision (MAP) with 48 bits by leveraging different number of labeled examples from target domain. (b) Precision-Recall curves with 48 bits (20 target labeled examples per class). (c) Precision curves with 48 bits w.r.t. different number of top returned samples (20 target labeled examples per class).

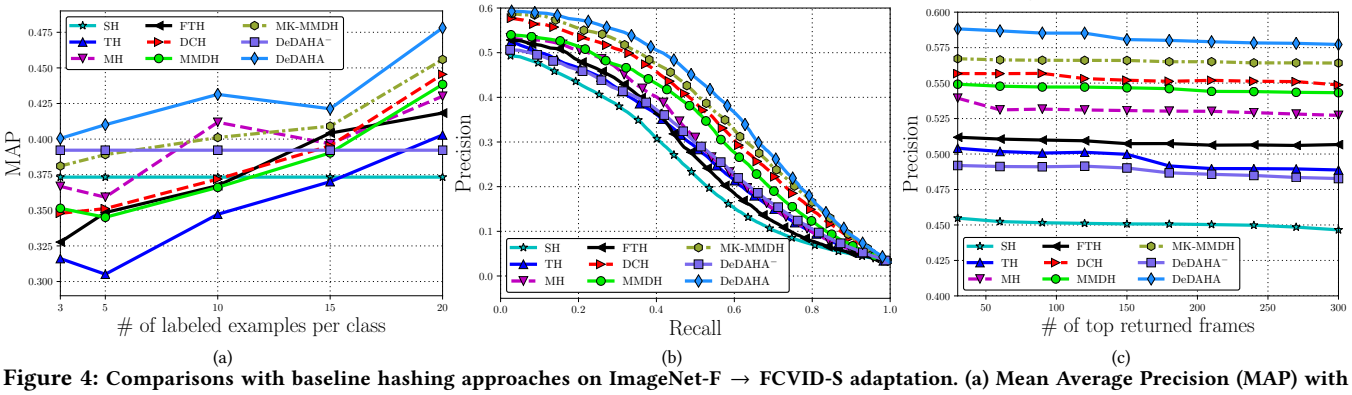


Figure 4: Comparisons with baseline hashing approaches on ImageNet-F → FCVID-S adaptation. (a) Mean Average Precision (MAP) with 48 bits by leveraging different number of labeled examples from target domain. (b) Precision-Recall curves with 48 bits (20 target labeled examples per class). (c) Precision curves with 48 bits w.r.t. different number of top returned samples (20 target labeled examples per class).

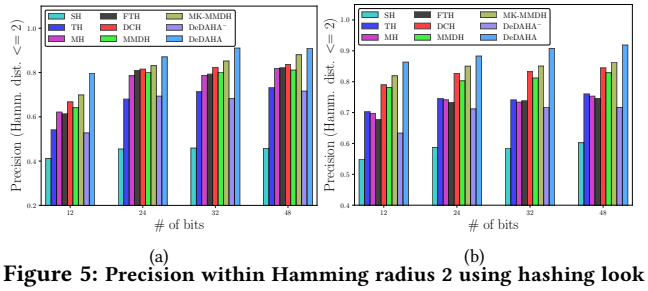


Figure 5: Precision within Hamming radius 2 using hashing look up performance (20 target labeled examples per class). (a) USPS → MNIST adaptation. (b) MNIST → USPS adaptation.

The evaluations of hash lookup within Hamming radius 2 are given in Figure 5 on the adaptation of USPS→MNIST and MNIST → USPS with 20 labeled examples per class from target domain. Although the number of samples falling into a bucket decreases exponentially for larger bits of hash codes, the precisions for most of the methods still have improvements with the increase of hash bits. Particularly, the precision of our DeDAHA even has a clear boost from 90.8% of 32 bits to 91.8% of 48 bits in the transfer direction of MNIST → USPS, verifying the effectiveness of DeDAHA.

4.5 Image to Image Adaptation and Image to Video Adaptation

The second and third experiment was conducted on more challenging real image to image and image to video adaptation, respectively. Figure 3 and Figure 4 detail the performances on the adaptation of CIFAR-10 → ImageNet-C and ImageNet-F → FCVID-S. Our DeDAHA consistently outperforms other runs on two adaptations in terms of MAP when leveraging different numbers of labeled examples in target domain, Precision-Recall curves with 48 bits and Precision curves with 48 bits w.r.t. different number of top returned samples. With the increase of labeled examples in target domain, the performances of all the supervised transfer hashing in general gradually increase on both adaptations. Similar to the observations on adaptations between Digit datasets, DeDAHA$^-$ performs better than SH, which verifies our domain shift modeling through adversarial learning. SH constantly exhibits better performance than TH on the adaptation of CIFAR-10 → ImageNet-C. In the transfer of ImageNet-F → FCVID-S, the MAP of TH is much lower than that of SH especially when there are very few labeled examples available from FCVID-S dataset (less than 15 in our case). Furthermore, TH starts to surplus the performance of SH when 20 labeled examples per class are available for training. The results give a clue that transferring from image to video is much harder than image to image. In other words, the gap across different image datasets is in general smaller than that between image and video domains.

MH and FTH involve utilization of both source and target labeled data. Different training strategy is employed by each in the way that MH is as a result of directly mixing source and target data in the training and treating the two domains equally, while FTH is by training the model in a two-stage scheme and at each stage

Figure 6: Examples showing the top 10 retrieval results by different methods in response to two query videos in FCVID-S dataset (better viewed in color). Each video is represented by one sampled frame. In each row, the first video with a red bounding box is the query video and the videos belong to the same category of the query video are regarded as correct ones, which are enclosed in a blue bounding box.

Table 3: Mean Average Precision (MAP) performance comparisons between DeDAHA and DeDAHA-I (w/o interaction) or DeDAHA-A (w/ interaction by element-wisely summating two representations). The results are reported with 20 labeled target samples/class on the adaptation of CIFAR-10→ImageNet-C and ImageNet-F→FCVID-S.

Method	CIFAR-10 → ImageNet-C				ImageNet-F → FCVID-S			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
DeDAHA	**0.422**	**0.428**	**0.411**	**0.405**	**0.471**	**0.482**	**0.478**	**0.478**
DeDAHA-A	0.415	0.420	0.392	0.397	0.466	0.471	0.470	0.471
DeDAHA-I	0.391	0.403	0.385	0.383	0.459	0.456	0.458	0.462

only the labeled data from one domain is exploited. As indicated by our results, FTH leads to better performance gain than MH on CIFAR-10 → ImageNet-C transfer. This observation is not surprise because the domain gap between CIFAR-10 and ImageNet-C datasets is not too large and thus directly fine-tuning the source model with target data tends to adapt the model to the target domain. Instead, the improvement of MH is more obvious than FTH on the adaptation of ImageNet-F → FCVID-S. This somewhat reveals the weakness of fine-tuning. When the domain gap is large, directly fine-tuning may degrade the source model sharply. In contrast, MH by mixing the source and target data could alleviate the effects of domain shift. Similar to the observations on adaptations between digital datasets, DCH and MK-MMDH outperform MMDH on both transfers. Compared to DCH, the performance gain of our DeDAHA is much larger in ImageNet-F → FCVID-S transfer than that of adaptation from CIFAR-10 to ImageNet-C. The results again indicate that DCH enforcing domain invariance will be detrimental to discriminative power when there is a big domain difference.

Figure 6 showcases the top ten video search results by different methods in response to two query videos. Each video is represented by one sampled frame. We can see that DeDAHA achieves the most satisfying results, and retrieves nine correct videos in the returned top ten videos to each query video, respectively.

4.6 The Effect of Stream Interaction

Next, we turn to evaluate how hashing performance is affected when streams of hashing and adversarial learning interact through DeDAHA training. We designed two runs for comparisons, i.e., DeDAHA-A and DeDAHA-I. DeDAHA-A performs interactions by element-wisely summating the intermediate representations from hash and adversary streams, and DeDAHA-I decouples the interaction between the two streams, in which each stream is optimized independently. The MAP performance comparisons across different hash bit on the adaptation of CIFAR-10 → ImageNet-C and ImageNet-F → FCVID-S are summarized in Table 3. Overall, DeDAHA and DeDAHA-A both exhibit better performance than DeDAHA-I. The results indicate the advantage of exploring interaction across two streams and endowing domain-specific hashing more power with domain-invariant knowledge learnt in adversary stream. Though DeDAHA and DeDAHA-A both involve utilization of stream interaction, they are different in the way of representation augmentation. DeDAHA is as a result of concatenating the representations from two streams and DeDAHA-A is by element-wisely summating the two intermediate representations. As indicated by our results, concatenation can lead to better performance than element-wise summation.

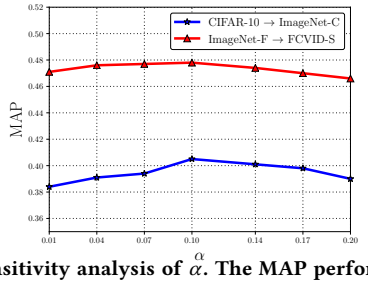


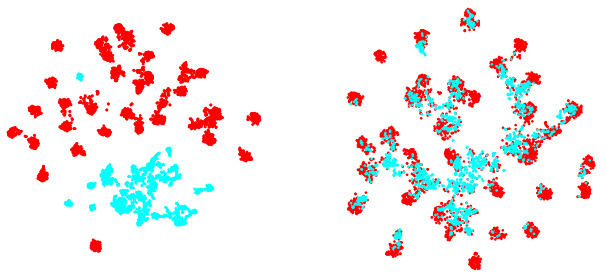
Figure 7: Sensitivity analysis of α. The MAP performances on 48 bits are reported with 20 labeled target samples/class.

4.7 Parameter Sensitivity

A common problem with combination of multiple losses is the need to set the tradeoff parameters in between. Figure 7 shows the MAP performance of DeDAHA with respect to different α in Eq.(8) on two adaptations. As shown in the Figure, the two curves are both like "∧" shape when α varies from 0.01 to 0.2. This validates the joint learning of hash stream and adversary stream to boost hashing in the context of domain adaptation.

4.8 Binary Representation Visualization

Figure 8 depicts the t-SNE [29] visualization of hash codes obtained by SH and DeDAHA. Specifically, we randomly select 3,400 samples from 34 classes (100 samples per class) in each domain (ImageNet-F as source and FCVID-S as target) and the obtained hash codes are then projected into 2-dimensional space using t-SNE. We can see that the distribution of target samples is far from that of source samples before domain adaptation. Through domain adaptation by our DeDAHA, the two distributions are brought closer, making the target distribution indistinguishable from the source one.



(a) (b)

Figure 8: Hash codes visualizations of source and target domain using t-SNE. Each image/video is visualized as one point and colors denote different domains (red: source, blue: target). (a) Before domain adaptation (i.e., SH). (b) After domain adaptation by our DeDAHA.

5 CONCLUSIONS

We have presented a Deep Domain Adaptation Hashing with Adversarial learning (DeDAHA) architecture which explores adversarial learning to model domain shift for binary representation generation. Particularly, we study the problem from three perspectives: 1) when there are sparsely or not any labeled target data available; 2) how to model domain distribution in an adversarial manner; 3) exploring the interaction between the streams of hash learning and domain discriminator. To verify our claim, we optimize the whole architecture of our hashing model by simultaneously fooling the domain discriminator with the learnt two distributions and preserving relative similarity between images/frames. Experiments conducted on three types of domain transfer tasks validate our proposal and analysis. Performance improvements are clearly observed when comparing to other domain adaptation methods on hashing.

Our future works are as follows. First, binary representation learning can be further enhanced by explicitly considering semantics of the image/frame. Hence, a classification stream could be exploited in addition to current two streams to leverage semantic supervision. Second, more in-depth studies of how two streams should be interacted to boost hashing will be investigated. Third, the idea of alleviating domain shift with adversarial learning will be explored for other problems, e.g., semantic segmentation.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61572451, Youth Innovation Promotion Association of the Chinese Academy of Sciences CX2100060016, and Fok Ying Tung Education Foundation WF2100060004.

REFERENCES

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.
[2] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaption by backpropagation. In *ICML*.
[3] Yunchao Gong and Svetlana Lazebnik. 2011. Iterative Quantization: A Procrustean Approach to Learning Binary Codes. In *CVPR*.
[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.
[5] J. Huang, A. Smola, A. Gretton, KM. Borgwardt, and B. Schölkopf. 2007. Correcting Sample Selection Bias by Unlabeled Data. In *NIPS*.
[6] Ian J.Goodfellow, Jeaen Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*.
[7] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM MM*.
[8] Yu-Gang Jiang, Zuxuan Wu, Jun Wang, Xiangyang Xue, and Shih-Fu Chang. 2017. Exploiting feature and class relationships in video categorization with regularized deep neural networks. *TPAMI* (2017).
[9] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. *Master's Thesis, Department of Computer Science, University of Toronto* (2009).
[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*.
[11] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous Feature Learning and Hash Coding with Deep Neural Networks. In *CVPR*.
[12] Yann Lecun, LÃlon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* (1998).
[13] Pengfei Li. 2015. Transfer Learning for Information Retrieval. In *ACM SIGIR*.
[14] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised Hashing with Kernels. In *CVPR*.
[15] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning Transferable Features with Deep Adaptation Networks. In *ICML*.
[16] Yuval Netzer, Tao Wang, Adam Coates, Ro Bissacco, Bo Wu, and Andrew Y. Ng. 2012. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS workshop on Deep Learning and Unsupervised Feature Learning*.
[17] Yingwei Pan, Ting Yao, Houqiang Li, Chong-Wah Ngo, and Tao Mei. 2015. Semi-supervised hashing with semantic confidence for large scale visual search. In *ACM SIGIR*.
[18] Zhaofan Qiu, Yingwei Pan, Ting Yao, and Tao Mei. 2017. Deep Semantic Hashing with Generative Adversarial Networks. In *ACM SIGIR*.
[19] Zhaofan Qiu, Ting Yao, and Tao Mei. 2017. Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks. In *ICCV*.
[20] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *ICLR*.
[21] Mohammad Rastegari, Ali Farhadi, and David Forsyth. 2012. Attribute discovery via predictable discriminative binary codes. In *ECCV*.
[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV* (2015).
[23] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic Hashing. *International Journal of Approximate Reasoning* (2009).
[24] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks For Large-Scale Image Recognition. In *ICLR*.
[25] Baochen Sun and Kate Saenko. 2016. Deep CORAL: correlation alignment for deep domain adaption. In *ICCV workshop on Transferring and Adapting Source Knowledge in Compute Vision (TASK-CV)*.
[26] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous deep transfer across domains and tasks. In *ICCV*.
[27] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial Discriminative Domain Adaption. In *CVPR*.
[28] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep Domain Confusion: Maximizing for Domain Invariance. *CoRR* (2014). http://arxiv.org/abs/1412.3474
[29] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* (2008).
[30] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep Hashing Network for Unsupervised Domain Adaption. In *CVPR*.
[31] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2012. Semi-Supervised Hashing for Large-Scale Search. *TPAMI* (2012).
[32] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2017. A Survey on Learning to Hash. *TPAMI* (2017).
[33] Yair Weiss, Antonio Torralba, and Rob Fergus. 2008. Spectral Hashing. In *NIPS*.
[34] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. 2014. Supervised Hashing for Image Retrieval via Image Representation Learning. In *AAAI*.
[35] Saining Xie, Ross Girshick, Piotr Dollar, Zhouwen Tu, and Kaiming He. 2017. Aggregated Residual Transformations for Deep Neural Networks. In *CVPR*.
[36] Ting Yao, Fuchen Long, Tao Mei, and Yong Rui. 2016. Deep semantic-preserving and ranking-based hashing for image retrieval. In *IJCAI*.
[37] Ting Yao, Yingwei Pan, Chong-Wah Ngo, Houqiang Li, and Tao Mei. 2015. Semi-supervised domain adaptation with subspace learning for visual recognition. In *CVPR*.
[38] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *NIPS*.
[39] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. 2018. Fully Convolutional Adaptation Networks for Semantic Segmentation. In *CVPR*.