# Effective Comparative Prototype Hashing for Unsupervised Domain Adaptation

**Hui Cui[1, 2], Lihai Zhao[3], Fengling Li[4], Lei Zhu[5*] Xiaohui Han[1,2,6*], Jingjing Li[7]**

[1]Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences)
[2]Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science
[3]University of Science and Technology Beijing
[4]University of Technology Sydney
[5]Tongji University
[6]Quan Cheng Laboratory
[7]University of Electronic Science and Technology of China
{cuihui2018, zhaolihai2020, fenglingli2023, leizhu0608, xiaohhan}@gmail.com,lijin117@yeah.net

## Abstract

Unsupervised domain adaptive hashing is a highly promising research direction within the field of retrieval. It aims to transfer valuable insights from the source domain to the target domain while maintaining high storage and retrieval efficiency. Despite its potential, this field remains relatively unexplored. Previous methods usually lead to unsatisfactory retrieval performance, as they frequently directly apply slightly modified domain adaptation algorithms to hash learning framework, or pursue domain alignment within the Hamming space characterized by limited semantic information. In this paper, we propose a simple yet effective approach named *Comparative Prototype Hashing* (CPH) for unsupervised domain adaptive image retrieval. We establish a domain-shared unit hypersphere space through prototype contrastive learning and then obtain the Hamming hypersphere space via mapping from the shared hypersphere. This strategy achieves a cohesive synergy between learning uniformly distributed and category conflict-averse feature representations, eliminating domain discrepancies, and facilitating hash code learning. Moreover, by leveraging dual-domain information to supervise the entire hashing model training process, we can generate hash codes that retain inter-sample similarity relationships within both domains. Experimental results validate that our CPH significantly outperforms the state-of-the-art counterparts across multiple cross-domain and single-domain retrieval tasks. Notably, on Office-Home and Office-31 datasets, CPH achieves an average performance improvement of 19.29% and 13.85% on cross-domain retrieval tasks compared to the second-best results, respectively. The source codes of our method are available at: https://github.com/christinecui/CPH.

## Introduction

The efficacy of hashing technique in the field of retrieval has been firmly established through extensive research (Wang et al. 2016, 2018; Zhu et al. 2024). While hashing offers inherent advantages in terms of data storage and retrieval efficiency, it frequently encounters challenges when aiming to achieve satisfactory retrieval performance in real-world unsupervised scenarios. This is primarily due to the absence of semantic information that guides the hashing model in learning similarity-preserving hash codes. Transferring valuable knowledge from relevant and labeled data (referred to as source domain) to enhance the retrieval performance of current unsupervised hashing models stands as a sensible and promising strategy.

However, the formulation mentioned above encounters a critical issue: the distribution of the source domain and the current dataset (referred to as the target domain) may diverge. Directly applying the model trained on the source domain to the target domain might lead to a significant performance degradation. This concern is confirmed by the results from the well-known domain adaptation method DANN (Ganin and Lempitsky 2015). The model trained on the MNIST dataset achieves a remarkable classification accuracy of 95.96% on MNIST itself. However, when this identical model is applied to the MNIST-M dataset, its classification accuracy plummets to 52.25%. In response to this challenge, the concept of domain adaptation emerged as a solution (Tzeng et al. 2015; Sun, Feng, and Saenko 2016; Zhang, Li, and Ogunbona 2017). Its objective is to bridge the disparities in domain distributions and facilitate the transfer of valuable knowledge across distinct domains. As a consequence, this concept contributes to improving performance within the target domain.

Currently, only a limited number of unsupervised domain adaptive hashing methods have been proposed (Zhou et al. 2018; Liu and Zhang 2019; Huang, Zhang, and Gao 2022). Regrettably, none of them have managed to achieve a satisfactory level of retrieval performance. This is demonstrated by the results obtained from the pioneering approach PWCF (Huang et al. 2020), where the mAP values across six cross-domain retrieval tasks on Office-Home dataset barely reach around 30%. Such results raise questions regarding whether PWCF effectively transfers knowledge from the source domain to the target domain. Even the most recent method, DANCE (Wang et al. 2023b), does not surpass an average retrieval performance of 50% on Office-Home. Thus, the primary objective of this paper is to enhance the retrieval

*Corresponding authors.

performance of unsupervised domain adaptive hashing, encompassing both cross-domain and single-domain retrieval tasks.

After a thorough analysis of the existing unsupervised domain adaptive hashing methods, we have discovered that early methods overly relied on general domain adaptation paradigms, leading to a lack of effective integration between domain adaptation and hash learning. For example, DAH (Venkateswara et al. 2017a) and the well-regarded domain adaptation method DAN (Long et al. 2015) exhibit a significant degree of similarity. The only distinctions are that DAH substitutes the final layer of DAN with a hash layer and employs a commonly used negative log-likelihood function in hash learning to train its model. DAH-IGAN (He et al. 2019) incorporates the GAN concept from DANN (Ganin and Lempitsky 2015) into its hashing framework, training the model by simultaneously maximizing the domain discriminator loss and minimizing the label predictor loss in a manner similar to DANN. In the past two years, methods have shifted their focus towards generating reliable pseudo-labels for the target domain, which in turn guide the learning process of hashing models, such as DHLing (Xia et al. 2021), PEACE (Wang et al. 2023a), and DANCE. However, these approaches share a common characteristic: they perform domain alignment in the Hamming space. Since the dimension of the Hamming space is usually much smaller than that of the feature space, conducting domain alignment in the Hamming space may lead to the loss of the rich semantic information embedded in the original samples. As a consequence, this has the potential to impact the ultimate retrieval performance of domain adaptive hashing.

Different from methods mentioned above, we propose a simple yet effective method, dubbed as *Comparative Prototype Hashing* (CPH). The core idea of CPH is to establish a domain-shared unit hypersphere space corresponding to the Hamming space of hash codes. Specifically, CPH maximizes the distance between prototypes belonging to different categories, thus encouraging a uniform distribution of features on the hypersphere of feature space and avoiding category conflict. Additionally, by aligning prototypes from both the source and target domains, CPH enhances the compactness of feature representations for samples within the same category across both domains. This prototype-based comparative learning not only promotes domain alignment but also advances the learning of discriminative feature representations, potentially transforming the relationships between samples from the source and target domains into relationships between prototypes. Besides, CPH incorporates the relation preservation and quantization from both domains into the hash learning framework. Empowered by these strategies, CPH effectively learn discriminative hash codes, resulting in a substantial enhancement in retrieval performance across cross-domain and single-domain retrieval tasks. The workflow of our approach is illustrated in Figure 1, and the primary contributions of our method can be summarized as follows:

- We seamlessly integrate domain adaptation and hash learning within a straightforward framework utilizing a domain-shared unit hypersphere space, resulting in a significant enhancement of retrieval performance in the context of unsupervised domain adaptive hashing.
- Technically, we perform prototype comparative learning to obtain the domain-shared space, and map this space into a Hamming space under the constraints of semantic relations and quantization in both the source and target domains.
- Extensive experimental results demonstrate that the proposed CPH method outperforms state-of-the-art unsupervised domain adaptive hashing methods in terms of retrieval accuracy, both in single-domain and cross-domain scenarios. Particularly, when compared to the second-best results on Office-Home and Office-31 databases, the average improvement in cross-domain retrieval is 19.29% and 13.85%.

## Related Work

### Learning to Hash

Hashing aims to learn hash functions that encode high-dimensional data into binary hash codes while maintaining semantic relationships. It offers advantageous attributes in terms of data storage efficiency and retrieval speed, which has consequently garnered significant attention (Zhu et al. 2020; Lu et al. 2019; Cui et al. 2020, 2021). Hashing can be broadly categorized based on its reliance on semantic labels: unsupervised hashing and supervised hashing. Supervised hashing methods utilize explicit labels as supervised information to learn discriminative hash codes. Examples include SDH (Shen et al. 2015), DSRH (Zhao et al. 2015), NINH (Lai et al. 2015) and DPSH (Li, Wang, and Kang 2016). However, despite their capability to achieve commendable performance, the substantial cost associated with annotating labels poses a significant obstacle to their scalability. In contrast, unsupervised hashing methods exhibit superior scalability compared to their supervised counterparts since they are independent of semantic labels. As a result, unsupervised hashing stands as a suitable solution for practical retrieval systems. Representative approaches in this category include LSH (Gionis, Indyk, and Motwani 1999), ITQ (Gong et al. 2013), SGH (Jiang and Li 2015), and GraphBit (Wang et al. 2023c). Nonetheless, the absence of explicit semantic label supervision in unsupervised scenarios can potentially limit the discriminative capability of hash codes, thereby undermining overall retrieval performance. This situation underscores the pressing need for cost-effective alternatives to manual annotation, which can serve as viable sources of semantic supervision in hashing.

### Unsupervised Domain Adaptation

Unsupervised domain adaptation aims to transfer knowledge acquired from a well-labeled source domain to a different yet related target domain where labeled data is unavailable (Pan and Yang 2010; Wang and Deng 2018; Xu et al. 2020). In this field, mainstream methods can be primarily categorized into two types: metric learning-based methods and adversarial learning-based methods. The former mitigates distribution gaps by minimizing statistical criteria. Representative approaches includes DAN (Long et al. 2015), RTN

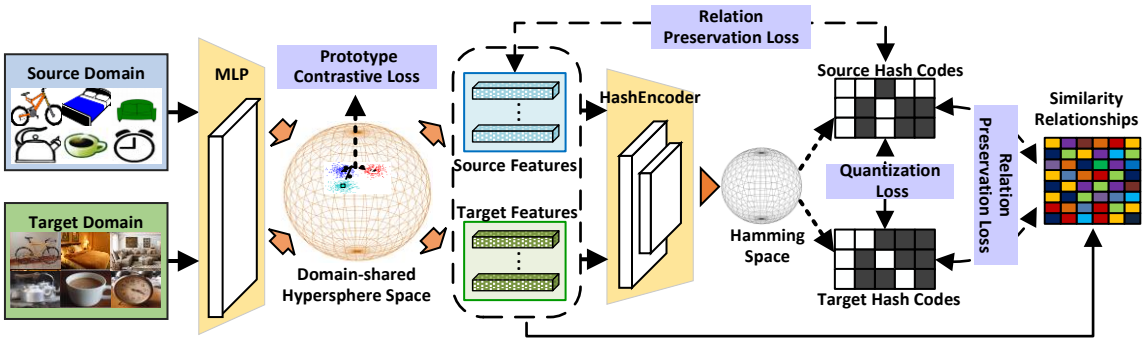


Figure 1: The workflow of the proposed PCH method.

(Long et al. 2016), JAN (Long et al. 2017), and CGDM (Du et al. 2021). The latter employs a domain classifier to distinguish whether a sample originates from the source or the target domain, while concurrently encouraging the feature generation network to extract domain-invariant representations through a two-player mini-max game. Noteworthy instances in this category include DANN (Ganin and Lempitsky 2015), ADDA (Tzeng et al. 2017), and AAA (Li et al. 2022). The domain adaptation field is continuously and vigorously evolving, potentially giving rise to a plethora of novel methodologies and concepts in the foreseeable future, along with fostering more extensive combinations with downstream tasks.

### Domain Adaptive Hashing

Over the last decade, a handful of domain adaptive hashing methods have emerged. Initial approaches are grounded in machine learning principles, such as LapITQ+ (Zhou et al. 2018), GTH (Liu and Zhang 2019), PWCF (Huang et al. 2020), and DAPH (Huang, Zhang, and Gao 2022). These methods commonly utilize distribution metrics to generate domain-invariant features. With the rise of deep learning, recent approaches have shifted towards deep learning-based domain adaptive hashing (Venkateswara et al. 2017a; Xia et al. 2021; Wang et al. 2023a,b). The strategies for domain adaptation have progressively expanded to encompass adversarial learning, construction of domain classifiers, parameter sharing across networks, alignment of pseudo-labels, and more. However, none of these methods have achieved satisfactory performance. They heavily relied on general domain adaptation paradigms, struggling to effectively integrate domain adaptation and hash learning. Enhancing the retrieval performance of domain adaptive hashing remains a compelling pursuit.

## Methodology

### Problem Definition

Assuming that we have a labeled source domain $D^s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ with $n_s$ samples and an unlabeled target domain $D^t = \{(\mathbf{x}_i^t)\}_{i=1}^{n_t}$ with $n_t$ samples. $D^s$ and $D^t$ exhibit distinct data distributions while sharing a common label space $\mathcal{Y} = \{1, 2, \cdots, C\}$. Our goal is to transfer knowledge from the source domain $D^s$ to assist the target domain $D^t$ in learning a hashing model that generates similarity-preserving hash codes $\mathbf{B}^t = [\mathbf{b}_1^t, \cdots, \mathbf{b}_{n_t}^t] \in \{-1, 1\}^{n_t \times r}$, where $r$ is the length of the hash code. For single domain retrieval, we investigate the retrieval system in which both queries and databases originate from the target domain $D^t$. In the context of cross-domain retrieval, we seek samples from the source domain $D^s$ that possess similar semantics to the given queries from the target domain $D^t$.

### Prototype Contrastive Learning

Previous studies have demonstrated the noteworthy progress in hash learning via contrastive learning (Qiu et al. 2021; Luo et al. 2021; Wang et al. 2022, 2023b). However, many of these methods consider instance-level contrastive learning, wherein samples other than the augmented view of the current sample within the same batch are considered as negative samples. These negative samples may inevitably lead to the category collision issue. Additionally, they usually directly apply the contrastive loss to hash codes, lacking the ability to effectively incorporate the semantic content inherent in the samples. To address these concerns, we conduct category-level contrastive learning based on a domain-shared feature representation space. By treating prototypes from the same category in both source and target domains as positive pairs, and those from different categories as negative pairs, we facilitate the alignment of the source and target domains in the shared feature space. Furthermore, this approach enables simultaneous learning of uniform, conflict-averse feature representations across both domains.

To achieve the aforementioned objectives, we first calculate the source prototype codes based on the source labels. The learning process for the prototype code of the $c$-th category ($c \in \mathcal{Y}$) can be described as follows:

$$\mathbf{p}_c^s = \frac{\sum_i^{n_s} \mathbf{f}_i^s \mathbb{1}\left(y_i^s = c\right)}{\sum_i^{n_s} \mathbb{1}\left(y_i^s = c\right)}, \mathbf{p}_c^s \leftarrow \frac{\mathbf{p}_c^s}{\|\mathbf{p}_c^s\|_2}, \tag{1}$$

where $\mathbf{f}_i^s$ represents the feature representation of the source domain sample in the shared feature space. This feature representation is obtained using a Multi-Layer Perceptron architecture $\mathbf{MLP}(\cdot; \theta_{mlp})$ as follows:

$$\mathbf{f}_i^s = \mathbf{MLP}\left(\mathbf{x}_i^s; \theta_{mlp}\right). \tag{2}$$

Here, $\theta_{mlp}$ denotes the trainable parameters. $\mathbb{1}(\cdot)$ acts as an indicator function that returns 1 if the argument is true and 0

otherwise. It's important to note that $\mathbf{p}_c^s$ represents the prototype code that considers all training data from $D^s$. During the training process, $\mathbf{p}_c^s$ is initialized using the prototype code learned from the first epoch. Subsequently, in each successive epoch, $\mathbf{p}_c^s$ is updated based on the sampled data.

Subsequently, we obtain pseudo-labels for the target data using a nearest source prototype approach. To be precise, the pseudo-label for the $j$-th target sample is obtained through the following process:

$$\hat{y}_j^t = \arg\max_c cos\left(\mathbf{f}_j^t, \mathbf{p}_c^s\right), \tag{3}$$

where $cos(\cdot)$ signifies the cosine similarity function.

After obtaining pseudo-labels for target domain samples, we can proceed to deduce the target prototype codes, applying a method similar to that used for source prototypes. Specifically, the prototype code for the $c$-th category in the target domain is determined by the following process:

$$\mathbf{p}_c^t = \frac{\sum_j^{n_t} \mathbf{f}_j^t \mathbb{1}\left(\hat{y}_j^t = c\right)}{\sum_j^{n_t} \mathbb{1}\left(\hat{y}_j^t = c\right)}, \mathbf{p}_c^t \leftarrow \frac{\mathbf{p}_c^t}{\|\mathbf{p}_c^t\|_2}. \tag{4}$$

Likewise, $\mathbf{p}_c^t$ undergoes updates during each epoch.

We leverage prototypes from both the source and target domains to facilitate prototype adaptation through contrastive learning. Specifically, within the shared feature representation space, we possess $C$ prototypes from the source domain, denoted as $\{\mathbf{p}_1^s, \ldots, \mathbf{p}_C^s\}$, as well as an analogous set of $C$ prototypes from the target domain, represented as $\{\mathbf{p}_1^t, \ldots, \mathbf{p}_C^t\}$. Our approach designs a Prototype Contrastive Loss (PCL), which is formulated as follows:

$$\mathcal{L}_{PCL} = \frac{1}{C}\sum_{c=1}^{C} -\log \frac{\exp\left(\frac{(\mathbf{p}_c^s)^T \mathbf{p}_c^t}{\tau}\right)}{\exp\left(\frac{(\mathbf{p}_c^s)^T \mathbf{p}_c^t}{\tau}\right) + \sum\limits_{i=1, i\neq c}^{C} \exp\left(\frac{(\mathbf{p}_c^s)^T \mathbf{p}_i^t}{\tau}\right)} \tag{5}$$

$$\approx \underbrace{\frac{1}{C}\sum_{c=1}^{C} -\frac{(\mathbf{p}_c^s)^T \mathbf{p}_c^t}{\tau}}_{prototypical\ alignment} + \underbrace{\frac{1}{C}\sum_{c=1}^{C} \log \sum_{i=1, i\neq c}^{C} \exp\left(\frac{(\mathbf{p}_c^s)^T \mathbf{p}_i^t}{\tau}\right)}_{prototypical\ uniformity}. \tag{6}$$

Based on the analysis presented in ProPos (Huang et al. 2023), it has been determined that Eq. (5) can be approximated through the prototypical alignment and prototypical uniformity outlined in Eq. (6). This theoretical validation supports the simultaneous acquisition of uniform and category conflict-averse feature representations across both domains by our method. Additionally, this strategy implicitly transforms the fundamental relationships among samples into relationships among prototypes. As these learned prototypes offer greater discrimination than the original sample features, they further enhance our subsequent unsupervised domain adaptive hashing learning.

## Hash Code Learning

To achieve the core essence of hashing, we preserve the inherent relationships among samples into the hash codes. As our prototypical contrastive learning ensures that feature representations are uniformly distributed on a unit hypersphere, we consequently opt to maintain the similarity between samples in this feature space into the hash codes.

Firstly, given that we possess the labels from the source domain, a logical step is to utilize them for directly guiding our hash learning process. This can be formalized as follows:

$$\mathcal{L}_{rel1} = \|\eta \mathbf{S}^s - cos(\mathbf{H}^s, \mathbf{H}^s)\|_{\mathrm{F}}^2. \tag{7}$$

Here, $\eta$ stands for the scaling factor, $\mathbf{S}^s$ signifies the similarity matrix computed by the source labels. $\mathbf{H}^s = \left[\mathbf{h}_1^s, \cdots, \mathbf{h}_{n_s}^s\right] \in \mathbb{R}^{n_s \times r}$ corresponds to the relaxed hash codes of the source domain, which are generated by our HashEncoder. For the $i$-th source sample, $\mathbf{h}_i^s$ is computed as follows:

$$\mathbf{h}_i^s = \mathbf{HashEncoder}\left(\mathbf{f}_i^s; \theta_{hash}\right), \tag{8}$$

where $\theta_{hash}$ denotes the trainable parameters.

Secondly, to transfer knowledge from the source domain to the target domain and engage in unsupervised domain adaptive hash learning, we enforce the constraint that similar features in the domain-shared feature representation space of both domains should generate corresponding hash codes. This can be expressed as follows:

$$\mathcal{L}_{rel2} = \left\|cos(\mathbf{F}^s, \mathbf{F}^t) - cos(\mathbf{H}^s, \mathbf{H}^t)\right\|_{\mathrm{F}}^2. \tag{9}$$

Here, $\mathbf{F}^* = \left[\mathbf{f}_1^*, \cdots, \mathbf{f}_{n_*}^*\right] \in \mathbb{R}^{n_* \times d}$, and $\mathbf{H}^* = [\mathbf{h}_1^*, \cdots, \mathbf{h}_{n_*}^*] \in \mathbb{R}^{n_* \times r}$, with $* \in \{s, t\}$. They are produced using $\mathbf{MLP}(\cdot; \theta_{mlp})$ and $\mathbf{HashEncoder}(\cdot; \theta_{hash})$, respectively.

By combining Eq. (7) and Eq. (9), we formulate our Relation Preservation Loss (RPL) as follows:

$$\mathcal{L}_{RPL} = \gamma \mathcal{L}_{rel1} + (1-\gamma)\mathcal{L}_{rel2}, \tag{10}$$

where $\gamma$ serves as the fusion factor that balances the importance of each term. With this, we successfully integrate the relationships between samples from both domains into our hash learning framework.

To ensure that the relaxed hash codes approximate the binary hash codes, we apply a quantization process to the hash codes for both domains. The formulation of our Quantization Loss (QL) is as follows:

$$\mathcal{L}_{QL} = \frac{1}{2}\left\|\mathbf{B}^t - \mathbf{H}^t\right\|_{\mathrm{F}}^2 + \frac{1}{2}\left\|\mathbf{B}^s - \mathbf{H}^s\right\|_{\mathrm{F}}^2, \tag{11}$$

where $\mathbf{B}^* = sgn(\mathbf{H}^*) = [\mathbf{b}_1, \cdots, \mathbf{b}_{n_*}] \in \{-1, 1\}^{n_* \times r}$ represents the binary hash codes. $sgn(\cdot)$ is the element-wise sign function that returns 1 if the element is positive and returns $-1$ otherwise.

Finally, we establish the overall objective function for our CPH as follows:

$$\min \mathcal{L} = \lambda_1 \mathcal{L}_{PCL} + \lambda_2 \mathcal{L}_{QL} + \lambda_3 \mathcal{L}_{RPL}, \tag{12}$$

where $\lambda_1, \lambda_2, \lambda_3$ are trade-off parameters.

## Out of Sample Extension

After undergoing several rounds of iterative optimizations, our CPH model is thoroughly trained, resulting in the attainment of optimal network parameters. Subsequently, for any

| Method | Office-Home | | | | | | Office-31 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A→R | R→A | C→R | R→C | P→R | R→P | A→D | A→W | D→A | D→W | W→A | W→D |
| LSH | 11.49 | 11.45 | 6.94 | 7.24 | 12.24 | 13.45 | 16.04 | 15.35 | 13.60 | 43.99 | 14.67 | 38.80 |
| ITQ | 25.88 | 25.37 | 14.83 | 14.92 | 26.81 | 28.19 | 29.55 | 28.53 | 26.83 | 58.89 | 25.09 | 58.00 |
| DSH | 9.69 | 9.67 | 5.47 | 5.28 | 8.49 | 8.26 | 16.66 | 15.09 | 16.33 | 41.07 | 13.58 | 39.24 |
| SGH | 22.93 | 22.53 | 13.62 | 13.51 | 24.51 | 25.73 | 24.98 | 22.47 | 22.17 | 56.36 | 20.52 | 53.94 |
| GraphBit | 18.18 | 16.87 | 11.51 | 10.81 | 18.91 | 21.32 | 24.48 | 23.12 | 22.09 | 53.82 | 21.34 | 51.43 |
| ITQ+ | - | 14.25 | 9.55 | - | 17.61 | - | 17.99 | 15.00 | - | - | - | 42.29 |
| GTH-g | 16.95 | 17.54 | 8.46 | 11.88 | 17.82 | 18.57 | 30.85 | 18.44 | 21.99 | 48.48 | 20.02 | 50.23 |
| GTH-h | 18.67 | 16.25 | 8.39 | 11.61 | 19.91 | 18.82 | 31.86 | 18.27 | 21.62 | 48.08 | 19.70 | 47.58 |
| PWCF | 34.57 | 28.95 | 24.22 | 18.42 | 34.03 | 34.44 | 39.78 | 34.86 | 35.12 | 72.91 | 35.01 | 67.94 |
| DAPH | 21.19 | 22.28 | 13.25 | 12.26 | 26.61 | 24.26 | 29.60 | 22.94 | 25.48 | 60.67 | 24.31 | 45.42 |
| PEACE | 45.97 | 42.68 | 38.72 | 28.36 | 53.04 | 54.39 | 46.69 | 48.89 | 46.91 | 83.18 | 46.95 | 78.82 |
| DANCE | 44.53 | 43.54 | 39.03 | 28.87 | 53.73 | 55.14 | 44.78 | 47.66 | 46.68 | 84.75 | 48.61 | 78.39 |
| Ours | **71.18** | **63.28** | **58.65** | **42.84** | **71.27** | **74.77** | **68.37** | **60.61** | **52.84** | **95.88** | **60.14** | **99.90** |
| Imp. | ↑ 25.21 | ↑ 19.74 | ↑ 19.62 | ↑ 13.97 | ↑ 17.54 | ↑ 19.63 | ↑ 21.68 | ↑ 11.72 | ↑ 5.93 | ↑ 11.13 | ↑ 11.53 | ↑ 21.08 |
| Avg. | ↑19.29 | | | | | | ↑13.85 | | | | | |

Table 1: Cross-domain retrieval performance comparison with baselines on Office-Home and Office-31. The best result in each column is marked in bold. The second best result in each column is underlined.

given target query sample, we can employ the trained model to generate its corresponding binary code. This procedure is neatly summarized in the following forward propagation sequence:

$$\mathbf{x}_q^t \overset{input}{\longrightarrow} \mathbf{MLP}\left(\mathbf{x}_q^t\right) \rightarrow \mathbf{HashEncoder}\left(\mathbf{x}_q^t\right) \overset{output}{\longrightarrow} \mathbf{h}_q^t \tag{13}$$

Finally, by employing a straightforward quantization approach, the hash code $\mathbf{b}_q^t$ for the given target query sample $\mathbf{x}_q^t$ can be derived as follows: $\mathbf{b}_q^t = sgn\left(\mathbf{h}_q^t\right)$.

# Experiment

## Experimental Dataset

We conduct experiments on three publicly available datasets to validate the performance of our method, namely Office-Home (Venkateswara et al. 2017b), Office-31 (Saenko et al. 2010), and Digits (Wang et al. 2023a). **Office-Home** is a dataset collected specifically for the evaluation of domain adaptation algorithms. This dataset comprises images of 65 categories found typically in office and home settings, divided into four domains: Artistic images (A), Clip Art (C), Product images (P), and Real-World images (R). Consistent with previous methods, we conduct experiments on six transferable image retrieval tasks, namely: A→R, R→A, C→R, R→C, P→R, and R→P. **Office-31** dataset serves as a vital resource for researching, evaluating, and comparing solutions to the domain shift problem. It collects images from three distinct domains: Amazon (A), DSLR (D), and Webcam (W), each containing images belonging to 31 common office environment categories. By randomly selecting a pair of domains as the source and target domains, we conduct experiments across six transferable image retrieval tasks: A→D, A→W, D→A, D→W, W→A, and W→D. **Digits** consists of two handwritten digit recognition datasets, MNIST (M) (LeCun et al. 1998) and USPS (U) (Hull 1994). They stand as prevalent cross-domain datasets, containing handwritten digits from 0 to 9. In our experiments, the two datasets serve as source and target domains for each other, resulting in two transferable image retrieval tasks: M→U and U→M.

Same as previous methods (Huang et al. 2020; Wang et al. 2023b), we employ the source domain images along with 90% of the target domain images for training purposes. A random subset of 10% of the target domain images is set aside for testing. In the case of cross-domain retrieval, the database is composed of source domain images, whereas in the case of single-domain retrieval, the database is constituted by target domain images. All the above experimental datasets are kindly organized and shared by PCWF.

## Baseline Method and Evaluation Metric

We compare our CPH with several state-of-the-art methods, containing five unsupervised hashing methods and seven transfer hashing methods. The unsupervised methods include LSH (Gionis, Indyk, and Motwani 1999), ITQ (Gong et al. 2013), DSH (Jin et al. 2014), SGH (Jiang and Li 2015) and GraphBit (Wang et al. 2023c). The transfer hashing methods include ITQ+ (Zhou et al. 2018), GTH-g (Liu and Zhang 2019), GTH-h (Liu and Zhang 2019), PWCF (Huang et al. 2020), DAPH (Huang, Zhang, and Gao 2022), PEACE (Wang et al. 2023a) and DANCE (Wang et al. 2023b). The former four are shallow transfer hashing methods while the latter three and ours are deep transfer hashing methods. Due to the lack of privileged knowledge, partial experimental results of baselines are derived from DAPH, PEACE and DANCE.

We use mean Average Precision (mAP), Top-K Precision Curve, and Precision-Recall Curve to measure the quality of obtained hash codes. For all evaluation metrics, higher values indicate better performance. In the experiments, we repeat our method 5 times and report their mAP results.

## Implementation Detail

Our hash learning model features a straightforward network structure. $\mathbf{MLP}(\cdot;\theta_{mlp})$ is comprised of one fully connected layer ($d \rightarrow d$), a batch-normalization layer, and an

| Method | Bit | Office-Home | | | | | | Office-31 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A→R | R→A | C→R | R→C | P→R | R→P | A→D | A→W | D→A | D→W | W→A | W→D |
| GTH-g | 16 | 10.20 | 9.51 | 6.04 | 5.90 | 10.84 | 11.08 | 26.25 | 11.85 | 15.76 | 34.40 | 16.14 | 31.79 |
| | 32 | 13.08 | 13.93 | 7.86 | 9.52 | 15.28 | 16.17 | 28.35 | 15.76 | 21.15 | 41.36 | 19.23 | 42.86 |
| | 128 | 16.51 | 19.52 | 8.53 | 13.92 | 20.81 | 21.24 | 31.68 | 20.55 | 21.93 | 50.09 | 20.17 | 53.54 |
| GTH-h | 16 | 9.54 | 8.18 | 6.17 | 6.30 | 11.32 | 10.81 | 24.86 | 11.94 | 19.02 | 34.15 | 14.66 | 40.58 |
| | 32 | 13.43 | 12.67 | 7.77 | 8.97 | 15.71 | 15.36 | 24.65 | 15.56 | 20.98 | 41.67 | 17.97 | 42.33 |
| | 128 | 13.78 | 19.73 | 8.57 | 14.54 | 21.16 | 20.16 | 32.01 | 22.16 | 22.56 | 51.62 | 21.55 | 51.57 |
| DAPH | 16 | 11.92 | 14.46 | 8.16 | 8.12 | 17.11 | 14.37 | 22.46 | 15.94 | 19.69 | 52.39 | 19.44 | 34.01 |
| | 32 | 17.72 | 19.63 | 10.48 | 10.64 | 22.47 | 20.25 | 25.15 | 19.09 | 21.99 | 54.28 | 22.00 | 36.58 |
| | 128 | 22.27 | 23.78 | 14.32 | 13.39 | 28.25 | 25.34 | 32.90 | 27.49 | 29.11 | 64.25 | 26.58 | 47.59 |
| Ours | 16 | 62.19 | 53.41 | 53.47 | 35.64 | 64.18 | 66.65 | 59.54 | 52.54 | 43.47 | 90.86 | 55.53 | 98.33 |
| | 32 | 67.87 | 60.16 | 56.71 | 39.71 | 68.31 | 71.72 | 64.00 | 56.33 | 49.34 | 95.04 | 59.18 | 99.49 |
| | 128 | 72.17 | 64.62 | 53.25 | 42.71 | 72.89 | 75.50 | 69.20 | 61.18 | 54.89 | 97.54 | 61.49 | 99.99 |

Table 2: Partial performance comparison of cross-domain retrieval on Office-Home and Office-31 with varying bits.

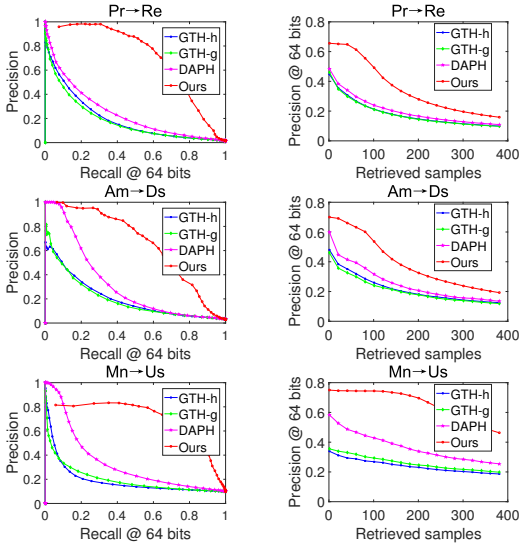


Figure 2: The Precision-Recall Curves and TopK Precision Curves and of GTH-g, GTH-h, DAPH and CPH..

activation function $ReLU(\cdot)$. In $\mathbf{HashEncoder}(\cdot;\theta_{hash})$, two fully connected layers ($d \rightarrow d' \rightarrow r$) are employed. The former one is followed by a batch-normalization layer and the activation function $ReLU(\cdot)$. The latter one is followed by the activation function $Tanh(\cdot)$. For the implementation of our CPH, we have utilized the open-source PyTorch. Adam optimizer (Kingma and Ba 2015) is utilized to train the whole framework by a standard back-propagation strategy. Specifically, the learning rate is set to 0.0001, the batch size is set to 256, and the training epoch is set to 70 on three datasets. The trade-off hyper-parameters within the overall objective function are set as $\{\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 100\}$, $\{\lambda_1 = 0.1, \lambda_2 = 0.01, \lambda_3 = 0.1\}$, and $\{\lambda_1 = 0.01, \lambda_2 = 0.01, \lambda_3 = 0.01\}$ on Office-Home, Office-31 and Digits, respectively. $\eta$ in set to 1.1 and $\gamma$ is set to 0.9.

## Evaluation on Cross-Domain Retrieval

To demonstrate the superiority of our method on cross-domain retrieval, we conduct experiments to compare the retrieval performance of our method and all baseline methods on Office-Home and Office-31 when the hash code length is fixed at 64 bits. The mAP results are presented in Table 1. Based on these experimental findings, we have made the following observations and analyses: Compared to all baseline methods, our CPH achieves significant performance improvements across all transferable tasks on both datasets. Specifically, on Office-Home, our CPH outperforms the second-best results by 25.21%, 19.74%, 19.62%, 13.97%, 17.54%, and 19.63% for different cross-domain tasks, respectively. The average mAP improvement across six cross-domain tasks is 19.29%. On Office-31, our CPH surpasses suboptimal results by 21.68%, 11.72%, 5.97%, 11.13%, 11.52%, and 21.08% for six cross-domain tasks, respectively. The average mAP improvement across the six tasks is 13.85%.

To further validate the effectiveness of our proposed method in cross-domain retrieval, we conduct a comparative analysis with several cross-domain retrieval hashing methods (namely GTH-g, GTH-h, and DAPH) for which we are able to obtain the original source codes. The mAP results for different methods on two datasets, with hash code lengths varying with different hash code lengths (16 bits, 32 bits, and 128 bits), have been shown in Table 2. From these results, it is evident that our CPH approach achieves a significantly superior performance compared to these methods. Furthermore, in Figure 2, we present the Top-K Precision Curves and Precision-Recall Curves of GTH-g, GTH-h, DAPH, and our methods for the three cross-domain tasks across three datasets, specifically using 64-bit hash codes. These curves further underscore the superiority of our CPH method in the realm of cross-domain retrieval. In the case of the Top-K Precision Curves, as the number of retrieved images increases, our method consistently outperforms other baselines across all three cross-domain tasks. The Precision-Recall Curves show that, across all scenarios, the area under the curve of our CPH method surpasses that of the competing approaches.

## Evaluation on Single-Domain Retrieval

To demonstrate the effectiveness of our method in single-domain retrieval, we conduct experiments to compare the

| Task | P→R | | | | A→D | | | | M→U | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 16 | 32 | 64 | 128 | 16 | 32 | 64 | 128 | 16 | 32 | 64 | 128 |
| LSH | 5.84 | 10.62 | 17.57 | 24.92 | 16.04 | 26.18 | 39.68 | 49.04 | 26.69 | 33.49 | 35.64 | 38.53 |
| ITQ | 20.07 | 29.64 | 33.15 | 34.81 | 40.83 | 49.27 | 56.16 | 59.41 | 13.39 | 22.58 | 39.67 | 40.16 |
| DSH | 6.10 | 11.44 | 16.61 | 14.45 | 22.45 | 33.38 | 40.09 | 46.31 | 41.42 | 45.30 | 47.85 | 50.76 |
| SGH | 18.97 | 26.18 | 32.61 | 34.97 | 38.67 | 45.59 | 53.57 | 57.37 | 15.60 | 30.78 | 35.55 | 41.78 |
| GraphBit | 15.42 | 21.80 | 24.89 | 28.97 | 33.21 | 41.17 | 51.46 | 53.48 | 24.96 | 32.54 | 37.54 | 44.82 |
| ITQ+ | 15.60 | 20.60 | 24.96 | 24.05 | 35.03 | 42.62 | 43.12 | 39.12 | 50.22 | 49.66 | 44.38 | 43.21 |
| GTH-g | 15.05 | 21.20 | 27.67 | 28.40 | 37.11 | 45.69 | 50.22 | 55.81 | 45.41 | 39.72 | 34.34 | 34.73 |
| GTH-h | 13.37 | 22.03 | 26.40 | 28.99 | 39.88 | 46.60 | 50.74 | 54.73 | 43.38 | 40.09 | 34.14 | 32.80 |
| DAPH | 20.77 | 29.01 | 33.35 | 34.92 | 46.74 | 49.43 | 58.63 | 60.41 | 47.53 | 54.86 | 60.15 | 60.39 |
| PEACE | 28.99 | 37.93 | 42.97 | 47.29 | 55.43 | 57.89 | 61.21 | 64.14 | 52.77 | 56.25 | 65.27 | 69.99 |
| DANCE | 31.37 | 37.64 | 44.13 | 48.93 | 54.42 | 58.02 | 63.09 | 67.91 | 52.65 | 55.98 | 66.81 | 70.47 |
| Ours | **44.99** | **49.35** | **52.45** | **51.40** | **60.60** | **62.11** | **65.76** | **68.20** | **66.76** | **71.34** | **72.64** | **72.46** |
| Imp. | ↑13.62 | ↑11.42 | ↑8.32 | ↑2.47 | ↑5.17 | ↑4.09 | ↑2.67 | ↑0.29 | ↑12.57 | ↑15.09 | ↑5.83 | ↑1.99 |
| Avg. Imp. | ↑8.96 | | | | ↑3.05 | | | | ↑8.87 | | | |

Table 3: Single-domain retrieval performances comparison with baselines. The best result in each column is marked in bold. The second best result in each column is underlined.

| Variant | Office-Home | | | | | |
|---|---|---|---|---|---|---|
| | A→R | R→A | C→R | R→C | P→R | R→P |
| CPH-v1 | 4.51 | 5.59 | 2.93 | 3.11 | 5.83 | 5.90 |
| CPH-v2 | 67.26 | 61.51 | 20.79 | 31.33 | 70.59 | 70.38 |
| CPH-v3 | 62.94 | 58.45 | 50.78 | 37.97 | 64.20 | 65.14 |
| CPH-v4 | 4.53 | 2.71 | 3.01 | 3.00 | 6.13 | 12.55 |
| CPH-v5 | 35.50 | 31.31 | 27.63 | 22.66 | 40.42 | 42.50 |
| Ours | 71.18 | 63.28 | 58.65 | 42.84 | 71.27 | 74.77 |

Table 4: Ablation study results on Office-Home.

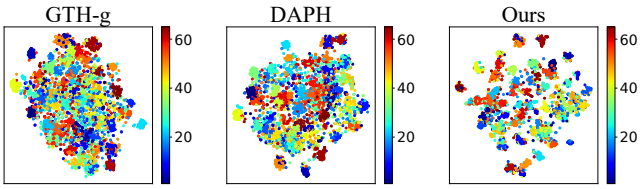


Figure 3: The t-SNE visualizations of 64-bit hash codes on Office-Home.

retrieval performance of our CPH approach with that of all baseline methods. We present the mAP results in Table 3 for three tasks (P→R, A→D, M→U) across Office-Home, Office-31, and Digits datasets, with hash code lengths varying from 16 to 128 bits. From the experimental results, it is evident that similar to cross-domain retrieval, CPH achieves significant improvements in various tasks. The average improvements in the three single-domain retrieval tasks are 8.96%, 3.05%, and 8.87%, respectively.

## Ablation Study

To clearly understand the influence of each components in CPH, we design several variants to further evaluate our model. CPH-v1 denotes the variant approach that does not transfer any source knowledge. CPH-v2 denotes the variant approach that removes the Prototype Contrastive Loss $\mathcal{L}_{PCL}$. CPH-v3 denotes the variant approach that removes the Quantization Loss $\mathcal{L}_{QL}$. CPH-v4 denotes the variant approach that removes the Relation Preservation Loss $\mathcal{L}_{RPL}$. CPH-v5 denotes the variant approach that does not preserve source domain knowledge. The experiments are conducted on Office-Home when the hash code length is fixed as 64 bits, and the corresponding results are reported in Table 4. We can easily observe that transferring knowledge from the source domain makes a significant contribution to improving model performance. Moreover, without the preservation of the semantic relations of images, the performance of model suffers from obvious reduction Thus, all components in our method lead to the improvements of CPH.

## Visualization

Figure 3 gives the t-SNE analyses on Office-Home datasets. The figures show that, the hash codes generated by our CPH show more distinct structures compared to GTH-g and DAPH. This phenomenon underscores the efficacy of the proposed CPH method in generating more discriminative binary codes, thereby enhancing the success of unsupervised domain adaptive image retrieval.

# Conclusion

While admiring the merits of domain adaptation and illuminating the challenges inherent in current domain adaptive hashing methods, we propose a simple yet effective *Comparative Prototype Hashing* (CPH) approach. This method substantially boosts the performance of unsupervised domain adaptive hashing across both cross-domain and single-domain retrieval scenarios. It performs a cohesive collaboration between discriminative feature representation learning, domain alignment, and hash code learning on a domain-shared unit hypersphere space. Comprehensive experimentation conducted on three widely recognized domain adaptation benchmarks validates the superior performance of the proposed CPH method.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62172263, in part by the CCF-Baidu Open Fund under Grant CCF-BAIDU OF2022008, in part by the Key Project of Quancheng Provincial Laboratory of Shandong under Grant QCLZD202303, in part by the Shandong Provincial Natural Science Foundation of China under Grant ZR2022MF295, in part by the Fundamental Research Promotion Plan of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2021JC02020, in part by the Pilot Project for Integrated Innovation of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences) under Grant 2022JBZ01-01, in part by the Research Project of Provincial Laboratory of Shandong, China under Grant SYS202201.

## References

Cui, H.; Zhu, L.; Li, J.; Cheng, Z.; and Zhang, Z. 2021. Two-pronged Strategy: Lightweight Augmented Graph Network Hashing for Scalable Image Retrieval. In *MM*, 1432–1440.

Cui, H.; Zhu, L.; Li, J.; Yang, Y.; and Nie, L. 2020. Scalable Deep Hashing for Large-Scale Social Image Retrieval. *TIP*, 29: 1271–1284.

Du, Z.; Li, J.; Su, H.; Zhu, L.; and Lu, K. 2021. Cross-Domain Gradient Discrepancy Minimization for Unsupervised Domain Adaptation. In *CVPR*, 3937–3946.

Ganin, Y.; and Lempitsky, V. S. 2015. Unsupervised Domain Adaptation by Backpropagation. In *ICML*, 1180–1189.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity Search in High Dimensions via Hashing. In *VLDB*, 518–529.

Gong, Y.; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *TPAMI*, 35(12): 2916–2929.

He, T.; Li, Y.; Gao, L.; Zhang, D.; and Song, J. 2019. One Network for Multi-Domains: Domain Adaptive Hashing with Intersectant Generative Adversarial Networks. In *IJCAI*, 2477–2483.

Huang, F.; Zhang, L.; and Gao, X. 2022. Domain Adaptation Preconceived Hashing for Unconstrained Visual Retrieval. *TNNLS*, 33(10): 5641–5655.

Huang, F.; Zhang, L.; Yang, Y.; and Zhou, X. 2020. Probability Weighted Compact Feature for Domain Adaptive Retrieval. In *CVPR*, 9579–9588.

Huang, Z.; Chen, J.; Zhang, J.; and Shan, H. 2023. Learning Representation for Clustering via Prototype Scattering and Positive Sampling. *TPAMI*, 45(6): 7509–7524.

Hull, J. J. 1994. A Database for Handwritten Text Recognition Research. *TPAMI*, 16(5): 550–554.

Jiang, Q.; and Li, W.-J. 2015. Scalable Graph Hashing with Feature Transformation. In *IJCAI*, 2248–2254.

Jin, Z.; Li, C.; Lin, Y.; and Cai, D. 2014. Density Sensitive Hashing. *TCYB*, 44(8): 1362–1371.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*, 1–15.

Lai, H.; Pan, Y.; Liu, Y.; and Yan, S. 2015. Simultaneous Feature Learning and Hash Coding with Deep Neural Networks. In *CVPR*, 3270–3278.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based Learning Applied to Document Recognition. *PROC*, 86(11): 2278–2324.

Li, J.; Du, Z.; Zhu, L.; Ding, Z.; Lu, K.; and Shen, H. T. 2022. Divergence-Agnostic Unsupervised Domain Adaptation by Adversarial Attacks. *TPAMI*, 44(11): 8196–8211.

Li, W.-J.; Wang, S.; and Kang, W. 2016. Feature Learning based Deep Supervised Hashing with Pairwise Labels. In *IJCAI*, 1711–1717.

Liu, J.; and Zhang, L. 2019. Optimal Projection Guided Transfer Hashing for Image Retrieval. In *AAAI*, 8754–8761.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning Transferable Features with Deep Adaptation Networks. In *ICML*, 97–105.

Long, M.; Zhu, H.; Wang, J.; and Jordan, M. I. 2016. Unsupervised Domain Adaptation with Residual Transfer Networks. In *NeurIPS*, 136–144.

Long, M.; Zhu, H.; Wang, J.; and Jordan, M. I. 2017. Deep Transfer Learning with Joint Adaptation Networks. In *ICML*, 2208–2217.

Lu, X.; Zhu, L.; Cheng, Z.; Li, J.; Nie, X.; and Zhang, H. 2019. Flexible Online Multi-modal Hashing for Large-scale Multimedia Retrieval. In *MM*, 1129–1137.

Luo, X.; Wu, D.; Ma, Z.; Chen, C.; Deng, M.; Ma, J.; Jin, Z.; Huang, J.; and Hua, X. 2021. CIMON: Towards High-quality Hash Codes. In *IJCAI*, 902–908.

Pan, S. J.; and Yang, Q. 2010. A Survey on Transfer Learning. *TKDE*, 22(10): 1345–1359.

Qiu, Z.; Su, Q.; Ou, Z.; Yu, J.; and Chen, C. 2021. Unsupervised Hashing with Contrastive Information Bottleneck. In *IJCAI*, 959–965.

Saenko, K.; Kulis, B.; Fritz, M.; and Darrell, T. 2010. Adapting Visual Category Models to New Domains. In *ECCV*, volume 6314, 213–226.

Shen, F.; Shen, C.; Liu, W.; and Shen, H. T. 2015. Supervised Discrete Hashing. In *CVPR*, 37–45.

Sun, B.; Feng, J.; and Saenko, K. 2016. Return of Frustratingly Easy Domain Adaptation. In *AAAI*, 2058–2065.

Tzeng, E.; Hoffman, J.; Darrell, T.; and Saenko, K. 2015. Simultaneous Deep Transfer Across Domains and Tasks. In *ICCV*, 4068–4076.

Tzeng, E.; Hoffman, J.; Saenko, K.; and Darrell, T. 2017. Adversarial Discriminative Domain Adaptation. In *CVPR*, 2962–2971.

Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017a. Deep Hashing Network for Unsupervised Domain Adaptation. In *CVPR*, 5385–5394.

Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017b. Deep Hashing Network for Unsupervised Domain Adaptation. In *CVPR*, 5385–5394.

Wang, H.; Sun, J.; Luo, X.; Xiang, W.; Zhang, S.; Chen, C.; and Hua, X. 2023a. Toward Effective Domain Adaptive Retrieval. *TIP*, 32: 1285–1299.

Wang, H.; Sun, J.; Wei, X.; Zhang, S.; Chen, C.; Hua, X.; and Luo, X. 2023b. DANCE: Learning A Domain Adaptive Framework for Deep Hashing. In *WWW*, 3319–3330.

Wang, J.; Liu, W.; Kumar, S.; and Chang, S. 2016. Learning to Hash for Indexing Big Data - A Survey. *PIEEE*, 104(1): 34–57.

Wang, J.; Zeng, Z.; Chen, B.; Dai, T.; and Xia, S. 2022. Contrastive Quantization with Code Memory for Unsupervised Image Retrieval. In *AAAI*, 2468–2476.

Wang, J.; Zhang, T.; Song, J.; Sebe, N.; and Shen, H. T. 2018. A Survey on Learning to Hash. *TPAMI*, 40(4): 769–790.

Wang, M.; and Deng, W. 2018. Deep Visual Domain Adaptation: A Survey. *Neurocomputing*, 312: 135–153.

Wang, Z.; Xiao, H.; Duan, Y.; Zhou, J.; and Lu, J. 2023c. Learning Deep Binary Descriptors via Bitwise Interaction Mining. *TPAMI*, 45(2): 1919–1933.

Xia, H.; Jing, T.; Chen, C.; and Ding, Z. 2021. Semi-supervised Domain Adaptive Retrieval via Discriminative Hashing Learning. In *MM*, 3853–3861.

Xu, R.; Liu, P.; Wang, L.; Chen, C.; and Wang, J. 2020. Reliable Weighted Optimal Transport for Unsupervised Domain Adaptation. In *CVPR*, 4393–4402.

Zhang, J.; Li, W.; and Ogunbona, P. 2017. Joint Geometrical and Statistical Alignment for Visual Domain Adaptation. In *CVPR*, 5150–5158.

Zhao, F.; Huang, Y.; Wang, L.; and Tan, T. 2015. Deep Semantic Ranking based Hashing for Multi-label Image Retrieval. In *CVPR*, 1556–1564.

Zhou, J. T.; Zhao, H.; Peng, X.; Fang, M.; Qin, Z.; and Goh, R. S. M. 2018. Transfer Hashing: From Shallow to Deep. *TNNLS*, 29(12): 6191–6201.

Zhu, L.; Lu, X.; Cheng, Z.; Li, J.; and Zhang, H. 2020. Deep Collaborative Multi-View Hashing for Large-Scale Image Search. *TIP*, 29: 4643–4655.

Zhu, L.; Zheng, C.; Guan, W.; Li, J.; Yang, Y.; and Shen, H. T. 2024. Multi-Modal Hashing for Efficient Multimedia Retrieval: A Survey. *TKDE*, 36(1): 239–260.