Bootstrap Dual Complementary Hashing with Semi-Supervised Re-ranking for Image Retrieval

Xing Tian[a], Xiancheng Zhou[a], Wing W. Y. Ng[a,*], Jiayong Li[a], Hui Wang[b]

[a] *Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou, China 510006*
[b] *School of Computing, Ulster University, Jordanstown, United Kingdom*

Abstract

With the rapid growth of multimedia data on the Internet, content-based image retrieval becomes a key technique for the Internet development. Hashing methods are efficient and effective for image retrieval. Dual Complementary Hashing (DCH) is one such method, which uses multiple hash tables and has good performance. However, DCH utilizes wrongly hashed image pairs to train the following hash table and discards correctly hashed image pairs. Therefore, the number of image pairs utilized for training the following hash tables will decrease rapidly. Moreover, each hash function in a hash table of DCH is trained by correcting the errors caused by its preceding one instead of holistically considering errors made by all previous hash functions. These restrictions significantly reduce the training efficiency and the overall performance of DCH. In this paper, we propose a new hashing method for image retrieval, *Bootstrap Dual Complementary Hashing with semi-supervised Re-ranking* (BDCHR). It is a semi-supervised multi-hashing method consisting of two parts: bootstrap DCH and semi-supervised re-ranking. The first part relieves the restrictions of DCH while the second part further enhances the image retrieval performance. Experimental results show that BDCHR yields better performance than other state-of-the-art multi-hashing methods.

*Corresponding author
Email address: wingng@ieee.org (Wing W. Y. Ng)

Keywords: Multi-hashing; dual complementary hashing; image retrieval; semi-supervised

1. Introduction

Multimedia data on the Internet grows rapidly in recent years which leads to higher requirements for existing data understanding and managing methods [1, 2, 3, 4]. Image retrieval has become an important means of harnessing and harvesting the vast number of images on the Internet. Content-based image retrieval is a class of methods that retrieve images based on image content rather than image meta data [5, 6, 7]. Given an image as the query, it seeks to find relevant images in the database that are similar to the query image based on the content of images. Approximate nearest neighbor search methods [8, 9] return approximately similar images as the returned set, succeeding with excellent retrieval performance in both speed and storage. For image retrieval purpose, compared with accurate search (e.g. [10, 11]) which has optimal performance but is usually time costly, approximate search is generally acceptable which has suboptimal performance but is time efficient. As a representative method for approximate search, hashing-based search has been widely researched in recent years due to its sublinear time complexity and good performance.

Hashing methods generate compact binary hash code for high-dimensional images. Hamming distance between hash codes of images is computed to evaluate their similarities. The main problem with hashing methods is how to generate hash codes. Generally, hash functions are learned firstly which can be regarded as hyperplanes that partition the original feature space into buckets. For each hash hyperplane, images located on different sides of it have different binary hash bits, i.e. 0 and 1; and images on the same side have same binary hash bits. Therefore, K hash functions generate at most 2^K different buckets and each bucket has a unique hash code. This set of K hash functions forms a single hash table. For a given query, Hamming distance is calculated between the query image and images in the database. The nearest images with the smallest Hamming distance can be viewed as being in a Hamming ball with a constant radius in the Hamming space and are returned as the final retrieval results. Existing hashing methods [12, 13, 14, 4] can achieve high retrieval performance when the radius of Hamming ball is small. However, the radius of Hamming ball needs to be increased when more relevant images are required. This could significantly lower the retrieval accuracy.

Multi-hashing methods (e.g. [15, 16, 17, 18]) generate multiple hash tables in order to improve recall rate without yielding a significant drop in precision. The pairwise similarity matrix which records the semantic relationship between image pairs is generally introduced into the objective function of hash functions training for semantic relationship preservation. Moreover, multi-hashing methods employ multiple hash tables and always train these hash tables one by one. Each hash table contains a set of hash functions. Image pairs in the database being wrongly hashed by previous hash tables are usually used to train the next hash table. Hashing methods using multiple hash tables usually yield better retrieval precision-recall performance than hashing methods using a single hash table. Boosting Iterative Quantization Hashing (BIQH) [16], Complementary Hashing (CH) [17], Dual Complementary Hashing (DCH) [15], and bagging-boosting-based semi-supervised multi-hashing with query-adaptive re-ranking (BBSHR) [18] are representative multi-hashing methods. BIQH is supervised and requires all data being labeled. CH is unsupervised and it could not achieve satisfying performance for semantic retrieval problems. Moreover, it has high time complexity due to Eigen-value decomposition. DCH is semi-supervised, which is more practical, and achieves decent performance. However, DCH ignores correctly hashed image pairs by the previous hash tables when training the following hash table. Therefore, the number of image pairs in the pairwise similarity matrix utilized for training reduces sharply. As a result, the performance of DCH cannot be further improved after several iterations of training. BBSHR is proposed recently which employs multiple hash tables in a bagging manner. By partitioning the dataset into several parts, multiple hash tables are trained in parallel, one for each part. A problem is that hash tables are trained independently, without considering the correlation between hash tables which is valuable for image retrieval.

Having considered the advantages and disadvantages of current multi-hashing methods, we propose a semi-supervised multi-hashing method for image retrieval, *Bootstrap Dual Complementary Hashing with semi-supervised Re-ranking* (BDCHR). BDCHR consists of two parts, the bootstrap dual complementary hashing and the semi-supervised re-ranking. In the bootstrap dual complementary hashing part, a hash function in one hash table is trained by correcting the errors caused by all previous ones, rather than only the last one. Furthermore, a hash table is trained by focusing on not only the wrongly hashed image pairs by all previous hash tables, but also correctly hashed image pairs. In the semi-supervised re-ranking part, based on the initial returned results by using multiple hash tables, a re-ranking method is used to further improve its retrieval performance. The contribution of this paper can be summarized as follows:

- BDCHR trains both hash tables and hash functions in a boosting manner. Different from DCH, which trains a new hash function by correcting the errors made by the previous one hash function, BDCHR trains a new hash function based on the errors made by all previous hash functions, considering all previous bits holistically.

- To train a new hash table, different from DCH which sets the weights of correctly hashed image pairs to zero, BDCHR increases the weights of wrongly hashed image pairs and decreases the weights of correctly hashed image pairs. In this way, the number of image pairs available for training the following hash table will not be reduced.

- A semi-supervised re-ranking method is introduced in BDCHR to improve its retrieval performance. BDCHR computes weights of each hash function for each category in advance in an offline manner. For a query image, query-adaptive weight of each hash function is computed. The weighted

Hamming distance is calculated to finally evaluate the similarities between the query and images in the dataset.

The rest of this paper is organized as follows. Related works are briefly introduced in Section 2. In Section 3, BDCHR is proposed. Experimental results and analyses are presented in Section 4. The paper concludes in Section 5.

2. Related works

Hashing methods [19, 20, 21] generally learn a set of hash functions to build a hash table and generate compact hash codes for images. For the hash table with K hash functions, the k^{th} hashing function can be represented as the following form:

$$h_k(x) = sign(w_k^T x + b) \tag{1}$$

where $sign(\bullet)$, x, w and b denote the sign function, the feature vector of one image, the hash mapping vector, and the intercept, respectively. b is 0 and can be omitted when the data set is centralized. The hash table with K hash functions is represented as $H(x) = \{h_1(x), h_2(x), ..., h_K(x)\}$. Hamming distance between two images x_i and x_j is computed to evaluate their similarities. If two images have larger Hamming distance, they are less likely to be similar, and vice versa. The Hamming distance between two images can be computed as follows:

$$d_H(x_i, x_j) = \frac{1}{4}||H(x_i) - H(x_j)||^2 \tag{2}$$

Most of existing hashing methods focus on the training procedure of hash functions in an individual hash table. These methods are introduced in Section 2.1. Hash codes of images in database are computed and stored offline when hash functions are learned. The hash code of the query image and Hamming distances between this query image and images in the database are computed. Images in the database with Hamming distances lower than a threshold are returned as the retrieval result. To achieve higher recall rate, larger Hamming distance threshold is required which will also return many dissimilar images and lead to a rapid decrease in retrieval precision. Hashing methods with multiple hash tables could achieve high recall rates and precisions simultaneously, which are introduced in Section 2.2.

2.1. Hashing methods with single hash table

Generally, according to whether label information is used for the training of hash functions, existing hashing methods can be categorized into unsupervised, supervised and semi-supervised methods. Unsupervised hashing methods learn functions without considering the semantic similarity information between images. Locality Sensitive Hashing (LSH) and its variants [22, 23, 24] are representative unsupervised hashing methods which generate hash functions in a random manner. Based on the data distribution information of images, Principal Component Hashing (PCH) [25] trains hash functions by principal component analysis [26] and utilizes the top-K principal components of the covariance matrix to construct its hashing projections. Iterative Quantization Hashing (ITQ) [12] learns the optimal rotation matrix for the data after the principle component analysis by minimizing the quantization error. SKLSH [27] gets the hash functions by randomly extracting Fourier features of images without considering the data distribution information. Asymmetric Cyclic Hashing [28] generates longer hash code for query image and short hash code for images in database to ensure higher retrieval accuracy and lower storage cost. Two-phase Mapping Hashing [29] projects the images to a high dimensional Hamming space firstly to preserve the initial data structure. Then images are projected to low Hamming space by minimizing the reconstruction error. Spectral Embedded Hashing [30] is a graph-based hashing method, which introduces a new regularizer to the objective function of original Spectral Hashing [13]. Ordinal Constraint Hashing (OCH) [31] trains hash functions based on an ordinal graph to preserve the permutation relation information among images. Distributed Graph Hashing [32] learns hash functions based on data located in a distributed manner. Special Structure-Based Hashing method is proposed in [33] which builds hash functions by preserving the underlying geometric information of images.

Supervised hashing methods train hash functions based on fully labeled dataset. LDA Hashing method [34] projects the high dimensional image descriptors into short binary hamming codes based on the semantic information with the linear discriminant analysis. Supervised Discrete Hashing (SDH) [35] generates compact hash code for images by optimizing a joint learning objective which combines hash code learning and linear classifier training simultaneously. To further improve retrieval performance of SDH, SDH with relaxation is proposed in [36] which learns the regression targets from data directly. Based on SDH, a fast SDH is proposed in [37] which regresses class label to the corresponding hash code. Column Sampling based Discrete Supervised Hashing (COSDISH) [38] optimizes the hash code learning problem without relaxation to achieve more accurate retrieval. Error correcting input and output coding method is proposed in [39] which learns hash codes based on distribution preservation and error correction. In [40], the evaluation for supervised hashing methods is analyzed based on the label information of data. The supervised matrix factorization hashing is a cross-modal hashing method based on collective matrix factorization [41]. Multimodal Discriminative Binary Embedding [42] aims to learn discriminative hash codes for multiple modalities of data to improve the retrieval performance. In recent years, we have also witnessed the rapid development of deep hashing methods which extract high-level features of images based on deep neural networks. For example, supervised deep hashing is proposed in [43] which learns features of images, hash codes, and classification simultaneously based on deep convolutional neural networks. Deep ordinal hashing is proposed in [44] which learns hash code based on the similarity ranking information of images.

Supervised hashing methods generally achieve higher accuracy than unsupervised hashing methods, but suffer from higher time complexity. Moreover, it is also impractical to require dataset being fully labeled. Therefore, semi-supervised hashing methods are proposed, which require the dataset being partially labeled. Sequential Projection Learning for Hashing (SPLH) [14] is a primary semi-supervised hashing method which learns hashing functions sequentially. In SPLH, a new hash function is learned by correcting errors brought by its previous one. Semi-supervised Composite Multi-view Discrete Hashing fuses multiple views information of data to generate hash codes, in which a hash projection is learned for each view [45]. Bootstrap Sequential Projection Learning for Hashing (BSPLH) is proposed in [4] which trains a new hash function by correcting errors caused by all previously learned hash functions.

2.2. Hashing methods with multiple hash tables

Most of existing hash methods focus on the training of a single hash table. However, with multiple hash tables being learned, relevant images of the query image could be returned in a smaller region in Hamming space comparing to hashing methods with a single hash table. Since images with smaller Hamming distance to the query have higher possibility to be similar to query image, therefore multi-table-based hashing methods could generally achieve higher precision-recall performance than single-table-based hashing methods [17].

Complementary Hashing (CH) is a representative multi-table-based hashing method which trains hash tables in a boosting manner [17]. The objective function to generate an individual hash table can be optimized as the eigenvalue decomposition problem. CH suffers from the high time complexity for eigenvalue decomposition, though it may be relieved using a sparse matrix to reduce the burden. Moreover, hash functions in a hash table are trained in a single shot which ignores the correlation information between hash bits. Dual Complementary Hashing (DCH) [15] is another semi-supervised hashing method which trains both hash tables and hash bits in a boosting manner. In DCH, an individual hash table is trained by SPLH which learns a new hash function by correcting errors caused by its previous one. Meanwhile, the correctly hashed pairwise similarity information will be reduced for the training of following hash tables. Boosting Iterative Quantization Hashing with query-adaptive re-ranking

(BIQH) [16] employs multiple hash tables and trains hash tables in a boosting manner with dynamically adjustment of weights of images. BIQH re-orders intermediate returned images by the query-adaptive re-ranking technique to enhance the final retrieval performance. To learn a new hash table, BIQH, CH, and DCH all discard correctly mapped image pairs and focus on incorrectly mapped image pairs. This leads to a significant drop in the number of training images for upcoming hash tables which seriously limit the performance improvement of these multi-table-based hashing methods. Bagging-boosting-based semi-supervised multi-hashing with query-adaptive re-ranking (BBSHR) is proposed in [18] which trains multiple hash tables in a bagging manner. However, hash tables in BBSHR are trained independently. The correlation between hash tables are ignored which is meaningful and should be taken into consideration. Therefore, in this paper, Bootstrap Dual Complementary Hashing with semi-supervised Re-ranking (BDCHR) is proposed to handle these problems.

3. Bootstrap Dual Complementary Hashing with Semi-Supervised Re-ranking

In this paper, BDCHR trains both hash functions and hash tables in a boosting manner which finally generates m hash tables with K hash functions per table. In one hash table, a new hash function is trained by correcting errors made by its previous ones which is similar to the idea in [4]. In order to make hash tables in BDCHR complementary, each hash table is trained by correcting errors made by previous hash tables and m hash tables in BDCHR are trained sequentially. Let $X \in R^{d \times n}$ be the dataset where d and n denote the dimensionality of image descriptor and the number of images, respectively. The labeled subset of X is represented as X_l while unlabeled dataset is represented as X_u, i.e. $X = X_l \cup X_u$ and $X_l \cap X_u = \emptyset$. The dataset is centralized firstly. In BDCHR, for an image descriptor x, its k^{th} hash bit of the t^{th} hash table is computed as follows:

$$h_{t,k} = sign(w_{t,k}^T x) \tag{3}$$

where $h_{t,k}(\bullet)$ and $w_{t,k}$ denote the k^{th} hash function in the t^{th} hash table and the hash projection vector, respectively. The superscript T denotes the transpose of the vector.

In Section 3.1, the training method of hash functions and updating method of the weight matrix for hash tables in BDCHR are introduced. The semi-supervised re-ranking method used in BDCHR is described in detail in Section 3.2.

3.1. Bootstrap Dual Complementary Hashing

In BDCHR, hash functions in each hash table are trained sequentially. Each hash function is learned by correcting errors made by its previous ones. Let P and N denote sets of similar and dissimilar image pairs, respectively. BDCHR aims to make Hamming distances between similar image pairs being small and Hamming distances between dissimilar image pairs being large. This objective of hash can be represented as follows:

$$min E\{d_H(x_i, x_j)|P\} - E\{d_H(x_i, x_j)|N\} \tag{4}$$

where H, $E\{\bullet\}$, and $d_H(\bullet)$ denote the set of hash functions, the expectation function, and the Hamming distance function, respectively. After replacing the Hamming distance function in Eq.4, the objective function of BDCHR with m hash tables and K hash functions per table can be formulated as follows:

$$J(H) = max \sum_{t=1}^{m} \sum_{k=1}^{K} \{ \sum_{(x_i, x_j) \in P} h_{t,k}(x_i) h_{t,k}(x_j) - \sum_{(x_i . x_j) \in N} h_{t,k}(x_i) h_{t,k}(x_j) \} \tag{5}$$

Moreover, the hash code outputted from the t^{th} hash table is computed as follows:

$$H_t(X) = sign(W_t^T X) \tag{6}$$

where W_t denotes the hash projection matrix of the t^{th} hash table. The semantic similarity matrix S stores the pairwise similarity information of labeled data.

The element in S is computed as follows:

$$S_{ij} = \begin{cases} 1 & (x_i, x_j) \in P \\ -1 & (x_i, x_j) \in N \\ 0 & otherwise \end{cases} \tag{7}$$

Thus, the objective function Eq.5 of BDCHR can be rewritten as follows:

$$J(H) = max \sum_{t=1}^{m} tr\{H(X_l)SH(X_l)^T\} \tag{8}$$

i.e.

$$J(W) = max \sum_{t=1}^{m} tr\{sign(W_t^T X_l)Ssign(W_t^T X_l)^T\} \tag{9}$$

By relaxing the constraint of sign function in Eq.9 as in [14, 4], the objective function of BDCHR is transformed as follows:

$$J(W) = max \sum_{t=1}^{m} tr\{W_t^T X_l S X_l^T W_t\} \tag{10}$$

The objective function above only considers the semantic information of labeled images. Given that the whole dataset also consists of many unlabeled data, a hash function which partitioning the dataset evenly achieves maximal entropy of the corresponding hash bit. Thus, to avoid overfitting, a penalty term is added to the objective function as follows:

$$\begin{aligned} J(W) &= max \sum_{t=1}^{m} tr\{W_t^T X_l S X_l^T W_t + \lambda W_t^T X X^T W_t\} \\ &= max \sum_{t=1}^{m} tr\{W_t^T M W_t\} \end{aligned} \tag{11}$$

where $M = X_l S X_l^T + \lambda X X^T$ and λ is the parameter for the penalty term. The bootstrap dual complementary hashing is used to construct hash tables sequentially by solving the objective in Eq.11 which is shown in the following algorithm.

BDCHR trains hash functions in boosting manner. Each hash function is trained by focusing on those wrongly hashed image pairs by its previous ones. Error brought by all previous k hash functions are corrected in the t^{th} hash

Algorithm 1 Bootstrap Dual Complementary Hashing

Input: data X, labeled data X_l, semantic matrix S, length of hash codes K, number of hash table m, parameters α, β, λ, c, δ.

Output: Hashing projections H_t, t=1,2,...m.

Initialize the weight matrix $S^1 = S$;
for $t = 1$ to m **do**
    $X_{tr} = X$
    $S^{t,1} = S^t$
    for $k = 1$ to K **do**
        $M = X_l S^t X_l^t + \lambda X_{tr} X_{tr}^T$
        Extract the first Eigen vector of M: $w_{t,k}$
        Update $S^{t,k+1}$ from $S^{t,k}$ by Eq.12
        Compute the residual: $X_{tr} = X_{tr} - w_{t,k} w_{t,k}^T X_{tr}$
    end for
    Get H_t from $w_{t,k}$, $k = 1, ...K$
    Update S^{t+1} from S^t with Eq.14
end for

table to get the new weight matrix $S^{t,k+1}$. After computing Hamming distances between labeled images using the learned k hash functions, errors are computed based on image pairs with the same label but yielding a large Hamming distance and image pairs with different labels but yielding a small hamming distance. The updating function for new weight matrix can be formalized as follows:

$$S^{t,k+1} = S^{t,1} + \Delta S^{t,k} \tag{12}$$

where $S^{t,1}$ and $\Delta S^{t,k}$ denote the original weight matrix, and increased weight matrix, respectively. Let $D_{ij}^{t,k} = \sum_{s=1}^{k} sign(w_s^T x_i x_j^T w_s)$ denote similarities between labeled images x_i and x_j according to the previously learned k hash functions. $\Delta S^{t,k}$ is evaluated according to the errors caused by all previous hash functions. Thus, we calculate the element in S as follows:

$$\Delta S_{ij}^{t,k} = \begin{cases} (\alpha k - D_{ij}^{t,k})/2k, & D_{ij}^{t,k} - \alpha k < 0 \quad and \quad S_{ij}^{1} > 0 \\ (\beta k - D_{ij}^{t,k})/2k, & D_{ij}^{t,k} - \beta k > 0 \quad and \quad S_{ij}^{1} < 0 \\ 0 & otherwise \end{cases} \tag{13}$$

where α and β denote thresholds of similarity and dissimilarity, respectively. The training procedure of each individual hash table is similar with BSPLH [4] which trains hash functions sequentially. After training one hash table, the similarity matrix is updated and used as weight matrix for the training of the following hash table.

In DCH, elements in the weight matrix for correctly hashed image pairs are set to be zero, which reduces the number of pairwise similarities for the following training seriously. With this drawback, hash tables trained afterwards cannot capture the similarity information of the whole dataset which leads to a low retrieval performance. Therefore, in BDCHR, elements in the weight matrix of correctly hashed image pairs are preserved while elements corresponding to wrongly hashed image pairs are increased. The updated weight matrix S^{t+1} from S^t is based on the errors caused by the previous hash table. After computing Hamming distances between labeled data using the t^{th} hash table, error occurs when images with the same label yield large Hamming distances or images with different labels yield small Hamming distance. The weight matrix S for training the $(t+1)^{th}$ hash table is updated based as follows:

$$S^{t+1} = S^t + c\Delta S^t \tag{14}$$

where c is the parameter to control the stride of updating the S matrix. The element in ΔS^t is computed as follows:

$$\Delta S_{ij}^t = \begin{cases} 1 & d_H(x_i, x_j) > \delta, S_{ij}^t > 0 \\ -1 & d_H(x_i, x_j) < \delta, S_{ij}^t < 0 \\ 0 & otherwise \end{cases} \tag{15}$$

where $d_H(x_i, x_j)$ and δ denote the Hamming distance function and a positive threshold with scale $(0, K)$, respectively.

3.2. Semi-supervised Re-ranking

After multiple hash tables being trained, we employ a semi-supervised re-ranking to further improve the retrieval performance. In the semi-supervised re-ranking method, a pseudo-label is firstly assigned to each unlabeled data. Then the category-specific weight of each hash function is computed. For a given query x_q, based on the learned multiple hash tables, the accumulated Hamming distances between x_q and images in database are computed to return an initial retrieval image set, i.e. X_R. Then, according to the appearance ratio of each category in X_R and pre-calculated category-specific weights of each hash function, the query adaptive weight of each hash function is computed. Finally, the weighted Hamming distances between x_q and the images in X_R are calculated and re-ranked. A subset of X_R with smallest weighted Hamming distance will be returned as the final retrieval results.

In semi-supervised re-ranking procedures of BDCHR, the first step is to assign pseudo-label to unlabeled images in the database based on the labeled data. For each unlabeled image, its top 1% closest labeled images based on Euclidean distance are found. The most frequently appearing category in these labeled images is used as the pseudo-label of the corresponding unlabeled image. Noted that the operation of pseudo-label assignment for all images in the database is time consuming but is conducted offline before queries. Then, according to the performance of each hash function to each category, the category-specific weight of each hash function is computed. In the ideal case, images sharing the same label (including the real label and pseudo-label) are expected to share the same hash code. With this concern, the category-specific weight $v_{h,c}$ for the hash function h with respect to the category c is computed as follows:

$$v_{h,c} = \frac{max(n_-, n_+)}{n_- + n_+} \tag{16}$$

where n_- and n_+ denote the number of images in category c which has hash value -1 and $+1$, respectively. It is obvious that the best performance of one hash function is achieved when all images in category c sharing the same hash value. The value of weight v is in the range $[0.5, 1]$, which is then normalized as follows:

$$v_{h,c} = 2(v_{h,c} - 0.5) \tag{17}$$

The method to compute the category-specific weight of each hash function is similar to that in [18]. The category-specific weight of each hash function is computed offline before retrieval, which is practical for real world applications. With Eq.16 and Eq.17, the matrix V is built to record the category-specific weights of all hash functions, in which its element $V_t(c,k)$ denotes the category-specific weight of the k^{th} hash functions in t^{th} hash table to the category c.

When given the query image x_q, the initial retrieval set X_R is returned based on the accumulated Hamming distance between x_q and images in database. The query has a high probability to have the label same to those with the maximum appearance number of images in X_R. According to the ratio of appearance of each category in X_R and the pre-learned category-specific weight, the query adaptive weight of the k^{th} hash function in the t^{th} hash table is computed as follows:

$$G_t(k) = \frac{\sum_{c=1}^{C} n_c V_t(c,k)}{\sum_{c=1}^{C} n_c} \tag{18}$$

where $G_t(k)$, C, and n_c denote the query adaptive weight function, number of categories in database, and number of images of category c in X_R, respectively. With the query adaptive weight of each hash function in each hash table, the weighted Hamming distance between x_q and the image x_i in X_R is computed as follows:

$$d_w(x_i, x_q) = \frac{1}{2} \sum_{t=1}^{m} \sum_{k=1}^{K} G_t(k) \{abs(h_{t,k}(x_i) - h_{t,k}(x_q))\} \tag{19}$$

A subset of X_R is returned based on the re-ranking according to the weighted Hamming distance.

4. Experiments

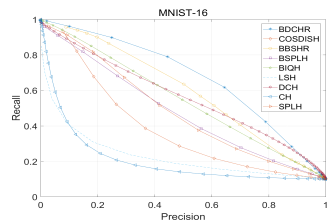
In this section, we evaluate the retrieval performance of BDCHR on four datasets: MNIST, CIFAR-10, USPS, and NUSWIDE. MNIST is an image dataset consisting of $70,000$ handwritten digital images belonging to 10 classes, i.e. 0, 1, 2,..., 9. Each image in MNIST has 28×28 pixels and is represented by a 784-dimension feature (pixel) vector. CIFAR-10 consists of $60,000$ real world images belong to 10 classes, such as dog and cat. Each image is represented by a 512-dimension GIST feature vector. USPS is a dataset consisting of $9,282$ 16×16-pixel images belonging to 10 categories. Each image is represented by a 256-dimension feature vector. NUSWIDE consists of $269,648$ images belonging to 81 categories. Each image is represented by a 500-dimensional bag-of-words feature. For all four databases, 1000 images are randomly selected as the query set while the rest of images are used as the training set. For semi-supervised case, 1000 images are randomly selected from the training set as labeled set. In experiments, recall-and-precision curves are used to evaluate the performance of hashing methods. MAP scores of all hashing methods are also shown in Tables 1, 2, 3, and 4.

In Section 4.1, experimental results of the proposed method, i.e. BDCHR, and comparative methods with different hash code lengths are shown. Recall-and-precision curves and MAP score are employed for evaluation of retrieval performance. Moreover, to further validate the efficiency of BDCHR, BDCHR and single-table hashing methods under the same storage cost of hash codes are also compared in Section 4.2. In Section 4.3, parameters of BDCHR are selected.

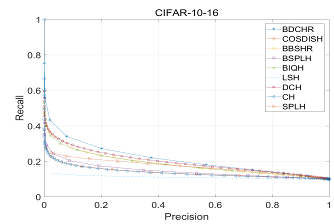
4.1. Experimental Results of BDCHR and Comparative Hashing Methods

In this paper, BDCHR is compared with LSH, SPLH, BSPLH, COSDISH, CH, DCH, BIQH, and BBSHR. Among them, LSH is a representative unsupervised hashing method and used as the baseline method. Both SPLH and BSPLH are representative semi-supervised hashing methods. In BDCHR, BSPLH is also utilized for the training of each single hash table, which makes this method very relevant to the work in this paper. The supervised COSDISH method is also compared in experiments. CH, DCH, BIQH, and BBSHR are representative hashing methods with multiple hash tables. Among comparative methods, both LSH and CH are unsupervised hashing methods while COSDISH and BIQH are supervised hashing method. The proposed method BDCHR is a semi-supervised hashing method with multiple hashing tables. In experiments, the number of hash tables employed by all multi-table-based hashing methods is 5. Recall-and-precision curves of these hashing methods with different hash code lengths, i.e. 16, 24, 32, 48, and 64, on four databases are shown in Figures 1, 2, 3, 4, and 5.

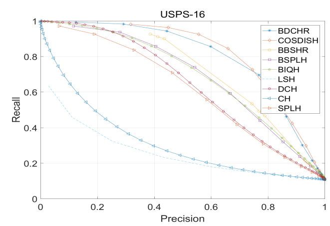
According to these figures, BDCHR achieves outstanding performance comparing with other hashing methods on four databases with different hash code lengths. Retrieval performances of LSH and CH are the worst because their hash functions are trained in unsupervised manners. The data distribution information and semantic information of data are not utilized for training which are very important for similarity preservation. The recently proposed BBSHR method is a semi-supervised multi-table-based hashing method which achieves satisfying performance on four databases with different hash code lengths. This method achieves promising retrieval performance which is just worse than BDCHR. The supervised COSDISH method achieves nearly the



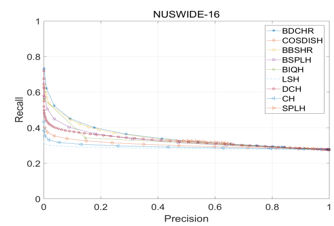
(a) MNIST-16



(b) CIFAR-10-16

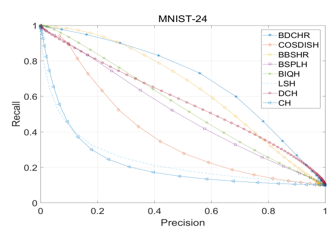


(c) USPS-16

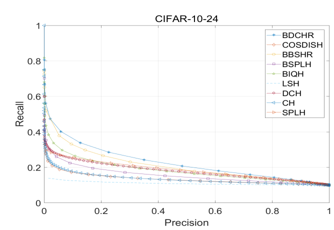


(d) NUSWIDE-16

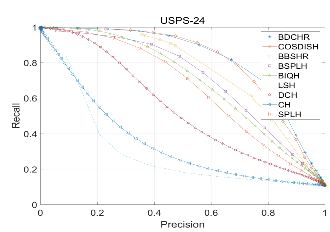
Figure 1: Recall-and-precision curves for 16 bits per table on the MNIST (a), the CIFAR-10 (b), the USPS (c), and the NUSWIDE (d).



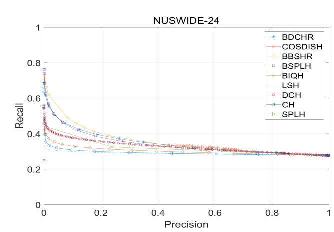
(a) MNIST-24



(b) CIFAR-10-24

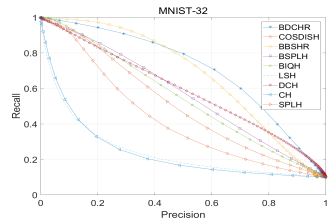


(c) USPS-24

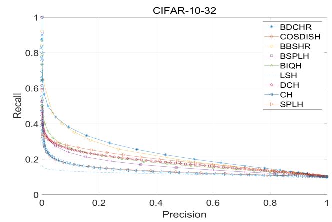


(d) NUSWIDE-24

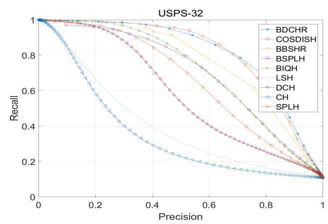
Figure 2: Recall-and-precision curves for 24 bits per table on the MNIST (a), the CIFAR-10 (b), the USPS (c), and the NUSWIDE (d).



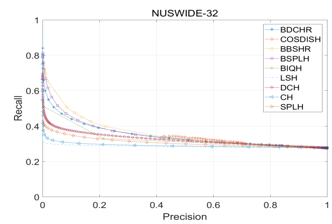
(a) MNIST-32



(b) CIFAR-10-32

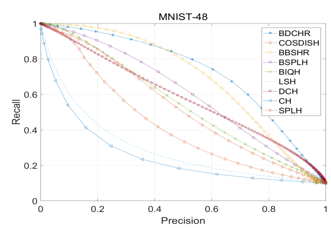


(c) USPS-32

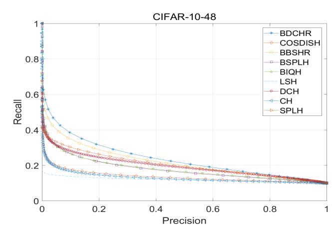


(d) NUSWIDE-32

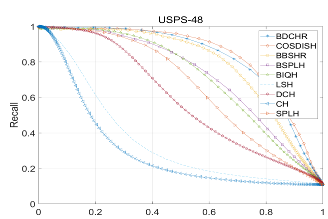
Figure 3: Recall-and-precision curves for 32 bits per table on the MNIST (a), the CIFAR-10 (b), the USPS (c), and the NUSWIDE (d).



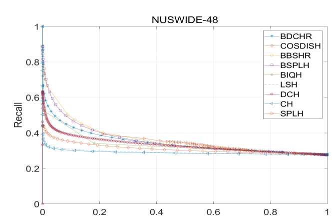
(a) MNIST-48



(b) CIFAR-10-48

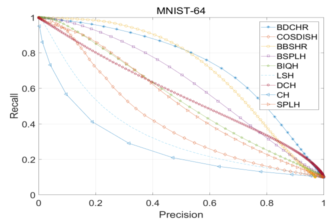


(c) USPS-48

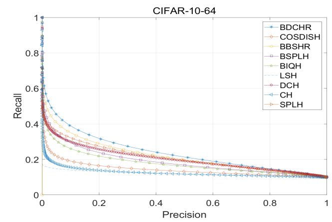


(d) NUSWIDE-48

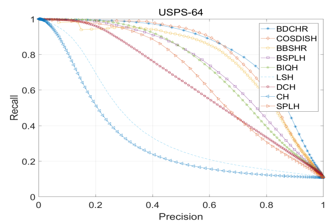
Figure 4: Recall-and-precision curves for 48 bits per table on the MNIST (a), the CIFAR-10 (b), the USPS (c), and the NUSWIDE (d).



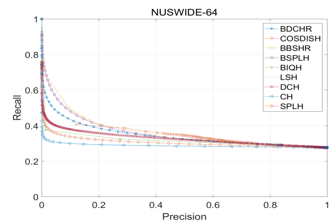
(a) MNIST-64



(b) CIFAR-10-64



(c) USPS-64



(d) NUSWIDE-64

Figure 5: Recall-and-precision curves for 64 bits per table on the MNIST (a), the CIFAR-10 (b), the USPS (c), and the NUSWIDE (d).

best performance on USPS database but very poor performances on other three databases. The reason is that the number of images in USPS is much smaller than other three databases. Given the fact that the training set of each database consists of 1000 images, a higher portion of samples in USPS are utilized as labeled images for training. Therefore, COSDISH achieves good performance with enough supervised information. In contrast, 1000 labeled images only take a low portion in training sets for the other three databases which do not contain enough semantic similarity information for training. Thus COSDISH cannot achieve good retrieval performance for semi-supervised problems. As a semi-supervised method, the proposed BDCHR method which could make full use of both the structural information of unlabeled data and semantic similarity information of labeled data achieves promising retrieval performance. Moreover, the performance of BDCHR gets worse with hash code length increasing. This may be caused that with longer hash bits learned, redundant hash bits are involved. This phenomenon also indicates that BDCHR could achieve promising

Table 1: MAP scores of BDCHR and comparative methods with different hash code lengths on the MNIST

	16 bits	24 bits	32 bits	48 bits	64 bits
BDCHR	64.72%	66.21%	67.5%	69.34%	69.78%
COSDISH	52.58%	53.99%	56.73%	58.33%	58.64%
BBSHR	57.43%	58.54%	59.44%	60.34%	61.43%
BSPLH	47.53%	48.54%	50.34%	52.77%	53.66%
BIQH	57.34%	58.34%	58.54%	58.98%	57.77%
DCH	56.98%	57.8%	58.67%	58.41%	57.14%
CH	25.34%	25.88%	28.12%	32.32%	36.65%
SPLH	46.52%	49.39%	50.13%	52.55%	52.91%
LSH	23%	23.96%	24.51%	29.66%	34.35%

Table 2: MAP scores of BDCHR and comparative methods with different hash code lengths on the CIFAR-10

	16 bits	24 bits	32 bits	48 bits	64 bits
BDCHR	22.44%	24%	24.78%	25.3%	25.54%
COSDISH	20.96%	21.12%	21.12%	21.83%	22.77%
BBSHR	20.43%	19.23%	19.43%	20.33%	21.65%
BSPLH	18.63%	19.43%	21.43%	22.23%	22.76%
BIQH	19.54%	18.34%	19.56%	20.54%	21.64%
DCH	21.72%	19.77%	20.01%	21.44%	22.17%
CH	12.54%	12.77%	12.88%	13.04%	13.65%
SPLH	18.82%	20.41%	21.18%	22.35%	23.07%
LSH	11.69%	11.46%	12.25%	12.61%	12.7%

performance without long hash codes.

In the training procedure of CH, DCH, and BIQH, the weights of correctly hashed images pairs are set to be 0 so that these image pairs will not be trained for the following hash tables. Thus, the number of image pairs utilized for the training of following hash tables will decrease rapidly. In BDCHR, we increase the weight of wrongly hashed image pairs and decrease the weight of correctly hashed image pairs, which is more reasonable and avoids the amount reduction of image pairs used for training following hash tables. Thus, BDCHR achieves better retrieval performance than existing multi-hashing methods.

MAP scores for hashing methods with different code lengths on MNIST, CIFAR-10, USPS, and NUSWIDE are shown in Tables 1, 2, 3, and 4, respectively. Compared with unsupervised and semi-supervised methods, BDCHR achieves the highest MAP scores on four databases with different hash code lengths. Similar to the results of Recall-and-precision curves, the

Table 3: MAP scores of BDCHR and comparative methods with different hash code lengths on the USPS

	16 bits	24 bits	32 bits	48 bits	64 bits
BDCHR	73.31%	74.61%	75.3%	75.91%	75.96%
COSDISH	78.91%	78.37%	78.91%	81.32%	80.02%
BBSHR	60.21%	64.23%	66.34%	67.34%	68.34%
BSPLH	53.45%	56.34%	57.45%	58.45%	60.24%
BIQH	54.75%	56.87%	58.45%	59.03%	61.23%
DCH	56.83%	51.18%	54.97%	56.59%	59.63%
CH	30.34%	33.21%	41.23%	43.23%	45.32%
SPLH	51.83%	55.27%	55.46%	55.71%	56.69%
LSH	28.97%	32.77%	40.36%	40.47%	44.6%

Table 4: MAP scores of BDCHR and comparative methods with different hash code lengths on the NUSWIDE

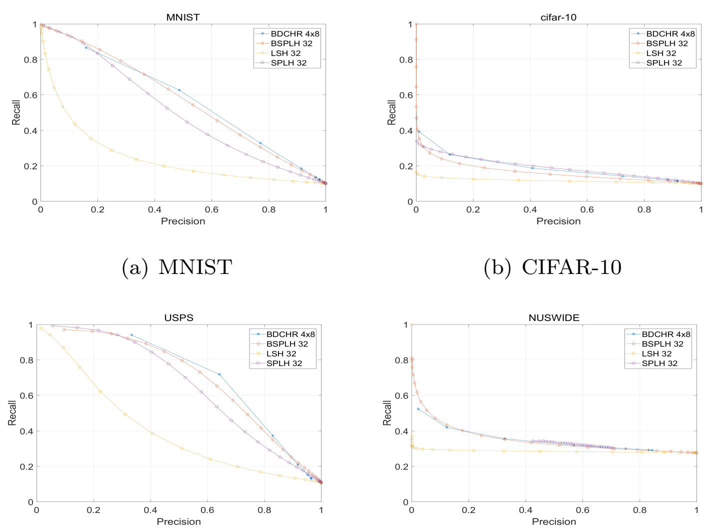
	16 bits	24 bits	32 bits	48 bits	64 bits
BDCHR	39.52%	39.92%	40.02%	40.1%	40.1%
COSDISH	30.21%	30.05%	30.53%	30.66%	30.32%
BBSHR	38.52%	39.33%	39.66%	39.89%	39.91%
BSPLH	32.75%	33.51%	33.94%	34.3%	34.47%
BIQH	31.71%	31.79%	31.81%	32.01%	32.11%
DCH	38.68%	38.7%	38.86%	38.84%	38.69%
CH	32.6%	32.45%	32.05%	31.41%	31.14%
SPLH	32.78%	33.27%	33.48%	34.79%	35.02%
LSH	28.42%	28.8%	28.62%	29.12%	28.92%

performance of BDCHR is the second highest on USPS dataset which is just worse than the supervised COSDISH method, and the highest on other three datasets. This phenomenon indicates that supervised hashing methods generally require a lot of supervised information for training to achieve satisfying performance. When the supervised information is not enough and cannot represent the semantic similarity information of datasets, semi-supervised hashing methods based on both labeled and unlabeled data are more suitable for image retrieval task.

4.2. Comparison Under the Same Storage Cost of Hash Codes

In this paper, to further validate the efficiency of the proposed method, BDCHR is also compared with representative semi-supervised single-table based hashing methods, i.e. SPLH and BSPLH, under the same storage cost of hash codes. The unsupervised LSH method is also used as the baseline method. BDCHR is compared with single-table based hashing methods using the same hash bits in total, i.e. BDCHR (4 tables with 8 bits per table) versus single-table based hash methods (LSH, SPLH, and BSPLH using 32 bits), and BDCHR (4 table with 16 bits per table) versus single-table hash methods (LSH, SPLH, and BSPLH using 64 bits). Experimental results are shown in Figures 6 and 7, respectively.

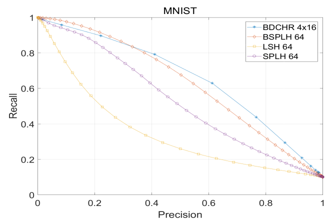
According to Figures 6 and 7, unsupervised LSH which generates hash functions randomly yields the worst performance. Semi-supervised hashing methods, i.e. BSPLH and SPLH, achieves better performance than LSH, because both data distribution information of unlabeled data and semantic similarity information of labeled data are utilized for training. The proposed BDCHR method trains hash functions by correcting errors caused by all the previous hash functions in a hash table and changes the updating rule of weight matrix for the training of new hash table. Comparing to single-table based hashing methods, BDCHR yields a better retrieval performance under the same storage cost of hash codes.



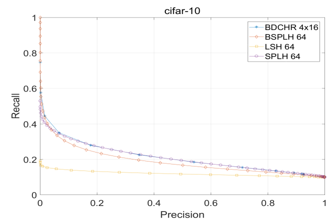
(a) MNIST (b) CIFAR-10

(c) USPS (d) NUSWIDE

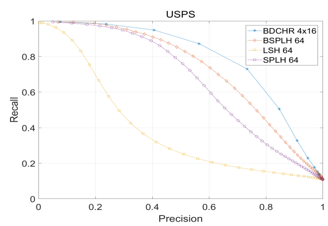
Figure 6: Recall-and-precision curves with 4×8 on the MNIST (a), the CIFAR-10 (b), the USPS (c), and the NUSWIDE (d).



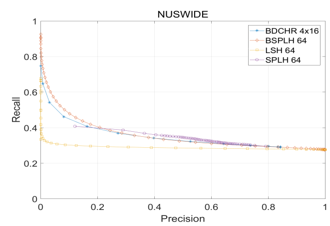
(a) MNIST



(b) CIFAR-10



(c) USPS



(d) NUSWIDE

Figure 7: Recall-and-precision curves with 4×16 on the MNIST (a), the CIFAR-10 (b), the USPS (c), and the NUSWIDE (d).

4.3. Parameters Selection

The Area Under the Curve (AUC) [46] is used to measure the retrieval performance of BDCHR with different values of parameters. The AUC is computed based on the recall-and-precision curve as follows:

$$AUC = \int precision \; d(recall) \tag{20}$$

BDCHR is trained in double loop. In the inner loop, BDCHR trains the hash functions of a hash table using BSPLH, and parameters α, β, λ can be set as the same way in [4]. δ controls the threshold of similar and dissimilar images using the hamming distances, which will affect the judge of error mapping of the previous hash tables. δ is finally set to be $round(K/4)$ in our experiments where $round(\bullet)$ is the rounding function.

BDCHR firstly returns an image set X_R using the trained multiple hashing tables and then uses the semi-supervised re-ranking technique to return the final retrieval images which is a subset of X_R. A parameter *ratio*, is used which controls the number of images in X_R being returned by each hash table. For example, *ratio*$= 0.4$ means return 40% of X_R as the final retrieval result. All the images in X_R will be ranked using the query-adaptive weighted Hamming distance. The Figure 8 shows AUC performance varies with different *ratio* values. This experiment is performed on the MNIST database with 5 hash tables and 32 bits per table. According to the Figure 8, BDCHR achieves best performance when *ratio*$= 0.8$. Therefore, we set *ratio*$= 0.8$ for BDCHR in all experiments.

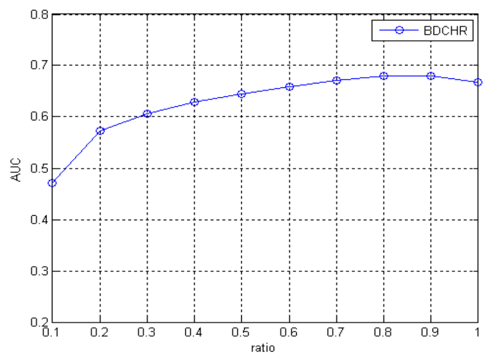


Figure 8: AUC varies with different values of *ratio* on MNIST of BDCHR with 32 bits.

The number of hash table m also has an influence on the performance of BDCHR. The Figure 10 shows the AUC values of BDCHR with variable number of hash tables while the number of bits per table is set to 32. This figure shows that more hash tables will lead to better AUC performances of BDCHR. Considering both the performance and the memory cost, the value of m is finally is set to 5 while the performance trends to remain unchanged when $m > 5$.

The value of c controls the updating step of matrix S. The experiment is done in MNIST with 5 tables and 32 bits per table which are showed in the Figure 10(a). The values of c have not significant effect on the performance of BDCHR, and BDCHR gets the best performance when $c = 9$. Thus, the parameter c is set to be 9 in all experiments.

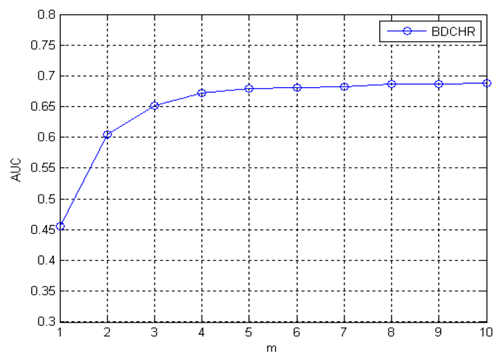


Figure 9: AUC varies with m on MNIST of BDCHR with 32 bits.

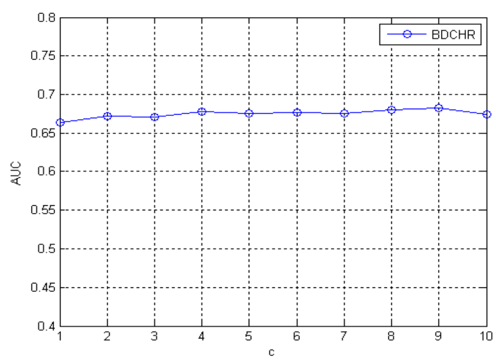


Figure 10: AUC varies with c on the MNIST of BDCHR with 32 bits.

5. Conclusion

A semi-supervised multi-hashing method for image retrieval, i.e. BDCHR, is proposed in this paper. BDCHR trains both hash functions and hash tables in the boosting manner. In one hash table, each hash function is trained by correcting the errors caused by its previous ones. To train the next hash table, the similarity matrix is updated by increasing the weight of wrongly hashed image pairs instead of ignoring the correctly hashed image pairs. In this way, the number of image pairs utilized for the training of following hash tables will not be reduced. Moreover, a semi-supervised re-ranking method is also introduced in BDCHR to further improve its retrieval performance.

Experimental results on four real world image datasets show that BDCHR outperforms other comparative hashing methods, even with same storage cost of hash codes.

In this paper, the proposed BDCHR method attempts to handle the image retrieval task in stationary data environments. However, the data environment in real world is always non-stationary with new data appearing sequentially. Thus, an important future work is to extend BDCHR to the non-stationary data environment by employing complementary multiple hash tables updated based on newly appearing data. Moreover, with the development of deep hashing methods which train hash functions based on high-level features of images, this will be an interesting future work to apply deep learning techniques to further strengthen the performance of BDCHR.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grants 61572201, 61272201, and 61876066, the Guangzhou Science and Technology Plan Project (201804010245), EU Horizon 2020 Programme (700381, ASGARD).

References

[1] C. Yan, L. Li, C. Zhang, B. Liu, Y. Zhang, Q. Dai, Cross-modality bridging and knowledge transferring for image understanding, IEEE Transactions on Multimedia (2019) Early Access.

[2] C. Yan, H. Xie, J. Chen, Z. Zha, X. Hao, Y. Zhang, Q. Dai, A fast uyghur text detector for complex background images, IEEE Transactions on Multimedia 20 (12) (2018) 3389–3398.

[3] C. Yan, Y. Tu, X. Wang, Y. Zhang, X. Hao, Y. Zhang, , Q. Dai, Stat: Spatial-temporal attention mechanism for video captioning, IEEE Transactions on Multimedia (2019) Early Access.

[4] C. Wu, J. Zhu, D. Cai, C. Chen, J. Bu, Semi-supervised nonlinear hashing using bootstrap sequential projection learning, IEEE Transactions on Knowledge and Data Engineering 25 (6) (2013) 1380–1393.

[5] J. Zhao, Research on content-based multimedia information retrieval, International Conference on Computational and Information Sciences (2011) 261–263.

[6] T. Kato, Cognitive view mechanism for content–based multimedia information retrieval, Interfaces to Database Systems (1993) 244–262.

[7] G. T. Zhou, M. T. Kai, F. T. Liu, Y. Yin, Relevance feature mapping for content–based multimedia information retrieval, Pattern Recognition 45 (4) (2012) 1707–1720.

[8] C. S. Tong, M. Wong, Adaptive approximate nearest neighbor search for fractal image compression, IEEE Transactions on Image Processing 11 (6) (2002) 605–615.

[9] M. Casey, M. Slaney, Song intersection by approximate nearest neighbor search, Proc.int.symp.on Music Information Retrieval 6 (2006) 144–149.

[10] C. Silpa-Anan, R. Hartley, Optimised kd-trees for fast image descriptor matching, IEEE International Conference on Computer Vision and Pattern Recognition (2008) 1–8.

[11] A. Beygelzimer, S. Kakade, J. Langford, Cover trees for nearest neighbor, IEEE International Conference on Machine Learning (2006) 97–104.

[12] G. Yunchao, L. Svetlana, G. Albert, P. Florent, Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (12) (2013) 2916–2929.

[13] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, Neural Information Processing Systems (NIPS) 21 (2008) 1753–1760.

[14] J. Wang, S. Kumar, , S.-F. Chang, Semi-supervised hashing for large-scale search, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (12) (2012) 2393–2406.

[15] P. Li, J. Cheng, H. Lu, Hashing with dual complementary projection learning for fast image retrieval, Neurocomputing 120 (10) (2013) 83–89.

[16] H. Fu, X. Kong, J. Lu, Large-scale image retrieval based on boosting iterative quantization hashing with query-adaptive reranking, Neurocomputing 122 (2013) 480–489.

[17] H. Xu, J. Wang, Z. Li, G. Zeng, Complementary hashing for approximate nearest neighbor search, IEEE International Conference on Computer Vision (2011) 1631–1638.

[18] W. W. Y. Ng, X. Zhou, X. Tian, X. Wang, D. S. Yeung, Bagging-boosting-based semi-supervised multi-hashing with query-adaptive re-ranking, Neurocomputing 275 (2018) 916–923.

[19] P. Li, M. Wang, J. Cheng, C. Xu, H. Lu, Spectral hashing with semantically consistent graph for image indexing, IEEE Transactions on Multimedia 15 (1) (2013) 141–152.

[20] J. Shaoa, F. Wu, C. Ouyang, X. Zhang, Sparse spectral hashing, Pattern Recognition Letters 33 (3) (2012) 271–277.

[21] J. Wang, T. Zhang, J. Song, N. Sebe, , H. T. Shen, A survey on learning to hash, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (4) (2018) 769–790.

[22] M. Datar, N. Immorlica, P. Indyk, V. S. Mirrokni, Locality-sensitive hashing scheme based on p–stable distributions, The Twentieth Annual Symposium on Computational Geometry (2004) 253–262.

[23] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, VLDB 99 (6) (1999) 518–529.

[24] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, 47th Annual IEEE Symposium on Foundations of Computer Science (2006) 459–468.

[25] Y. Matsushita, T. Wada, Principal component hashing: an accelerated approximate nearest neighbor search, Image and Video Technology (2009) 374–385.

[26] K. L. Du, M. N. S. Swamy, Principal component analysis, Neural Networks and Statistical Learnin (2014) 355–405.

[27] M. Raginsky, S. Lazebnik, Locality-sensitive binary codes from shift-invariant kernels, Neural Information Processing Systems (NIPS) 22 (2009) 2130–2137.

[28] Y. Lv, W. W. Y. Ng, Z. Zeng, D. S. Yeung, P. P. Chan, Asymmetric cyclical hashing for large scale image retrieval, IEEE Transactions on Multimedia 17 (8) (2015) 1225–1235.

[29] W. W. Y. Ng, Y. Lv, D. S. Yeung, P. P. K. Chan, Two-phase mapping hashing, Neurocomputing 151 (3) (2015) 1423–1429.

[30] L. Chen, D. Xu, I. W.-H. Tsang, X. Li, Spectral embedded hashing for scalable image retrieval, IEEE Transactions on Cybernetics 44 (7) (2014) 1180–1190.

[31] H. Liu, R. Ji, J. Wang, , C. Shen, Ordinal constraint binary coding for approximate nearest neighbor search, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (4) (2018) 941–955.

[32] S. Wang, C. Li, , H. L. Shen, Distributed graph hashing, IEEE Transactions on Cybernetics 49 (5) (2019) 1896–1908.

[33] R. Ye, X. Li, Compact structure hashing via sparse and similarity preserving embedding, IEEE Transactions on Cybernetics 46 (3) (2016) 718–729.

[34] C. Strecha, A. M. Bronstein, M. M. Bronstein, P. Fua, Ldahash: improved matching with smaller descriptors, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (1) (2012) 66–78.

[35] F. Shen, C. Shen, W. Liu, H. T. Shen, Supervised discrete hashing, IEEE International Conference on Computer Vision and Pattern Recognition (2015) 37–45.

[36] J. Gui, T. Liu, Z. Sun, D. Tao, T. Tan, Supervised discrete hashing with relaxation, IEEE Transactions on Neural Networks and Learning Systems 29 (3) (2018) 608–617.

[37] J. Gui, T. Liu, Z. Sun, D. Tao, T. Tan, Fast supervised discrete hashing, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (2) (2018) 490–496.

[38] W. C. Kang, W. J. Li, Z. H. Zhou, Column sampling based discrete supervised hashing, Thirtieth AAAI conference on Artificial Intelligence (2016) 1230–1236.

[39] C. Ma, I. W. Tsang, F. Shen, , C. Liu, Error correcting input and output hashing, IEEE Transactions on Cybernetics 49 (3) (2019) 781–791.

[40] A. Sablayrolles, M. Douze, N. Usunier, H. Jgou, How should we evaluate supervised hashing?, IEEE International Conference on Acoustics, Speech and Signal Processing (2017) 1732–1736.

[41] J. Tang, K. Wang, l. Shao, Supervised matrix factorization hashing for cross-modal retrieval, IEEE Transactions on Image Processing 25 (7) (2016) 3157–3166.

[42] D. Wang, X. Gao, L. He, B. Yuan, Multimodal discriminative binary embedding for large-scale cross-modal retrieval, IEEE Transactions on Image Processing 25 (10) (2016) 4540–4554.

[43] H. F. Yang, K. Lin, C. S. Chen, Supervised learning of semantics–preserving hash via deep convolutional neural networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (2) (2018) 437–451.

[44] L. Jin, X. Shu, K. Li, Z. Li, G. J. Qi, J. Tang, Deep ordinal hashing with spatial attention, IEEE Transactions on Image Processing 28 (5) (2019) 2173–2186.

[45] C. Zhang, W. S. Zheng, Semi-supervised multi-view discrete hashing for fast image search, IEEE Transactions on Image Processing 26 (6) (2017) 2604–2617.

[46] J. Myerson, L. Green, M. Warusawitharana, Area under the curve as a measure of discounting, Journal of the Experimental Analysis of Behavior 76 (2) (2001) 235–243.