Deep Uncoupled Discrete Hashing via Similarity Matrix Decomposition

DAYAN WU, Institute of Information Engineering, Chinese Academy of Sciences, China
QI DAI, Microsoft Research Asia, China
BO LI and WEIPING WANG, Institute of Information Engineering, Chinese Academy of Sciences, China

Hashing has been drawing increasing attention in the task of large-scale image retrieval owing to its storage and computation efficiency, especially the recent asymmetric deep hashing methods. These approaches treat the query and database in an asymmetric way and can take full advantage of the whole training data. Though it has achieved state-of-the-art performance, asymmetric deep hashing methods still suffer from the large quantization error and efficiency problem on large-scale datasets due to the tight coupling between the query and database. In this article, we propose a novel asymmetric hashing method, called **D**eep **U**ncoupled **D**iscrete **H**ashing (DUDH), for large-scale approximate nearest neighbor search. Instead of directly preserving the similarity between the query and database, DUDH first exploits a small similarity-transfer image set to transfer the underlying semantic structures from the database to the query and implicitly keep the desired similarity. As a result, the large similarity matrix is decomposed into two relatively small ones and the query is decoupled from the database. Then both database codes and similarity-transfer codes are directly learned during optimization. The quantization error of DUDH only exists in the process of preserving similarity between the query and similarity-transfer set. By uncoupling the query from the database, the training cost of optimizing the CNN model for the query is no longer related to the size of the database. Besides, to further accelerate the training process, we propose to optimize the similarity-transfer codes with a constant-approximation solution. In doing so, the training cost of optimizing similarity-transfer codes can be almost ignored. Extensive experiments on four widely used image retrieval benchmarks demonstrate that DUDH can achieve state-of-the-art retrieval performance with remarkable training cost reduction (30%–50% relative).

CCS Concepts: • **Information systems** → **Image search;**

Additional Key Words and Phrases: Deep hashing, similarity-transfer, large-scale image retrieval

ACM Reference format:
Dayan Wu, Qi Dai, Bo Li, and Weiping Wang. 2023. Deep Uncoupled Discrete Hashing via Similarity Matrix Decomposition. *ACM Trans. Multimedia Comput. Commun. Appl.* 19, 1, Article 22 (January 2023), 22 pages.
https://doi.org/10.1145/3524021

This work was supported by the National Natural Science Foundation of China (No. 62106258 and No. 62006242).
Authors' addresses: D. Wu, B. Li (corresponding author), and W. Wang, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China; emails: {wudayan, libo, wangweiping}@iie.ac.cn; Q. Dai, Microsoft Research Asia, Beijing, China; email: qid@micorsoft.com.

1551-6857/2023/1-ART22 $15.00
https://doi.org/10.1145/3524021

1 INTRODUCTION

Hashing aims to encode raw data into short binary codes while preserving data similarity information in the Hamming space. Due to its high storage and computational efficiency, hashing has been widely used in various computer vision tasks, e.g., large-scale image retrieval [14, 15, 18, 42, 51], video retrieval [25, 34], cross-modal retrieval [19, 23, 54], person re-identification [45, 57], and classification [37]. In this article, we focus on incorporating deep neural networks into the learning of hash codes for large-scale image retrieval. Such approaches [24, 30], also named deep hashing methods, have shown better performance than traditional hashing methods with hand-crafted features like **Locality-Sensitive Hashing (LSH)** [11], **Spectral Hashing (SH)** [43], and Iterative Quantization [12].

To date in the literature, deep hashing methods generally leverage the strong representation capability of the powerful **convolutional neural networks (CNNs)** to capture the underlying semantics of images. With the paradigm of simultaneously learning features and hash functions, these methods can take full advantage of the pre-trained CNN, achieving satisfactory results. A large number of approaches have been proposed recently, including both symmetric hashing [1, 5, 6, 10, 20, 28, 32, 39, 40, 53, 61] and asymmetric hashing [3, 18, 44] methods.

Among plentiful deep hashing methods, deep asymmetric hashing methods [3, 18, 36] have demonstrated superior retrieval performance to the conventional symmetric ones. Such methods exploit different hashing functions for the query and database, which are proven to be more effective in preserving similarity information. The state-of-the-art **Asymmetric Deep Supervised Hashing (ADSH)** [18] and **Deep Anchor Graph Hashing (DAGH)** [3] are two representative approaches. By learning hash function only for queries while directly obtaining hash codes for the database, they can capitalize on the whole training data and thus achieve promising results. Nevertheless, they still suffer from the large quantization error problem on large-scale datasets, though their database codes are directly optimized. ADSH and DAGH adopt the *tanh* function to approximate the *sign* function when learning the hash function for queries, which means the quantization error still exists on the query side. As the query codes are closely bound up with the database codes, the quantization error is directly related to the size of the similarity matrix between the query and database. Apart from the quantization problem, the tight coupling between the database and query also causes an efficiency problem, as the computation cost for optimizing the convolutional neural network is directly related to the size of the database.

The main cause of the quantization and efficiency problem in deep asymmetric hashing methods is the tight coupling between the database and query. In this article, we introduce a novel asymmetric hashing method for learning binary hash codes, named **Deep Uncoupled Discrete Hashing (DUDH)**. The proposed DUDH elaborately designs a similarity-transfer matrix to decouple the query from the database, and thus both quantization error and training cost are no longer related to the size of the database. Specifically, a small similarity-transfer image set is constructed by sampling the database, whose code matrix is dubbed the similarity-transfer matrix. Instead of preserving the similarity between the query and database straightly as in ADSH and DAGH (upper part in Figure 1), DUDH leverages the similarity-transfer set to bridge the gap between them. By keeping its similarity to both the query and the database, the underlying semantic structures are transferred from the database to the query and aligned accordingly, which implicitly preserves the similarity between them (lower part in Figure 1). As such, the large similarity matrix is decomposed into two relatively small ones. As illustrated in Figure 2, both database and similarity-transfer codes are directly learned, and query codes are generated by the CNN model. These three parts are guided by the similarity-transfer hashing loss. More specifically, the similarity between the query and similarity-transfer set is preserved by the query-transfer loss, and that

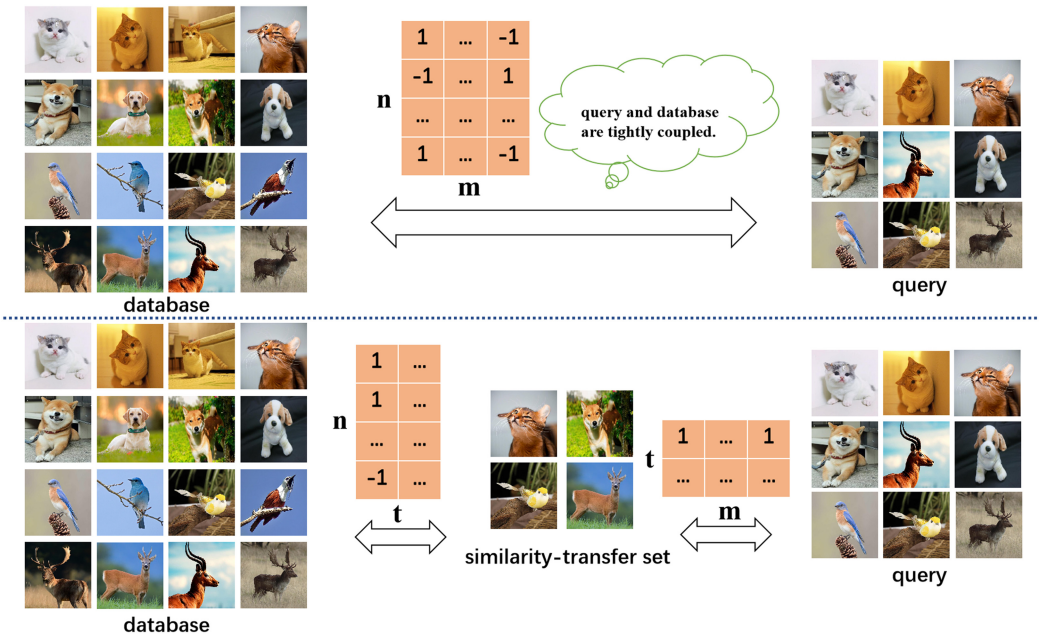


Fig. 1. Schematic of similarity-transfer matrix. The upper part shows that the query and database in conventional deep asymmetric hashing methods are tightly coupled. The bottom part shows that DUDH adopts a similarity-transfer matrix to decouple the query and database while implicitly preserving their similarities. n, t, m denote the size of the database, similarity-transfer set, and query set, respectively ($t < m \ll n$). c denotes the code length.

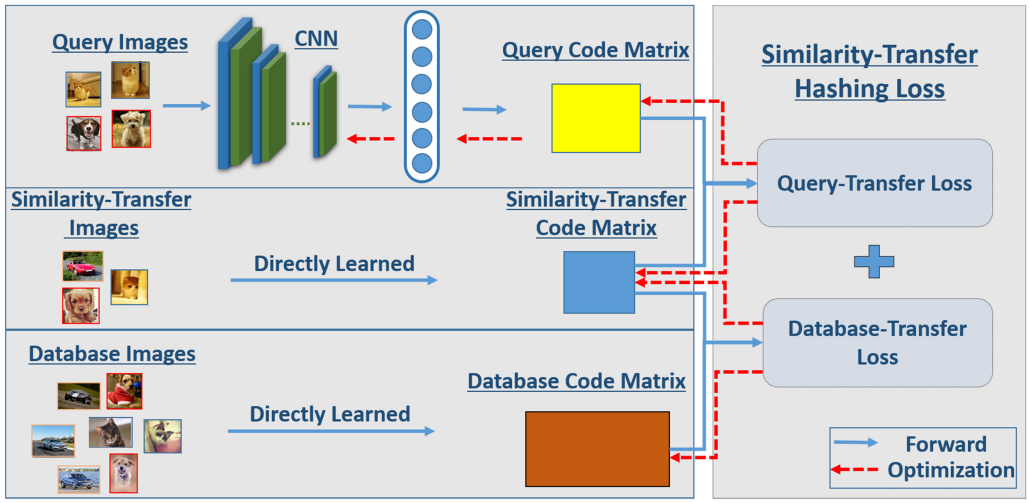


Fig. 2. Framework of DUDH. Similarity-transfer images are randomly sampled from database images to preserve semantic similarity between query and database images. The binary codes for similarity-transfer and database images are directly learned during the optimization, while the ones for query images are generated by the CNN model. Best viewed in color.

between the database and similarity-transfer set is preserved by the database-transfer loss. Note that the quantization error only exists in the query-transfer loss and is no longer related to the size of the database. The main contributions can be summarized as follows:

- A similarity-transfer matrix is elaborately designed to bridge the gap between query and database images. With such matrix, the quantization error and the cost of optimizing the CNN model can be largely reduced and the semantic similarity between the query and database can be preserved as well. Besides, we further propose to accelerate the optimization of the similarity-transfer matrix with a constant-approximation solution. As a result, the cost of optimizing the similarity-transfer matrix can be ignored as well.
- We devise a similarity-transfer hashing loss function by preserving the similarity between training images. It incorporates a similarity-transfer set to train the hash functions for query images. Simultaneously, the binary hash codes for the database and similarity-transfer set can be directly obtained during the optimization.
- Extensive experiments demonstrate that the proposed DUDH can significantly decrease the training time while achieving state-of-the-art retrieval accuracy.

The rest of this article is organized as follows: Related works are briefly discussed in Section 2. Section 3 describes DUDH in detail. Section 4 extensively evaluates the proposed method on three widely used image retrieval datasets. Finally, Section 5 concludes this article.

2 RELATED WORKS

Hashing is an important technique for fast approximate similarity search. Generally speaking, hashing methods can be divided into two categories: data-independent methods and data-dependent methods. Data-independent methods randomly generate a set of hash functions without any training. Representative data-independent methods include LSH [11] and its variants [35]. However, it has been proven that the LSH method needs long codes to meet the accuracy requirement. To generate more compact binary codes, some data-dependent methods are proposed. They try to learn appropriate hash functions that can well separate the training samples. Existing data-dependent hashing methods can be further classified into supervised hashing and unsupervised hashing. Compared with unsupervised hashing methods [12, 43], supervised hashing methods can leverage the label information to achieve better retrieval performance. Representative supervised hashing methods include KSH [31], LFH [56], SDH [38], and FDUDH [27].

With the rapid development of deep learning techniques, the deep models have been applied to hashing approaches, raising the deep hashing methods. In this article, we mainly focus on improving the training efficiency of deep hashing methods. These methods can be summarized into symmetric deep hashing methods and asymmetric ones. The former methods encode database and query images into binary codes through the same CNN model, while the latter methods learn two different deep hash functions for query and database images separately.

Symmetric deep hashing methods have shown great advantages over the hand-crafted feature-based hashing methods. **Convolutional Neural Network Hashing (CNNH)** [46] tries to fit the binary codes computed from the similarity matrix. **Deep Supervised Hashing (DSH)** [30] and **Deep Pairwise Supervised Hashing (DPSH)** [26] are pair-wise label based hashing methods. **Two-stream DH (TSDH)** [8] constructs a two-stream ConvNet architecture and learns hash codes with class-specific centers to minimize the intra-class variation. **Deep Supervised Discrete Hashing (DSDH)** [24] is the first deep hashing method with discrete optimization. Some methods further consider more complicated semantic similarity between images [58]. Other methods [17, 19, 47, 50] focus on learning unified hash codes for different modalities, such as image, text, and video. The aforementioned deep hashing methods are all supervised methods; i.e., they

cannot apply to the scenario where label information is unavailable or incomplete. Recently, some deep unsupervised hashing [7, 9, 28, 39, 49, 51, 53] methods were proposed. **Deep binary descriptors (DeepBit)** [28] treats original images and their corresponding rotated images as similar pairs and attempts to preserve such similarities. DistillHash [51] learns a distilled dataset composed of data pairs that have confident similarity signals. Apart from unsupervised and supervised deep hashing methods, semi-supervised deep hashing [48, 55] methods also have attracted much attention. SSDH [55] tries to utilize the unlabeled images through online graph construction. BGDH [48] constructs a bipartite graph to discover the underlying structure of data, based on which an embedding is generated for each instance.

Asymmetric deep hashing methods recently showed better retrieval performance than symmetric methods. **Deep Asymmetric Pairwise Hashing (DAPH)** [36] adopts two different CNN models to generate hash codes for query and database images separately. ADSH [18] learns a CNN model only for query images and directly learns the binary hash codes for database images. DAGH [3] is similar to ADSH, and it adopts an anchor graph to benefit the learning of binary codes. ADSH and DAGH perform better than DAPH in most cases since ADSH and DAGH can efficiently utilize the supervision. However, ADSH and DAGH still require a large amount of computation during the optimization.

Several hashing works have exploited the "anchors" for generating feature representation (e.g., SDH [38], KSH [31], and GCNH [59]), which shares the form with the proposed similarity-transfer set that both of them are a subset of the database. However, our similarity-transfer set is fundamentally different from them. Anchor-based methods generally employ anchors to learn features of other points by calculating the distances between the points and anchors. Such process does not aim to reduce the computation or the quantization error but is more like the feature enhancement. In contrast, the similarity-transfer set in DUDH is treated as a springboard to transfer the underlying semantic structures from the database to the query, which reduces both computation cost and quantization error. Meanwhile, the similarities between the query and database are also implicitly preserved.

Apart from learning a representation where the intra-class distances are minimized and inter-class distances are maximized, hashing methods need to pay more attention to reducing or avoiding quantization loss during optimization, which is the major difference between deep hashing methods and deep metric learning methods.

3 DEEP UNCOUPLED DISCRETE HASHING

3.1 Problem Definition

Suppose we have m query images denoted as $X = \{x_i\}_{i=1}^{m}$, n database images denoted as $Y = \{y_j\}_{j=1}^{n}$, and t similarity-transfer images denoted as $Z = \{z_k\}_{k=1}^{t}$. The similarity-transfer image set is simply sampled from the database. Normally, the number of query images is much smaller than that of database images but much larger than that of similarity-transfer images; i.e., $t < m \ll n$. $\widehat{S} \in \{-1, +1\}^{m \times t}$ denotes the similarity matrix between the query and similarity-transfer set, and $\widetilde{S} \in \{-1, +1\}^{n \times t}$ denotes the similarity matrix between the database and similarity-transfer set. $\widehat{S}_{ij} = 1/-1$ indicates that x_i and z_j are similar/dissimilar, and $\widetilde{S}_{ij} = 1/-1$ indicates that y_i and z_j are similar/dissimilar. The goal of DUDH is to learn a hash function $h(x_q) \in \{-1, +1\}^c$ to generate the binary hash codes $U = \{u_i\}_{i=1}^{m} \in \{-1, +1\}^{m \times c}$ for query images and directly learn the hash codes $V = \{v_i\}_{i=1}^{n} \in \{-1, +1\}^{n \times c}$, $W = \{w_i\}_{i=1}^{t} \in \{-1, +1\}^{t \times c}$ for the database and similarity-transfer set, respectively, where c is the code length. The Hamming distance between u_i/v_i and w_j should be as small as possible if $\widehat{S}_{ij}/\widetilde{S}_{ij} = 1$; otherwise, the distance should be as large as possible. The notations and their descriptions are listed in Table 1.

Table 1. Notations and Their Descriptions

Notation	Description
m	the number of query images
n	the number of database images
t	the number of similarity-transfer images
c	the length of binary hash codes
X	$X = \{x_i\}_{i=1}^{m}$: the set of m query images
Y	$Y = \{y_j\}_{j=1}^{n}$: the set of n database images
Z	$Z = \{z_k\}_{k=1}^{t}$: the set of t similarity-transfer images
$\widehat{S}$	$\widehat{S} \in \{-1, +1\}^{m \times t}$: the similarity matrix between query and similarity-transfer set
$\widetilde{S}$	$\widetilde{S} \in \{-1, +1\}^{n \times t}$: the similarity matrix between database and similarity-transfer set
h	$h(x_q) \in \{-1, +1\}^{c}$: the deep hash function
U	$U = \{u_i\}_{i=1}^{m} \in \{-1, +1\}^{m \times c}$: the binary hash codes of query images
V	$V = \{v_i\}_{i=1}^{n} \in \{-1, +1\}^{n \times c}$: the binary hash codes of database images
W	$W = \{w_i\}_{i=1}^{t} \in \{-1, +1\}^{t \times c}$: the binary hash codes of similarity-transfer images

3.2 Framework Overview

As illustrated in Figure 2, DUDH has three inputs, i.e., the query images, the similarity-transfer images, and the database images, respectively. In practice, the query and similarity-transfer set are generally unavailable. Therefore, we sample two subsets of Y to form X and Z. The supervision $\widehat{S}$ and $\widetilde{S}$ can be easily obtained given the label of the whole dataset. DUDH contains three core parts: *hash function learning*, *database code learning*, and *similarity-transfer code learning*. In the hash function learning part, a deep CNN model is employed to extract representative features for query images. Deep hash functions are then learned on top of the features to produce the hash codes U. Note that this module is only applied to the query set. The database code learning part directly learns the hash codes V for the database, while the similarity-transfer code learning part learns the codes W for the similarity-transfer set. To achieve the above goals, a similarity-transfer hashing loss function is devised with U, V, W, which implicitly preserves the similarity between the query and database through the similarity-transfer set. We detail each part in the following sections.

3.3 Deep Hash Functions

We construct the hash functions through a CNN model, the output of which is a c-dimensional vector. c is the code length. Accordingly, our deep hash function is defined as

$$\boldsymbol{u_i} = h\left(\boldsymbol{x_i}; \boldsymbol{\theta}\right) = sign\left(f\left(\boldsymbol{x_i}; \boldsymbol{\theta}\right)\right), \tag{1}$$

where θ denotes the parameters of CNN model, and $f(\cdot)$ denotes the output of the CNN model.

3.4 Similarity-Transfer Hashing Loss

DUDH utilizes the similarity-transfer set as a springboard to boost the computation efficiency. It aims to preserve the similarity of the similarity-transfer set with both the query and the database,

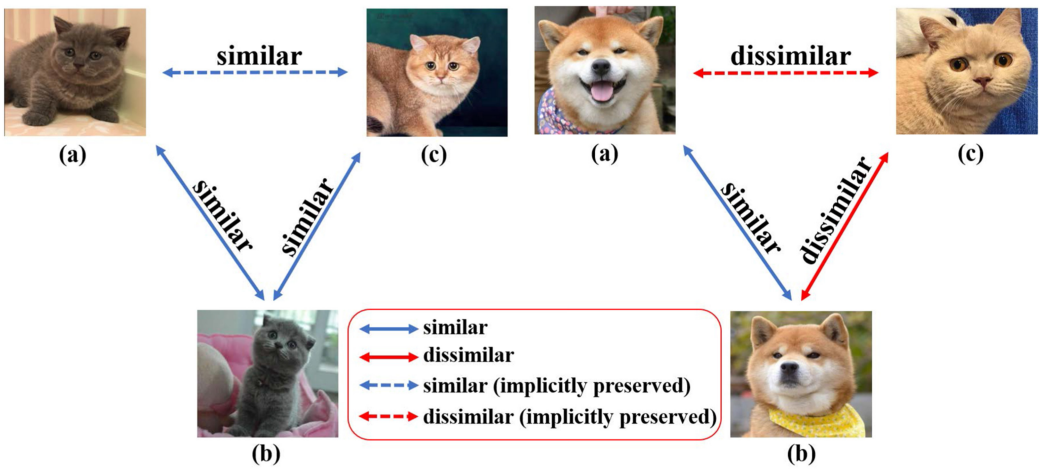


Fig. 3. Examples of how similarity-transfer images work. The left part shows that if the images (a) and (c) are both similar to (b), then the similarity between (a) and (c) can be implicitly preserved. The right part shows that if the image (b) is similar to (a) but dissimilar to (c), then the dissimilarity between (a) and (c) can be implicitly preserved. Best viewed in color.

rather than preserve the similarity between the query and database directly. As such, the underlying semantic structures are transferred from the database to the query and aligned accordingly, which implicitly preserves the similarity between them. In this way, the necessary large similarity matrix between the database and query is "decomposed" into two small ones, thus evading the huge computation overhead. Figure 3 illustrates how similarity-transfer images work. When the images (a) and (c) are both similar to (b) (the similarity-transfer image), the similarity between (a) and (c) can be implicitly preserved (left in the figure). In the same way, the dissimilarity between (a) and (c) can also be preserved through a specific image (b) (right in the figure). The proposed similarity-transfer hashing loss thus follows the above paradigm to separately optimize the hash codes, which consists of two components: *database-transfer loss* and *query-transfer loss*.

Database-transfer loss tries to preserve the similarity between database and similarity-transfer images. With the similarity matrix $\widetilde{S}$, the goal is to reduce/enlarge the Hamming distances between the hash codes of similar/dissimilar pairs. The L_2-norm loss is then adopted to minimize the difference between the supervision and the inner product of binary code pairs, which is given by

$$\begin{aligned} &\min_{V,W} J(V,W) = \sum_{i=1}^{n}\sum_{j=1}^{t}(v_i w_j^T - c\widetilde{S}_{ij})^2, \\ &s.t.\ V = \{v_1, v_2, \ldots, v_n\} \in \{-1,+1\}^{n\times c}, \\ &\qquad W = \{w_1, w_2, \ldots, w_t\} \in \{-1,+1\}^{t\times c}. \end{aligned} \tag{2}$$

Query-transfer loss aims at preserving the similarity between query and similarity-transfer images. Similar to the above database-transfer loss, the query-transfer loss can be defined as

$$\begin{aligned} &\min_{U,W} J(U,W) = \sum_{i=1}^{m}\sum_{j=1}^{t}(u_i w_j^T - c\widehat{S}_{ij})^2, \\ &s.t.\ U = \{u_1, u_2, \ldots, u_m\} \in \{-1,+1\}^{m\times c}, \\ &\qquad W = \{w_1, w_2, \ldots, w_t\} \in \{-1,+1\}^{t\times c}. \end{aligned} \tag{3}$$

Note that u_i is obtained through the CNN model. We integrate Equation (1) into Equation (3) and reformulate the loss function as follows:

$$\begin{aligned} &\min_{\theta, W} J(\theta, W) = \sum_{i=1}^{m} \sum_{j=1}^{t} [sign\left(f(x_i; \theta)\right) w_j^T - c\widehat{S}_{ij}]^2, \\ &s.t. \qquad W = \{w_1, w_2, \ldots, w_t\} \in \{-1, +1\}^{t \times c}. \end{aligned} \tag{4}$$

It should be noted that Equation (4) is in general NP-hard since the function $sign\,(\cdot)$ is not differentiable at 0. We adopt a common continuous relaxation to replace the $sign\,(\cdot)$ with $tanh\,(\cdot)$ function, which brings new formulations as

$$\begin{aligned} &\min_{\theta, W} J(\theta, W) = \sum_{i=1}^{m} \sum_{j=1}^{t} [tanh\left(f(x_i; \theta)\right) w_j^T - c\widehat{S}_{ij}]^2, \\ &s.t. \qquad W = \{w_1, w_2, \ldots, w_t\} \in \{-1, +1\}^{t \times c}. \end{aligned} \tag{5}$$

As aforementioned, both query and similarity-transfer set images are sampled from the database. Let $\Psi = \{1, 2, 3, \ldots, n\}$ denote the indices of database images, $\Omega = \{1, 2, 3, \ldots, m\}$ the indices of query images, and $\Phi = \{1, 2, 3, \ldots, t\}$ the indices of similarity-transfer images. Combining the above *database-transfer loss* and *query-transfer loss*, we can finally formulate **similarity-transfer hashing loss** as

$$\begin{aligned} \min_{\theta, W, V} J(\theta, W, V) &= \sum_{i \in \Psi} \sum_{j \in \Phi} [v_i w_j^T - c\widetilde{S}_{ij}]^2 \\ &+ \lambda \sum_{i \in \Omega} \sum_{j \in \Phi} [tanh\left(f(x_i; \theta)\right) w_j^T - c\widehat{S}_{ij}]^2 \\ &+ \gamma \sum_{i \in \Omega} [v_i - tanh\left(f(x_i; \theta)\right)]^2, \\ s.t.\ V = \{v_1, v_2, \ldots, v_n\} &\in \{-1, +1\}^{n \times c}, \\ W = \{w_1, w_2, \ldots, w_t\} &\in \{-1, +1\}^{t \times c}, \end{aligned} \tag{6}$$

where λ, γ are two hyper-parameters. It is worth noting that we further add a constraint $\sum_{i \in \Omega} [v_i - tanh\left(f(x_i; \theta)\right)]^2$. This is because each image x_i in the query set possesses two kinds of code representations: one is the learned binary hash codes v_i from the database, and the other is the representation $tanh\left(f(x_i; \theta)\right)$ from the CNN model. By minimizing the difference between them, we hope the two representations can be consistent.

3.5 Optimization

We design an alternating optimization strategy to learn the parameters θ, V, and W in Equation (6). More specifically, in each iteration we learn one parameter with the other two fixed. The three steps are repeated for several iterations.

3.5.1 θ-step. When V and W are fixed, we use the standard back-propagation algorithm to optimize θ. For simplicity, we denote $p_i = tanh(f(x_i; \theta))$ and $q_i = f(x_i; \theta)$. The partial derivative of $J(\theta, V, W)$ with respect to q_i is

$$\begin{aligned} \frac{\partial J}{\partial q_i} = 2\Big(&\lambda \sum_{j \in \Phi} [(p_i w_j^T - c\widehat{S}_{ij}) w_j] \\ &+ \gamma (p_i - v_i)\Big) \cdot (\mathbf{1} - p_i \cdot p_i), \end{aligned} \tag{7}$$

where **1** denotes the all-ones vector, and $(\cdot)$ means the element-wise multiplication operation between two vectors. Then we can use the chain rule to update θ.

3.5.2 V-step. When θ is fixed, it is not straightforward to update V. We first rewrite Equation (6) into the following matrix form:

$$\begin{aligned}\min_{V} J(V) &= ||VW^T - c\widetilde{S}||_F^2 + \gamma\, ||V_\Omega - P||_F^2 \\ &= ||VW^T||_F^2 - 2ctr(VW^T\widetilde{S}^T) \\ &\quad - 2\gamma tr(V_\Omega P^T) + const, \\ s.t.\ V &\in \{-1,+1\}^{n\times c},\end{aligned} \tag{8}$$

where $P = \{p_i | i \in \Omega\} \in [-1,+1]^{m\times c}$ and V_Ω is the binary hash codes of the database images indexed by Ω, i.e., $V_\Omega = \{v_i | i \in \Omega\} \in \{-1,+1\}^{m\times c}$. "const" is a constant independent of V. To simplify Equation (8), we newly define $\widetilde{P} = \{\widetilde{p}_i | i \in \Psi\} \in [-1,+1]^{n\times c}$, where $\widetilde{p}_i$ is defined as

$$\widetilde{p}_i = \begin{cases} p_i, & i \in \Omega, \\ 0, & otherwise, \end{cases} \tag{9}$$

and Equation (8) can be rewritten as follows:

$$\begin{aligned}\min_{V} J(V) &= ||VW^T||_F^2 - 2tr(V(cW^T\widetilde{S}^T + \gamma\widetilde{P}^T)), \\ &= ||VW^T||_F^2 + tr(VQ^T), \\ s.t.\ V &\in \{-1,+1\}^{n\times c},\end{aligned} \tag{10}$$

where $Q = -2c\widetilde{S}W - 2\gamma\widetilde{P}$. Note that the constant term in Equation (8) is omitted for simplicity. Then we adopt the **discrete cyclic coordinate descent (DCC)** algorithm proposed in [38] to solve Equation (10). Specifically, we update V bit by bit, which means each time we update one column of V with other columns fixed. We denote V_{*l} as the lth column of V ($l = 1,\ldots,c$) and $\hat{V}_l$ as the matrix of V excluding V_{*l}. For W, let W_{*l} be the lth column of W and $\hat{W}_l$ be the matrix of W excluding W_{*l}. Similarly, let Q_{*l} denote the lth column of Q and $\hat{Q}_l$ denote the matrix of Q excluding Q_{*l}. Therefore, Equation (10) can be transformed to

$$\begin{aligned}\min_{V_{*l}} J(V_{*l}) &= ||VW^T||_F^2 + tr(VQ^T), \\ &= tr(V_{*l}[2W_{*l}^T\hat{W}_l\hat{V}_l^T + \hat{Q}_l^T]), \\ s.t.\ V_{*l} &\in \{-1,+1\}^n.\end{aligned} \tag{11}$$

Obviously, when the sign of each bit in V_{*l} is different from that of the corresponding bit in $2W_{*l}^T\hat{W}_l\hat{V}_l^T + \hat{Q}_l^T$, $J(V_{*l})$ can reach its minimum value. Therefore, the solution to Equation (11) is as follows:

$$V_{*l} = -sign(2\hat{V}_l\hat{W}_l^T W_{*l} + \hat{Q}_l). \tag{12}$$

We then update V by replacing the lth column of V with V_{*l}. Accordingly, all columns of V are updated sequentially by repeating Equation (12).

3.5.3 W-step. Given the fixed θ and V, the objective in Equation (6) can be rewritten into the formulation of

$$\begin{aligned}\min_{W} J_2(W) &= ||VW^T - c\widetilde{S}||_F^2 + \lambda||PW^T - c\widehat{S}||_F^2, \\ s.t. \quad & W \in \{-1,+1\}^{t\times c}.\end{aligned} \tag{13}$$

ALGORITHM 1: The learning algorithm for DUDH

Input: database set Y; code length c; iteration number T_i; epoch number T_e.
Output: database hash codes V, similarity-transfer hash codes W, and neural network parameter θ.
Initialize $V \in \{-1, +1\}^{n \times c}$ and neural network parameter θ.
for $i = 1 \rightarrow T_i$ **do**
  Randomly sample t images from Y and initialize $W = V_{\Phi} \in \{-1, +1\}^{t \times c}$.
  Randomly sample m images from Y to form query set X.
  for $j = 1 \rightarrow T_e$ **do**
    1. Forward computation to compute $f(x; \theta)$ in mini-batch;
    2. Compute derivation according to Equation (7);
    3. Update the neural network θ by utilizing back propagation.
  end
  Update W according to Equation (15).
  Update V according to Equation (12).
end

Different from θ and V, we optimize W in a discrete way, where the error of the solution is proven to be bounded. Let's first consider the following problem, which changes Equation (13) from Frobenius norm to L_1-norm:

$$\begin{aligned} \min_{W} J_1(W) &= ||VW^T - c\widetilde{S} + \lambda(PW^T - c\widehat{S})||_1 \\ &= ||(V + \lambda\overline{P})W^T - c(\widetilde{S} + \lambda\overline{\widehat{S}})||_1, \\ s.t. \quad & W \in \{-1, +1\}^{t \times c}, \end{aligned} \tag{14}$$

where $\overline{P} \in [-1, +1]^{n \times c}$ and $\overline{\widehat{S}} \in \{-1, +1\}^{n \times t}$. The first m rows of $\overline{P}$ are the same as P, and the remaining elements are all zeros. Similarly, elements in the first m rows of $\overline{\widehat{S}}$ are the same as $\widehat{S}$, and the rest are all zeros.

The solution of Equation (14) can be simply found as

$$W = sign((\widetilde{S} + \lambda\overline{\widehat{S}})^T(V + \lambda\overline{P})). \tag{15}$$

Furthermore, we have the following theorem.

Theorem 3.1. *Suppose that $J_1(W)$ and $J_2(W)$ reach their minimum at the points W_1^* and W_2^*, respectively. We have $J_2(W_1^*) \leq 2cJ_2(W_2^*) + (\frac{1}{2}c + 8c^2)\lambda mt$.*

The proof of Theorem 3.1 can be found in the appendix. Note that the parameters c, λ, m, t are all constants and usually small. In other words, when we utilize Equation (15) to solve Equation (13), the solution is a constant-approximation solution, which provides an error bound for Equation (13). Finally, the entire learning algorithm of DUDH is summarized in Algorithm 1.

3.6 Analysis on Quantization Error and Computational Complexity

The quantization error of DUDH only exists in the query-transfer loss, and its value is no longer related to the size of the database. For ADSH, as the query is tightly coupled with the database, its quantization error is proportional to the size of the database. Compared with ADSH, the quantization error of DUDH can be significantly reduced. The computational cost for training DUDH comprises three parts: one for optimizing θ, one for optimizing W, and one for optimizing V. Specifically, the training complexity is $O(T_i T_e mtc)$ for optimizing θ, $O(T_i ntc)$ for W, and $O(T_i ntc^2)$ for V. Here T_i is the iteration number, and T_e is the network epoch number in each iteration. On the

Table 2. Comparison of Computational Complexity

Methods	θ	$\mathbf{W}$	$\mathbf{V}$
ADSH	$O(T_i T_e mnc)$	-	$O(T_i nmc^2)$
DUDH	$O(T_i T_e mtc)$	$O(T_i ntc)$	$O(T_i ntc^2)$

other hand, the complexity of ADSH is $O(T_i T_e mnc)$ for optimizing θ, and $O(T_i mnc^2)$ for V. Generally, optimizing θ and V costs much more time than optimizing W. In comparison with ADSH, the complexity of DUDH for optimizing θ is much lower because $t \ll n$, which indicates that the network optimization cost is simply proportional to the size of the similarity-transfer set. Besides, the complexity of DUDH for optimizing V is also lower because $t < m$. Therefore, though additional optimization for W is exhibited, DUDH is still much more efficient than ADSH. The comparison of computational complexity is listed in Table 2.

4 EXPERIMENTS

4.1 Evaluation Setup

4.1.1 Datasets. We conduct extensive evaluations of our proposed method DUDH on four widely used datasets, **CIFAR-10** [22], **SVHN** [33], **NUS-WIDE** [4], and **MS-COCO** [29].

- **CIFAR-10**[1] consists of 60,000 single-label color images in 10 classes. Each class contains 6,000 images of size 32×32. For CIFAR-10, two images will be treated as a similar pair if they share one common label.
- **SVHN**[2] consists of 73,257 training images and 26,032 testing images. The images are categorized into 10 classes, each corresponding to a digital number. For SVHN, two images are similar if they share the same label.
- **NUS-WIDE**[3] contains 269,648 multi-label web images collected from Flickr. The association between images and 81 concepts is manually annotated. Following [30, 60], we only select the images associated with the 21 most frequent concepts (labels), where each of these concepts (labels) associates with at least 5,000 images, resulting in a total of 195,834 images. For NUS-WIDE, two images will be defined as a similar pair if they share at least one common label.
- **MS-COCO**[4] is a multi-label dataset. It contains 82,783 training images and 40,504 validation images, which belong to 80 classes. Two images are similar if they share at least one common label.

For CIFAR-10, we randomly select 1,000 images (100 images per class) as the query set, with the remaining images as database images. For SVHN, we randomly select 1,000 images (100 images per class) from the testing set as the query set and utilize the whole training set as the retrieval set. Similarly, for NUS-WIDE, we randomly choose 2,100 images (also 100 images per class) as the query set, leaving the rest as the database. For MS-COCO, we randomly sample 5,000 images as the query set, and the rest of the images are used as the training and gallery images.

4.1.2 Comparison Methods. We compare DUDH with both traditional and deep hashing methods. For traditional approaches, we compare with LSH [11] and ITQ [12] from unsupervised hashing, and FASTH [27], LFH [56], SDH [38], and COSDISH [21] from supervised hashing. The deep hashing methods include DSH [30], DHN [60], DIHN [44], ADSH [18], JLDSH [13], and CSQ [52].

[1] http://www.cs.toronto.edu/~kriz/cifar.html.
[2] http://ufldl.stanford.edu/housenumbers/.
[3] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm.
[4] https://cocodataset.org/.

Table 3. Comparison of MAP w.r.t. Different Number of Bits on Three Datasets

Methods	CIFAR-10				NUS-WIDE				SVHN			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
LSH [11]	0.147	0.173	0.180	0.193	0.341	0.351	0.351	0.371	0.107	0.108	0.109	0.111
ITQ [12]	0.258	0.273	0.283	0.294	0.505	0.504	0.503	0.505	0.111	0.114	0.115	0.116
FDUDH [27]	0.620	0.673	0.687	0.716	0.741	0.783	0.795	0.809	0.251	0.296	0.318	0.344
LFH [56]	0.401	0.605	0.657	0.700	0.705	0.759	0.778	0.794	0.193	0.256	0.284	0.325
SDH [38]	0.520	0.646	0.658	0.669	0.739	0.762	0.770	0.772	0.151	0.300	0.320	0.334
COSDISH [21]	0.609	0.683	0.696	0.716	0.730	0.764	0.787	0.800	0.238	0.295	0.320	0.341
DSH [30]	0.646	0.749	0.786	0.811	0.762	0.794	0.797	0.808	0.370	0.480	0.523	0.583
DHN [60]	0.673	0.711	0.705	0.714	0.790	0.810	0.809	0.818	0.380	0.410	0.416	0.430
ADSH [18]	0.890	0.928	0.931	0.939	0.840	0.878	0.895	0.906	0.797	0.890	0.912	0.919
DIHN [44]	0.892	0.927	0.933	0.940	0.835	0.882	0.900	0.902	0.790	0.887	0.913	0.915
CSQ [52]	0.937	0.941	**0.952**	0.947	0.840	0.886	0.887	0.886	**0.903**	0.919	0.927	0.935
JLDSH [13]	0.877	0.933	0.933	0.942	0.840	0.888	0.900	**0.910**	0.796	0.880	0.895	0.886
DUDH (*Ours*)	**0.938**	**0.942**	0.943	**0.948**	**0.863**	**0.894**	**0.900**	0.906	0.895	**0.922**	**0.929**	**0.941**

The best results for MAP are shown in bold.

Note that JLDSH, DIHN, and ADSH are asymmetric deep hashing methods. JLDSH and DIHN are inherited from ADSH.

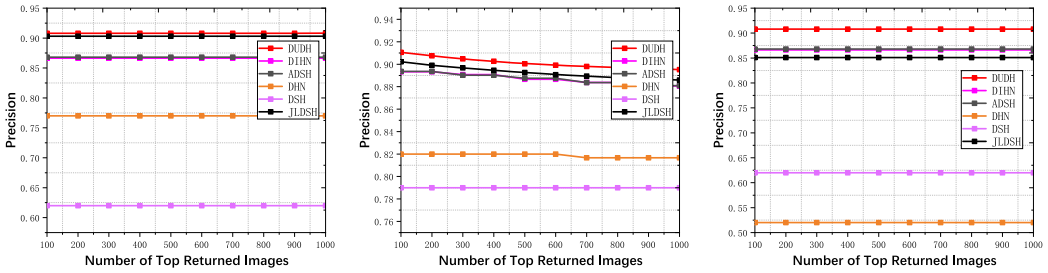
4.1.3 Implementation Details. Following [16, 18], we adopt the CNN-F model [2] as the basic network architecture for both DUDH and all the other deep hashing approaches. This CNN architecture has five convolutional layers and two FC layers. For traditional (non-deep) methods, we utilize the 4,096-dim deep features extracted from the CNN-F model pre-trained on ImageNet. In addition, we set $m = 2{,}000$ for ADSH, JLDSH, and DUDH by cross-validation. In view of both computation cost and retrieval accuracy, we set $t = 100$ for CIFAR-10/SVHN and $t = 1{,}000$ for NUS-WIDE/MS-COCO. For DIHN, we randomly select three classes as incremental classes, and we follow [44] that adopts ADSH to train the base model. Note that ADSH, DIHN, JLDSH, and DUDH are all trained on the whole databases for a fair comparison.

4.1.4 Evaluation Metrics. We report the **Mean Average Precision (MAP)** to evaluate the overall retrieval performance of DUDH and baselines. MAP is widely used in image retrieval evaluation, which includes the mean of the average precision values obtained from the top returned samples. Following [18, 30, 44], the MAP results for NUS-WIDE are calculated based on the top-5k returned samples. Additionally, we evaluate the performance from other aspects, including the top-k precision and precision-recall curve. Note that DUDH also aims to decrease the training cost. Therefore, we also compare the training time of several deep hashing methods, including the detailed training time for optimizing θ, W, and V.

4.1.5 Experimental Environment. All of our proposed approaches are implemented on the MatConvNet [41] framework. We carry out the experiments on NVIDIA RTX 2080Ti.

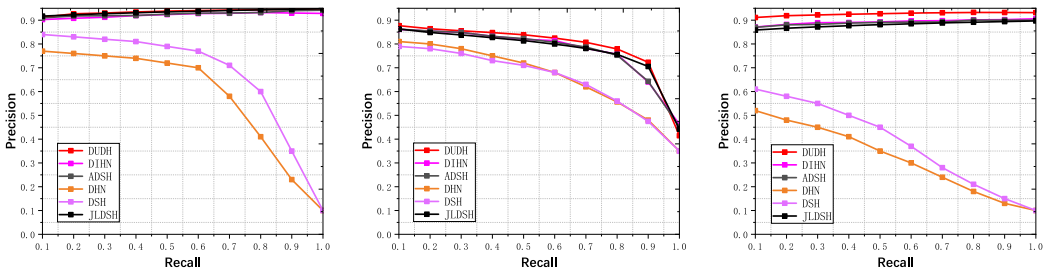
4.2 Accuracy Comparison

4.2.1 Comparison of Retrieval Accuracy. Table 3 shows the MAP performance comparisons on three datasets. As shown in the table, the most competitive methods are JLDSH, ADSH, and DIHN. Among these three methods, ADSH-based methods (ADSH, DIHN, and JLDSH) are more competitive. DUDH performs the best in most cases. We also report the performance of the top-k precision and precision-recall curve with 24 bits on three datasets. As shown in Figure 4, DUDH achieves the best performance in all cases on CIFAR-10 and SVHN, which is consistent with the MAP results. For NUS-WIDE, DUDH achieves the best top-k performance for small k. When k gets larger, DUDH



(a) CIFAR-10 (b) NUS-WIDE (c) SVHN

Fig. 4. Performance of top-k precision on three datasets. The code length is 24. Best viewed in color.



(a) CIFAR-10 (b) NUS-WIDE (c) SVHN

Fig. 5. Performance of precision-recall curve on three datasets. The code length is 24. Best viewed in color.

is a little inferior to JLDSH. For the precision-recall curve, as shown in Figure 5, it is interesting to observe that the precisions of DUDH, JLDSH, ADSH, and DIHN increase with the increasing of recalls on CIFAR-10 and SVHN, which is different from the normal precision-recall curves. This indicates that the three methods may only fail on some extremely hard negative examples. Similar to the top-k and MAP performance, DUDH can achieve the best precision-recall performance on CIFAR-10 and SVHN, while being a little inferior to ADSH and JLDSH in certain cases on NUS-WIDE. In general, the proposed DUDH can achieve the best accuracy performance in all cases on CIFAR-10 and SVHN and competitive accuracy performance on NUS-WIDE. Meanwhile, the training cost of our DUDH can be largely reduced, which will be discussed later.

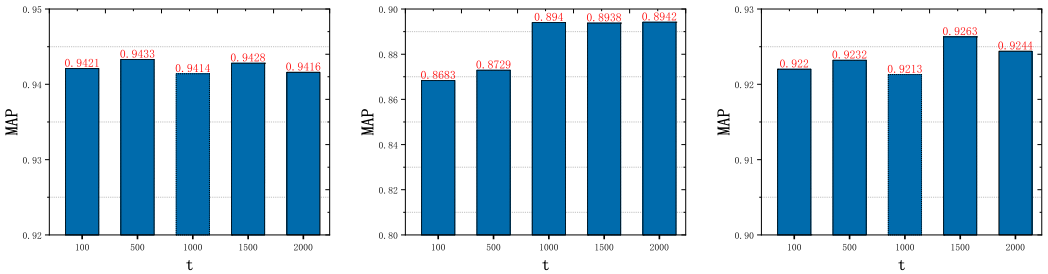
4.2.2 DUDH vs. ADSH. For CIFAR-10 and SVHN, DUDH performs better than ADSH-based methods in all cases. The reason lies in the fact that the quantization error of DUDH is much smaller than that of ADSH-based methods. Due to the tight coupling between the query and database, the quantization errors of ADSH-based methods are directly related to the size of the similarity matrix between the database and query ($m \times n$). However, the quantization error of DUDH is only related to the size of the similarity matrix between the query and similarity-transfer set ($m \times t$). Meanwhile, DUDH can implicitly preserve similarity between database and query through the similarity-transfer set, which is achieved by keeping its similarity to both the database and the query. For NUS-WIDE, DUDH achieves a comparative performance with ADSH. The reason is that NUS-WIDE is a multi-label dataset, in which the similarities are more complex. Thus, a higher standard for the selection of the similarity-transfer set is demanded.

4.2.3 DUDH vs. CSQ. We further compare DUDH with the Hadamard-matrix-based method CSQ [52] in the aspect of MAP and training time. For MAP, as listed in Table 3, DUDH is superior to

Table 4. Comparison of MAP and Training Time on MS-COCO

Methods	MAP				Training Time				
	12 bits	24 bits	32 bits	48 bits	θ	W	V	**Total**	**MAP**
CSQ [52]	0.744	0.844	0.872	0.886	-	-	-	512.1	0.886
CSQ-$\frac{1}{2}$	0.690	0.787	0.809	0.828	-	-	-	347.9	0.828
CSQ-$\frac{1}{8}$	0.635	0.729	0.755	0.769	-	-	-	57.9	0.769
CSQ-$\frac{1}{16}$	0.605	0.693	0.724	0.738	-	-	-	9.6	0.738
DUDH(Ours)	**0.871**	**0.899**	**0.901**	**0.910**	19.5	0.04	0.06	**23.6**	**0.910**

For the training time, the code length is 48. CSQ-$\frac{1}{2}$ / $\frac{1}{8}$ / $\frac{1}{16}$ denotes CSQ trained with $\frac{1}{2}$ / $\frac{1}{8}$ / $\frac{1}{16}$ training images.



(a) CIFAR-10 (b) NUS-WIDE (c) SVHN

Fig. 6. Impact of the size of similarity-transfer set on MAP. The code length is 24.

CSQ in most cases. More specifically, DUDH gains obvious advantages on the NUS-WIDE dataset. For CIFAR-10 and SVHN, DUDH is a little inferior to CSQ in a few cases. For training time, as listed in Table 5, DUDH can achieve better retrieval performance using much less training time. Table 4 shows the comparison results on MS-COCO. DUDH shows much better retrieval performance than CSQ on MS-COCO. Meanwhile, the training efficiency of DUDH is much higher than that of CSQ. Besides, with the decreasing number of training images, though CSQ may cost less training time (still much slower than DUDH), the gap between the two methods is getting wider as well.

4.2.4 Impact of the Size of Similarity-Transfer Set. To evaluate the impact of the size of the similarity-transfer matrix on retrieval accuracy, we plot the MAP results of DUDH with different values of t in Figure 6. For CIFAR-10 and SVHN, the size of the similarity-transfer matrix almost has no impact on the retrieval performance. More specifically, when the size of the similarity-transfer matrix is small, e.g., $t = 100$, DUDH can still achieve competitive retrieval performance. For NUS-WIDE, the MAP value improves with the increasing of the size of the similarity-transfer set at first and then stays stable. Therefore, it requires more similarity-transfer images for DUDH to preserve semantic similarity between images in NUS-WIDE than those in CIFAR-10 and SVHN. It is because the semantic similarities between images in CIFAR-10 and SVHN are much simpler than the ones in NUS-WIDE.

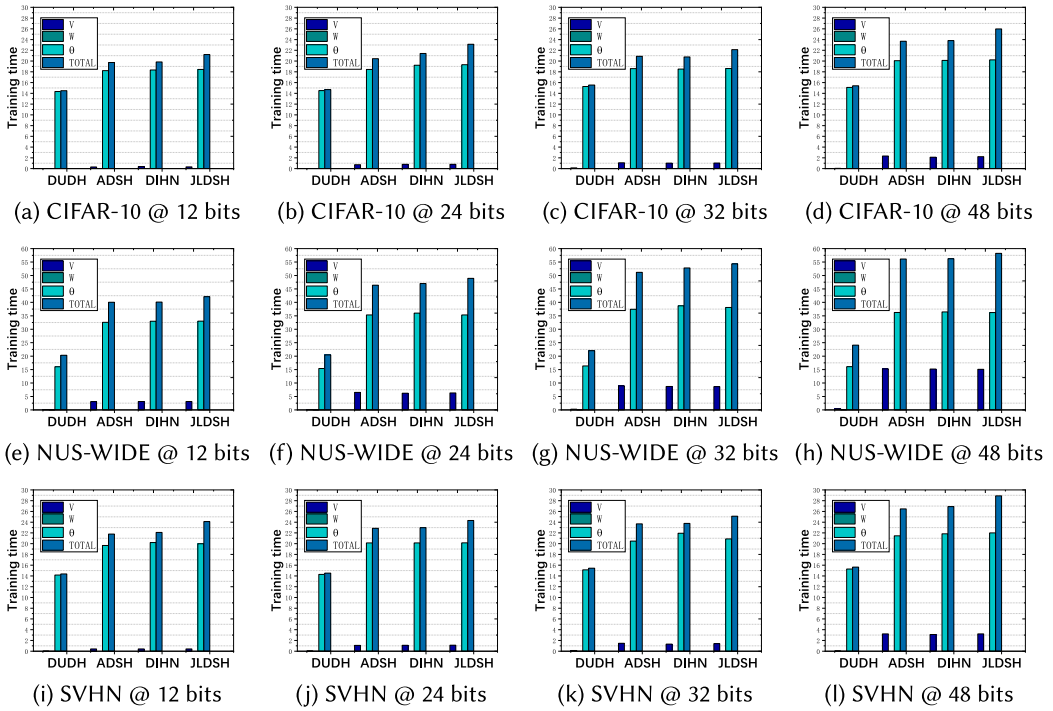
4.3 Time Complexity

4.3.1 Comparison of Training Costs. Our other contribution is to accelerate the hash code learning process. Therefore, we further compare the training costs of DUDH with other deep hashing methods. Table 5 displays the detailed computation costs of the six methods on three datasets with 48 bits. Note that the training time of DIHN includes the one for base model training and the one for incremental learning. Both W and V are updated on GPU. In general, DUDH shows the best training efficiency, while ADSH is the second best, which is consistent with the computational complexity analysis in Section 3.6.

Table 5. Comparison of Training Time (in Minutes) for Different Variables on Three Datasets with 48 Bits

Methods	CIFAR-10					NUS-WIDE					SVHN				
	θ	W	V	**Total**	**MAP**	θ	W	V	**Total**	**MAP**	θ	W	V	**Total**	**MAP**
ADSH [18]	20.1	-	2.3	23.7	0.939	36.2	-	15.3	56.1	0.906	21.5	-	3.2	26.5	0.919
DIHN [44]	20.2	-	2.4	23.9	0.940	36.3	-	16.1	57.3	0.902	22.2	-	3.1	27.3	0.915
CSQ [52]	-	-	-	92.2	0.947	-	-	-	309.8	0.886	-	-	-	515.2	0.935
JLDSH [13]	20.2	-	-	26.1	0.942	36.5	-	-	58.4	0.910	21.8	-	3.1	29.0	0.886
DUDH(*Ours*)	15.1	0.004	0.06	**15.3**	0.948	16.1	0.07	0.4	**24.1**	0.906	15.3	0.003	0.05	**15.7**	0.941

W and V are updated on GPU.



(a) CIFAR-10 @ 12 bits (b) CIFAR-10 @ 24 bits (c) CIFAR-10 @ 32 bits (d) CIFAR-10 @ 48 bits

(e) NUS-WIDE @ 12 bits (f) NUS-WIDE @ 24 bits (g) NUS-WIDE @ 32 bits (h) NUS-WIDE @ 48 bits

(i) SVHN @ 12 bits (j) SVHN @ 24 bits (k) SVHN @ 32 bits (l) SVHN @ 48 bits

Fig. 7. Comparison of the training time (in minutes) among asymmetric deep hashing methods on three datasets. The four sub-figures in each row show the training time on the same dataset with different code length. Best viewed in color.

For example, it takes only about 24 minutes for DUDH to achieve promising performance on NUS-WIDE. In contrast, ADSH costs more than two times (56 minutes) as much as DUDH to reach the similar retrieval accuracy. The main advantage of DUDH lies in the process of optimizing θ and V. For example, it only takes 0.4 minutes for DUDH to optimize V, while it costs 15.3 minutes for ADSH to complete it. Besides, the training cost for W is very little. Therefore, the overhead for learning the similarity-transfer matrix can almost be ignored. For DUDH, the training time for θ is longer related to the size of database n, which can drastically reduce the training costs especially for large datasets. Besides, the training time for V is proportional to the size of similarity-transfer set t instead of the size of query set m, which can also contribute to the reduction of training cost since $t < m$.

To further prove the superiority of DUDH, we compare the training costs between DUDH and the three most competitive methods, i.e., JLDSH, DIHN, and ADSH. As shown in Figure 7, DUDH

Table 6. Comparison of Training Time (in Minutes) for Different Variables with CPU on Three Datasets with 48 Bits

Methods	CIFAR-10			NUS-WIDE			SVHN		
	W	V	T	W	V	T	W	V	T
ADSH [18]	-	52.1	72.9	-	218.9	264.8	-	67.8	90.8
DIHN [44]	-	53.2	73.4	-	220.2	266.3	-	68.2	91.4
JLDSH [13]	-	53.8	78.7	-	221.4	273.3	-	69.9	97.3
DUDH (*Ours*)	0.02	3.2	**19.1**	0.4	103.9	**124.6**	0.02	3.75	**19.6**

can gain obvious advantages in all cases. It clearly shows that the training cost for W is very little under all circumstances, which means the optimization cost for W can be ignored. As expected, the advantages of DUDH are more obvious with the increasing length of hash codes. In summary, the optimization cost produced by similar-transfer set W can be ignored, while the cost for θ and V can be largely reduced.

4.3.2 Differences between Optimizing with GPU and CPU. Table 6 shows the training time for W and V with GPU/CPU on three datasets with 48 bits. Note that it is time consuming to optimize θ with CPU. Therefore, we only discuss W and V here. Generally, it takes much more time for JLDSH, ADSH, DIHN, and DUDH to update V and W on CPU than on GPU. The advantages of DUDH become much more obvious when training with CPU. For example, on CIFAR-10, ADSH costs about four times (72.9 minutes) as much as DUDH to reach the similar retrieval accuracy. On SVHN, ADSH costs about five times (90.8 minutes) as much as DUDH to converge, and its retrieval performance on SVHN is also worse than that of DUDH. An interesting observation is that it costs less time to train DUDH with CPU than to train ADSH/DIHN with GPU on CIFAR-10 and SVHN. Besides, though it takes additional time for DUDH to optimize W with CPU, the training cost for W is still negligible, which can further prove the superiority of DUDH.

4.3.3 Impact of the Size of Similarity-Transfer Set. Figure 8 shows the training time of DUDH when the size of W varies. In general, with the increasing value of t, the training time for optimizing W, V, and θ increases accordingly, which is consistent with the computational complexity listed in Table 2. Specifically, when optimizing V with CPU, with the increasing value of t, the training time for optimizing V rises notably. When optimizing V with GPU, the training cost of V is dramatically reduced, which is much less than the training cost of θ. For W, its training cost can almost be ignored no matter whether it is optimized with GPU or CPU.

4.3.4 MAP Comparison When Training Time Is Similar. We further conduct experiments that reduce the number of training or query images for ADSH-based methods (JLDSH, ADSH, and DIHN) to make them have a similar training time as DUDH. Table 7 shows the comparison of MAP on three datasets with 48 bits when the training time is similar by reducing the number of training images. Specifically, we randomly select a subset of database images as the training images. After the training process, we adopt the CNN model to generate hash codes for database images, which is different from the original settings. As listed in the table, the training time of ADSH-based methods can drop to a similar level as DUDH when using only a quarter of database images. However, the performances of existing methods also drop significantly accordingly, and DUDH can achieve much better retrieval performance with similar training time. Table 8 shows the comparison of MAP on three datasets with 48 bits when the training time is similar by reducing the number of sampled query images. Specifically, we reduce the number of query images sampled from database images. Similar to the previous results, DUDH can achieve the best retrieval performance while

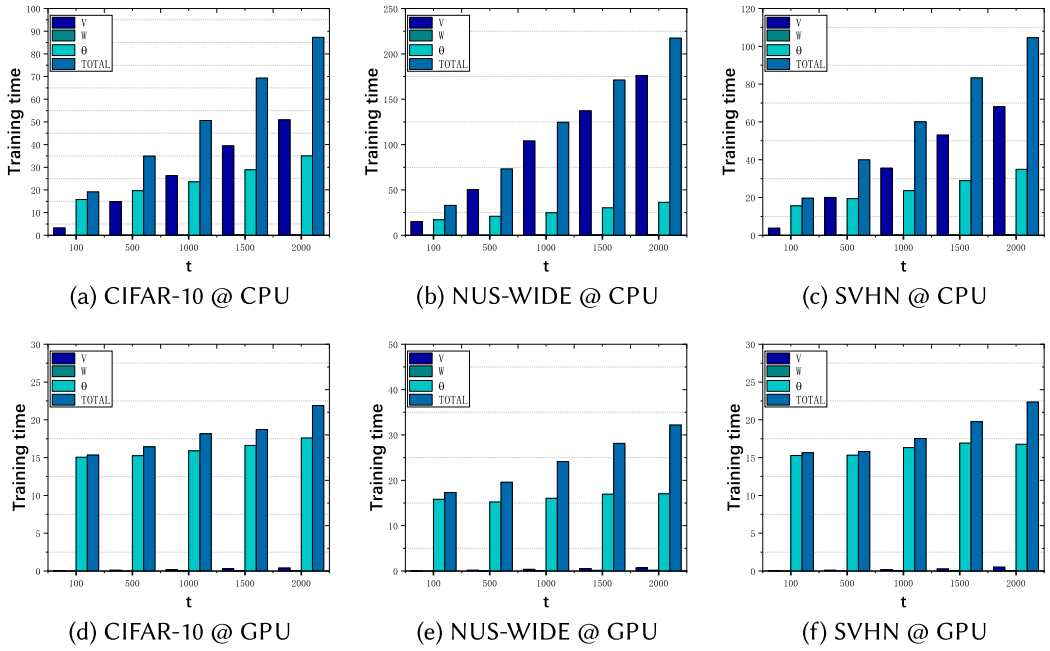


Fig. 8. Comparison of the training time (in minutes) for different variables when t varies on three datasets. The four sub-figures in each row show the training time on three datasets when optimizing W and V with CPU/GPU. Best viewed in color.

Table 7. Comparison of MAP on Three Datasets with 48 Bits When Training Time Is Similar by Reducing the Number of Training Images

Methods	CIFAR-10			NUS-WIDE			SVHN		
	sr	time	MAP	sr	time	MAP	sr	time	MAP
ADSH [18]	0.25	17.5	0.925	0.25	24.3	0.809	0.25	18.2	0.864
DIHN [44]	0.25	17.3	0.922	0.25	24.6	0.812	0.25	17.9	0.871
JLDSH [13]	0.25	18.5	0.935	0.25	25.7	0.823	0.25	19.7	0.832
DUDH(*Ours*)	1	**15.3**	**0.948**	1	**24.1**	**0.906**	1	**15.7**	**0.941**

W and V are updated on GPU. sr denotes sampling rate.

using the least time. However, the performance gap between ADSH-based methods and DUDH has narrowed compared to Table 7. The reason is that the database hash codes in the former experiment have to be generated by the CNN model, thus leading to the larger quantization errors. In contrast, the database hash codes in latter experiment are directly learned in the optimization process and the quantization errors are smaller.

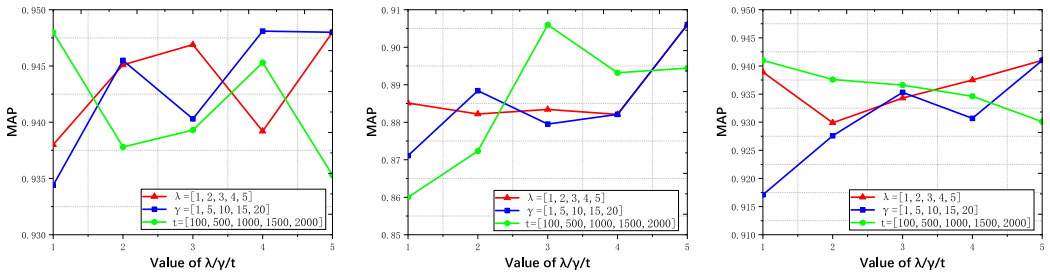
4.4 Parameter Sensitivity

Figure 9 shows the performance of DUDH on three datasets with different parameters. We tune one parameter with the other two fixed. More specifically, on CIFAR-10 and SVHN, we tune λ in the range of [1, 2, 3, 4, 5] by fixing $t = 100$ and $\gamma = 20$, respectively. On NUS-WIDE, we tune λ in the range of [1, 2, 3, 4, 5] by fixing $t = 1{,}000$ and $\gamma = 20$, respectively. Similarly, we set $\lambda = 5$ and $\gamma = 20$ when tuning t. For γ, on CIFAR-10 and SVHN, we tune γ in the range of [1, 5, 10, 15, 20] by fixing $t = 100$ and $\lambda = 5$. On NUS-WIDE, we tune γ in the range of [1, 5, 10, 15, 20] by fixing $t = 1{,}000$ and $\lambda = 5$. As shown in the figure, DUDH obtains stable performance when the values of

Table 8. Comparison of MAP on Three Datasets with 48 Bits When Training Time Is Similar by Reducing the Number of Query Images

Methods	CIFAR-10			NUS-WIDE			SVHN		
	m	time	MAP	m	time	MAP	m	time	MAP
ADSH [18]	1500	17.6	0.930	1000	25.1	0.881	1200	16.4	0.910
DIHN [44]	1500	17.3	0.926	1000	25.6	0.878	1200	16.5	0.912
JLDSH [13]	1500	19.8	0.933	1000	27.1	0.889	1200	18.5	0.870
DUDH(*Ours*)	2000	**15.3**	**0.948**	2000	**24.1**	**0.906**	2000	**15.7**	**0.941**

W and V are updated on GPU.



(a) CIFAR-10 (b) NUS-WIDE (c) SVHN

Fig. 9. MAPs versus the variations of λ, γ, and t on three datasets. The code length is 48.

γ and λ vary. For t, on CIFAR-10 and SVHN, it has almost no impact on MAP. On NUS-WIDE, the MAP value improves with the increasing of t, which is consistent with the results aforementioned.

5 CONCLUSIONS

In this article, we propose a novel asymmetric deep hashing method, namely DUDH, for large-scale image retrieval. Different from the tight coupling between query and database in conventional asymmetric deep hashing methods, DUDH adopts a similarity-transfer matrix to decouple the query from the database. As a result, both the quantization error and the cost of optimizing the CNN model in our DUDH are no longer related to the size of the database and can be largely reduced. Meanwhile, the training cost for optimizing similarity-transfer codes can be ignored with the proposed constant-approximation optimization solution. Extensive experiments demonstrate that DUDH can achieve state-of-the-art performance with much less training cost.

APPENDIX

A PROOF OF THE THEOREM 3.1

First, let's consider the following problem:

$$\begin{aligned} \min_{W} J_3(W) &= ||VW^T - c\widetilde{S}||_1 + \lambda||PW^T - c\widehat{S}||_1, \\ s.t. \quad & W \in \{-1, +1\}^{t\times c}. \end{aligned} \tag{16}$$

Suppose that $J_3(W)$ reaches its minimum at the point W_3^*. We will first prove that $J_2(W_1^*) \leq 2cJ_3(W_1^*)$:

$$\begin{aligned} J_2(W_1^*) &= ||VW_1^{*T} - c\widetilde{S}||_F^2 + \lambda||PW_1^{*T} - c\widehat{S}||_F^2, \\ &= c^2\left(\left\|\frac{1}{c}VW_1^{*T} - \widetilde{S}\right\|_F^2 + \lambda\left\|\frac{1}{c}PW_1^{*T} - \widehat{S}\right\|_F^2\right). \end{aligned} \tag{17}$$

Since all elements of $V, W, \widehat{S}, \widetilde{S}$, and P are in $[-1, +1]$, we have $\frac{1}{c}VW_1^{*T} - \widetilde{S} \in [-2, +2]^{n\times t}$ and $\frac{1}{c}PW_1^{*T} - \widehat{S} \in [-2, +2]^{m\times t}$. Hence:

$$\left\|\frac{1}{c}VW_1^{*T} - \widetilde{S}\right\|_F^2 + \lambda\left\|\frac{1}{c}PW_1^{*T} - \widehat{S}\right\|_F^2 \tag{18}$$

$$\leq 2\left(\left\|\frac{1}{c}VW_1^{*T} - \widetilde{S}\right\|_1 + \lambda\left\|\frac{1}{c}PW_1^{*T} - \widehat{S}\right\|_1\right). \tag{19}$$

Then we have

$$\begin{aligned} J_2(W_1^*) &\leq 2c\left(c\left\|\frac{1}{c}VW_1^{*T} - \widetilde{S}\right\|_1 + c\lambda\left\|\frac{1}{c}PW_1^{*T} - \widehat{S}\right\|_1\right) \\ &= 2cJ_3(W_1^*). \end{aligned} \tag{20}$$

Recall that

$$J_1(W) = ||VW^T - c\widetilde{S} + \lambda(PW^T - c\widehat{S})||_1. \tag{21}$$

Then we will prove $J_3(W_1^*) \leq J_1(W_1^*) + 4c\lambda mt$.

It is easy to get

$$\begin{aligned} &||VW^T - c\widetilde{S}||_1 + \lambda||PW^T - c\widehat{S}||_1 \\ &\leq ||VW^T - c\widetilde{S} + \lambda(PW^T - c\widehat{S})||_1 + 2||\lambda(PW^T - c\widehat{S})||_1 \end{aligned} \tag{22}$$

and

$$\begin{aligned} ||(PW^T - c\widehat{S})||_1 &\leq 2cmt \\ 2||\lambda(PW^T - c\widehat{S})||_1 &\leq 4c\lambda mt. \end{aligned} \tag{23}$$

Combining Equations (22) and (23), we can get

$$J_3(W_1^*) \leq J_1(W_1^*) + 4c\lambda mt. \tag{24}$$

Then, combining Equations (20) and (24), we have

$$\begin{aligned} J_2(W_1^*) \leq 2cJ_3(W_1^*) &\leq 2cJ_1(W_1^*) + 8c^2\lambda mt, \\ &\leq 2cJ_1(W_2^*) + 8c^2\lambda mt, \\ &\leq 2cJ_3(W_2^*) + 8c^2\lambda mt. \end{aligned} \tag{25}$$

Next, we will prove $J_3(W_2^*) \leq J_2(W_2^*) + \frac{1}{4}\lambda mt$.

Since all the elements in $VW_2^{*T} - c\widetilde{S}$ are all integer value, it is easy to have

$$||VW_2^{*T} - c\widetilde{S}||_1 \leq ||VW_2^{*T} - c\widetilde{S}||_F^2. \tag{26}$$

For the elements in $PW_2^{*T} - c\widehat{S}$ the values of which are in $[-1, 1]$ (assume they are indexed by ψ), we will prove

$$||(PW_2^{*T} - c\widehat{S})_\psi||_1 \leq ||(PW_2^{*T} - c\widehat{S})_\psi||_F^2 + \frac{1}{4}|\psi|, \tag{27}$$

where $|\psi|$ denotes the size of ψ.

First, considering the following function:

$$f(x) = -x^2 + x, \tag{28}$$

when $x = \frac{1}{2}$, $f(x)$ reaches its maximum value $\frac{1}{4}$. Then we can easily prove Equation (27).

Then we have

$$\begin{aligned} \lambda||(PW_2^{*T} - c\widehat{S})_\psi||_1 &\leq \lambda||(PW_2^{*T} - c\widehat{S})_\psi||_F^2 + \frac{1}{4}\lambda|\psi|, \\ &\leq \lambda||(PW_2^{*T} - c\widehat{S})_\psi||_F^2 + \frac{1}{4}\lambda mt. \end{aligned} \tag{29}$$

For the left elements in $PW_2^{*T} - c\widehat{S}$ (assume they are indexed by ϕ), we have

$$\lambda||(PW_2^{*T} - c\widehat{S})_\phi||_1 \leq \lambda||(PW_2^{*T} - c\widehat{S})_\phi||_F^2. \tag{30}$$

Combining Equations (26), (29), and (30), we can get

$$J_3(W_2^*) \leq J_2(W_2^*) + \frac{1}{4}\lambda mt. \tag{31}$$

Finally, combining Equations (25) and (31), we have

$$\begin{aligned} J_2(W_1^*) &\leq 2cJ_3(W_2^*) + 8c^2\lambda mt, \\ &\leq 2cJ_2(W_2^*) + \left(\frac{1}{2}c + 8c^2\right)\lambda mt. \end{aligned} \tag{32}$$

REFERENCES

[1] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. 2016. Deep quantization network for efficient image retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3457–3463.

[2] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference*.

[3] Yudong Chen, Zhihui Lai, Yujuan Ding, Kaiyi Lin, and Wai Keung Wong. 2019. Deep supervised hashing with anchor graph. In *Proceedings of the IEEE International Conference on Computer Vision*. 9795–9803.

[4] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. NUS-WIDE: A real-world web image database from National University of Singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*.

[5] Hui Cui, Lei Zhu, Jingjing Li, Yang Yang, and Liqiang Nie. 2019. Scalable deep hashing for large-scale social image retrieval. *IEEE Transactions on Image Processing* 29 (2019), 1271–1284.

[6] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao. 2018. Triplet-based deep hashing network for cross-modal retrieval. *IEEE Transactions on Image Processing* 27, 8 (2018), 3893–3903.

[7] Cheng Deng, Erkun Yang, Tongliang Liu, Jie Li, Wei Liu, and Dacheng Tao. 2019. Unsupervised semantic-preserving adversarial hashing for image search. *IEEE Transactions on Image Processing* 28, 8 (2019), 4032–4044.

[8] Cheng Deng, Erkun Yang, Tongliang Liu, and Dacheng Tao. 2019. Two-stream deep hashing with class-specific centers for supervised image search. *IEEE Transactions on Neural Networks and Learning Systems* 31, 6 (2019), 2189–2201.

[9] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi, Yanhua Yang, Cheng Deng, and Heng Huang. 2018. Unsupervised deep generative adversarial hashing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3664–3673.

[10] Thanh Toan Do, Anh Dzung Doan, and Ngai Man Cheung. 2016. Learning to hash with binary deep neural network. In *Proceedings of the European Conference on Computer Vision*. 219–234.

[11] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Data Bases*. 518–529.

[12] Yunchao Gong and Svetlana Lazebnik. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 817–824.

[13] Guanghua Gu, Jiangtao Liu, Zhuoyi Li, Wenhua Huo, and Yao Zhao. 2020. Joint learning based deep supervised hashing for large-scale image retrieval. *Neurocomputing* 385 (2020), 348–357.

[14] Tao He, Yuan-Fang Li, Lianli Gao, Dongxiang Zhang, and Jingkuan Song. 2019. One network for multi-domains: Domain adaptive hashing with intersectant generative adversarial networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 2477–2483.

[15] Xiangyu He, Peisong Wang, and Jian Cheng. 2019. K-Nearest neighbors hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2839–2848.

[16] Qing-Yuan Jiang, Xue Cui, and Wu-Jun Li. 2018. Deep discrete supervised hashing. *IEEE Transactions on Image Processing* 27 (2018), 5996–6009.

[17] Qing-Yuan Jiang and Wu-Jun Li. 2017. Deep cross-modal hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3270–3278.

[18] Qing-Yuan Jiang and Wu-Jun Li. 2018. Asymmetric deep supervised hashing. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3342–3349.

[19] Qing-Yuan Jiang and Wu-Jun Li. 2019. Discrete latent factor model for cross-modal hashing. *IEEE Transactions on Image Processing* 28, 7 (2019), 3490–3501.

[20] S. Jin, H. Yao, X. Sun, S. Zhou, L. Zhang, and X. Hua. 2020. Deep saliency hashing for fine-grained retrieval. *IEEE Transactions on Image Processing* 29 (2020), 5336–5351.
[21] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou. 2016. Column sampling based discrete supervised hashing. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1230–1236.
[22] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report.
[23] Chuanxiang Li, Ting-Kun Yan, Xin Luo, Liqiang Nie, and Xin-Shun Xu. 2019. Supervised robust discrete multimodal hashing for cross-media retrieval. *IEEE Transactions on Multimedia* 21 (2019), 2863–2877.
[24] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. 2017. Deep supervised discrete hashing. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems*. 2482–2491.
[25] Shuyan Li, Zhixiang Chen, Jiwen Lu, Xiu Li, and Jie Zhou. 2019. Neighborhood preserving hashing for scalable video retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*. 8212–8221.
[26] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. 2016. Feature learning based deep supervised hashing with pairwise labels. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 1711–1717.
[27] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton Van den Hengel, and David Suter. 2014. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1963–1970.
[28] Kevin Lin, Jiwen Lu, Chu Song Chen, and Jie Zhou. 2016. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1183–1192.
[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft Coco: Common objects in context. In *European Conference on Computer Vision*. Springer, 740–755.
[30] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2064–2072.
[31] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2074–2081.
[32] Xin Luo, Peng Fei Zhang, Zi Huang, Liqiang Nie, and Xin Shun Xu. 2019. Discrete hashing with multiple supervision. *IEEE Transactions on Image Processing* 28 (2019), 2962–2975.
[33] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems*.
[34] Xiushan Nie, Weizhen Jing, Chaoran Cui, Jason Zhang, Lei Zhu, and Yilong Yin. 2019. Joint multi-view hashing for large-scale near-duplicate video retrieval. *IEEE Transactions on Knowledge and Data Engineering* 32 (2019), 1951–1965.
[35] Maxim Raginsky and Svetlana Lazebnik. 2009. Locality-sensitive binary codes from shift-invariant kernels. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems*. 1509–1517.
[36] Fumin Shen, Xin Gao, Li Liu, Yang Yang, and Heng Tao Shen. 2017. Deep asymmetric pairwise hashing. In *Proceedings of the ACM International Conference on Multimedia*. 1522–1530.
[37] Fumin Shen, Yadong Mu, Yang Yang, Wei Liu, Li Liu, Jingkuan Song, and Heng Tao Shen. 2017. Classification by retrieval: Binarizing data and classifiers. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 595–604.
[38] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 37–45.
[39] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. 2018. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (2018), 3034–3044.
[40] Jinhui Tang, Jie Lin, Zechao Li, and Jian Yang. 2018. Discriminative deep quantization hashing for face image retrieval. *IEEE Transactions on Neural Networks and Learning Systems* 29 (2018), 6154–6162.
[41] Andrea Vedaldi and Karel Lenc. 2015. Matconvnet: Convolutional neural networks for Matlab. In *Proceedings of the ACM International Conference on Multimedia*. 689–692.
[42] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2018. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 4 (2018), 769–790.
[43] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems*. 1753–1760.
[44] Dayan Wu, Qi Dai, Jing Liu, Bo Li, and Weiping Wang. 2019. Deep incremental hashing network for efficient image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9069–9077.
[45] Lin Wu, Yang Wang, Zongyuan Ge, Qichang Hu, and Xue Li. 2018. Structured deep hashing with convolutional neural networks for fast person re-identification. *Computer Vision and Image Understanding* 167 (2018), 63–73.

[46] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. 2014. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2156–2162.
[47] De Xie, Cheng Deng, Chao Li, Xianglong Liu, and Dacheng Tao. 2020. Multi-task consistency-preserving adversarial hashing for cross-modal retrieval. *IEEE Transactions on Image Processing* 29 (2020), 3626–3637.
[48] Xinyu Yan, Lijun Zhang, and Wu Jun Li. 2017. Semi-supervised deep hashing with a bipartite graph. In *Proceedings of the International Joint Conference on Artificial Intelligence*.
[49] Erkun Yang, Cheng Deng, Tongliang Liu, Wei Liu, and Dacheng Tao. 2018. Semantic structure-based unsupervised deep hashing. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 1064–1070.
[50] Erkun Yang, Cheng Deng, Wei Liu, Xianglong Liu, Dacheng Tao, and Xinbo Gao. 2017. Pairwise relationship guided deep hashing for cross-modal retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.
[51] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. 2019. DistillHash: Unsupervised deep hashing by distilling data pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2946–2955.
[52] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis E. H. Tay, Zequn Jie, Wei Liu, and Jiashi Feng. 2020. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3083–3092.
[53] Haofeng Zhang, Li Liu, Yang Long, and Ling Shao. 2018. Unsupervised deep hashing with pseudo labels for scalable image retrieval. *IEEE Transactions on Image Processing* 27 (2018), 1626–1638.
[54] Jian Zhang and Yuxin Peng. 2019. Multi-pathway generative adversarial hashing for unsupervised cross-modal retrieval. *IEEE Transactions on Multimedia* 22 (2019), 174–187.
[55] Jian Zhang and Yuxin Peng. 2019. SSDH: Semi-supervised deep hashing for large scale image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology* 29 (2019), 212–225.
[56] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. 2014. Supervised hashing with latent factor models. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 173–182.
[57] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. 2015. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing* 24 (2015), 4766–4779.
[58] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. 2015. Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1556–1564.
[59] Xiang Zhou, Fumin Shen, Li Liu, Wei Liu, Liqiang Nie, Yang Yang, and Heng Tao Shen. 2020. Graph convolutional network hashing. *IEEE Transactions on Cybernetics* 50 (2020), 1460–1472.
[60] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep hashing network for efficient similarity retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2415–2421.
[61] Bohan Zhuang, Guosheng Lin, Chunhua Shen, and Ian Reid. 2016. Fast training of triplet-based deep binary embedding networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5955–5964.

Received 15 August 2021; revised 22 January 2022; accepted 2 March 2022