

# Binary code reranking method with weighted hamming distance

Haiyan Fu · Xiangwei Kong · Zhenfan Wang

Received: 31 December 2013 / Revised: 9 April 2014 / Accepted: 8 May 2014 /  
Published online: 16 September 2014  
© Springer Science+Business Media New York 2014

**Abstract** Due to its compact binary codes and efficient search scheme, image hashing method is suitable for large-scale image retrieval. In image hashing methods, Hamming distance is used to measure similarity between two points. For  $K$ -bit binary codes, the Hamming distance is an int and bounded by  $K$ . Therefore, there are many returned images sharing the same Hamming distances with the query. In this paper, we propose two efficient image ranking methods, which are distance weights based reranking method (DWR) and bit importance based reranking method (BIR). DWR method aim to rerank PCA hash codes. DWR averages Euclidean distance of equal hash bits to these bits with different values, so as to obtain the weights of hash codes. BIR method is suitable for all type of binary codes. Firstly, feedback technology is adopted to detect the importance of each binary bit, and then big weights are assigned to important bits and small weights are assigned to minor bits. The advantage of this proposed method is calculation efficiency. Evaluations on two large-scale image data sets demonstrate the efficacy of our methods.

**Keywords** Large-scale image retrieval · Image hashing · Binary code reranking · Distance weights based reranking method · Bit importance based reranking method

## 1 Introduction

In Web2.0 time, with the popularity of camera, mobile phone and iPad, people can take photos and share them on social websites anytime and anywhere. According to the research, 1.8

---

H. Fu · X. Kong (✉) · Z. Wang  
School of Information and Communication Engineering, Dalian University of Technology,  
Dalian, China  
e-mail: kongxwpurdue@gmail.com

H. Fu  
e-mail: fuh@dlut.edu.cn

Z. Wang  
e-mail: wangzhf@dlut.edu.cn

ZB data has been created and duplicated in 2011, 75 % of which is unstructured data including images, videos, and music files. Dealing with such large-scale data, how to quickly and accurately obtain valuable information is essential. In these years, many methods proposed to deal with image retrieval and 3D object retrieval [1, 2]. Gao utilizes both visual and textual information to estimate the relevance of user tagged images [3]. In Ref. [4], a hypergraph analysis method is proposed to address 3D object retrieval by avoiding the estimation of the distance between objects. Wang proposes a diverse relevance ranking scheme that is able to take relevance and diversity into account by exploring the content of social images and their associated tags [5].

When using traditional Nearest Neighbor search methods to deal with large-scale image retrieval, it results in large memory cost and low retrieval speed owing to the curse of dimensionality.

Due to the compact feature representation, small storage cost, and high retrieval speed, image hashing technology has been widely applied to large scale Approximate Nearest Neighbor (ANN) retrieval. Given a dataset, hashing method embeds high-dimensional data to Hamming space, and generates a short binary sequence. Then, the efficient Hamming distance is used to measure the similarity between two binary sequences.

Although image hashing has made great achievement, ranking of search results is not optimal. Since the Hamming distance is discrete and bounded by the code length, in practice, there will be a lot of data points sharing the same Hamming distance to the query and the ranking of these data points is ambiguous, which poses a critical issue for similarity search. For  $K$ -bit binary code, there would be  $C_K^i$  different hash codes sharing Hamming distance  $i$  ( $0 \leq i \leq K$ ) with the query. How to rank the these points?

Previously, the research of hashing techniques has been concentrating on hashing function. Recently, many researchers start to study binary code ranking method. Jiang et al proposed QAIS image reranking method, which retains traditional hash retrieval efficiency, and improves retrieval effect at the same time [6]. Zhang et al proposed QsRank method, which uses the Euclidean distance between image visual features to generate the candidate set, so the time and space complexity is higher and retrieval efficiency is lower [7].

In this paper, we propose two weighted Hamming distance methods to improve the initial Hamming retrieval results. For the PCA-based image hashing methods, we propose a reranking method based on distance weight. By taking account of the relationship between PCA projection and hashing quantization, we learn data-dependent weight to reduce the

In Section 2, classical image hashing methods are summarized, and a binary code ranking method based on bit importance is described in Sections 4, and 5 describes our experiments and Section 6 concludes this paper.

## 2 Image hashing methods

Given a dataset, hashing method generates binary code for each data point and approximates the similarity of two points by the Hamming distance between their binary codes. In image hashing method, learning hashing functions to embed high-dimensional feature to Hamming space is a key step for accuracy retrieval. Based on the style of hashing functions learning, we divide the existing hashing methods into two groups: random hyperplane hashing methods and compact hashing methods.

The representative method of random hyperplane hashing is Locality Sensitive Hashing (LSH) method, which is the first image hashing method. The main idea of LSH is to insert

data points into a hash table such that similar things fall in the same bucket, with high probability [8–11]. In the early definitions of LSH, functions were designed to preserve a given geometric distance, hash functions  $h_k(\cdot)$  of LSH should satisfy:

$$P\{h(x_i) = h(x_j)\} = \text{sim}(x_i, x_j), \quad (1)$$

where  $\text{sim}(x_i, x_j)$  is the similarity function of data  $x_i$  and  $x_j$ , that is to say, the probability that two point collide in one hash table is equal to their similarity. Since hashing functions learning is based on randomized hyperplane mapping, one drawback of LSH is that in order to preserve the locality of the data points, they have to generate long codewords, which need large storage space and high computational cost.

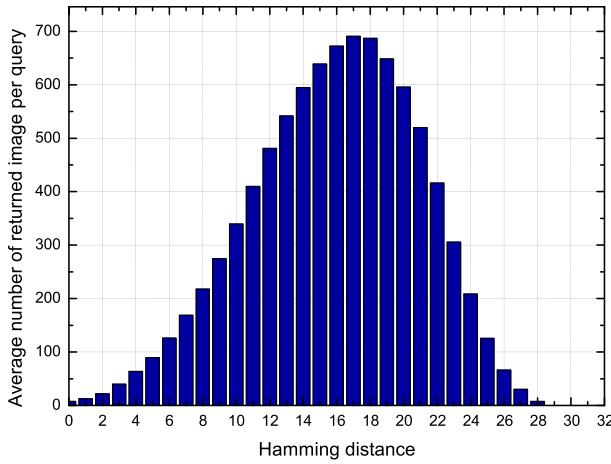
Since the drawback of LSH-like methods, compact hashing methods attract researchers' attention. Compact hashing methods include SH [16], ItQH [13], SSH, CH [14] etc., which can produce compact binary codes that satisfy generic properties effectively. In compact hashing methods, hashing functions are based on three requirements: binary codes should (1) be easily computed for a novel input; (2) be a small number of bits; (3) map similar data points to similar binary codes. Ignoring item (1), hashing functions should be uncorrelated and each hashing function partitions the data points into two balanced parts.

Spectral hashing(SH) [16] generates binary codes by thresholding with nonlinear functions along the principal directions of the data. Wang et al. [17] propose a semi-supervised hashing (SSH)method by utilizing labeled data. In SSH, a pairwise label matrix is defined to express the semantic similarity well. Under the SSH framework, Wang et al. also propose sequential projection learning hashing (SPLH) to correct the errors produced by the previous hashing functions [15]. Zhou et al. propose a balance SSH(BSSH) method by dividing image into several blocks [12]. In the BSSH, the supervised information is completed by combining the similarity of image pairs and label information. Gong et al. [13] formulate the hashing functions learning in terms of directly minimizing the quantization error of mapping the principal component analysis(PCA) projected data to vertices of the binary hypercube. Xu et al. [14] propose complementary hashing(CH) to balance the precision and recall using multiple learned hash tables. Fu et al. adopt Boosting-based method to generate inputs to learn hashing functions, and optimize the hashing functions with a loss function by considering the relationship between samples [18].

Once binary codes are generated using hashing methods, Hamming distance is calculated efficiently between binary codes to measure the similarity of data points. However, since this distance metric is a finite int number, there are hundreds of images sharing the same Hamming distance to a query, it would causes ambiguity for ranking. As shown in Fig. 1, there are hundreds of images sharing the same Hamming distance to a query. How to rank these images with the same Hamming distance? In the next section, the returned images are reranked using weighted Hamming distance.

### 3 Reranking based on distance-based weight

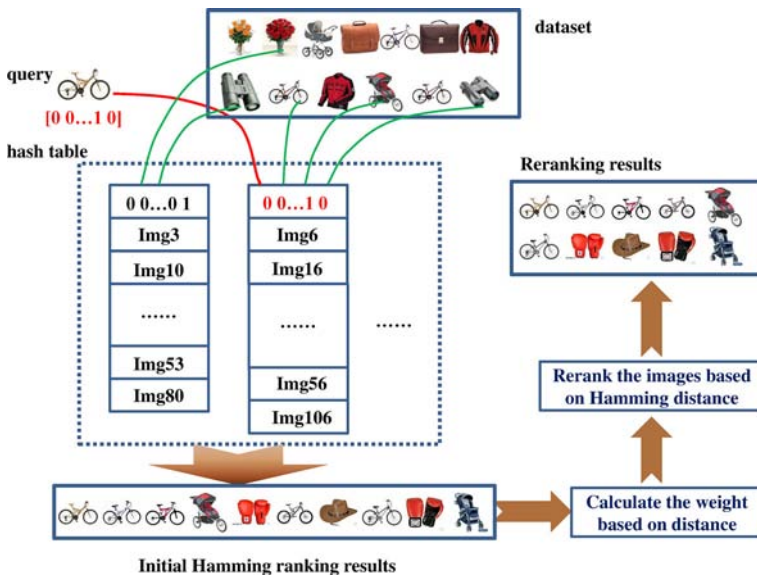
PCA technical is widely used in compact hashing methods, e.g. SH, SSH, ItQH, CH and BH method etc., whose objective functions can be transformed into eigenvectors calculation. In this section, we propose a distance-weighted ranking algorithm (DWR) for these PCA-based hashing methods, as shown in Fig. 2. We firstly analyze the data distribution in different data space, and then learn weight for every binary code. Finally, the initial returned images are reranked based on the weighted Hamming distance.



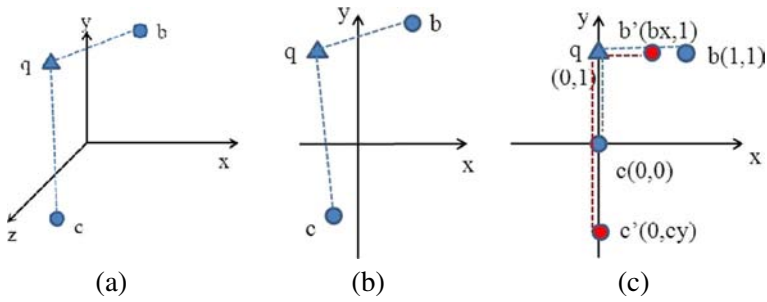
**Fig. 1** The average number of returned images to a query at each Hamming distance on the PI 100 dataset [19], with 32-bit ITQ hash codes

### 3.1 Motivation

Given the query point  $q$  and its two neighborhoods  $b$  and  $c$ . The distribution of them in Euclidean space is shown in Fig. 3a. The Euclidean distance of  $q$  and  $b$  is smaller than  $q$  and  $c$ . Since the PCA projection is orthogonal and preserves the  $L_2$  distance, therefore, the  $\varepsilon$ -neighbors of query remain the same after the PCA projection. That is to say, the distribution of projected data in the PCA space is consistent with Euclidean space, as shown in Fig. 3b. If we quantize the PCA projection



**Fig. 2** Image ranking based on distance-weighted Hamming distance



**Fig. 3** The distribution of query image *q*, image *b* and *c* in different space: **a** In the Euclidean space, the distance of *q* and *b* is small than that of *q* and *c*; **b** In PCA projection space, the  $L_2$  distance is preserved; **c** In the Hamming space, the distance between *q* and *b'* is equal to the distance between *q* and *c'*. When using weighted distance, *b* is moved to *b'*, and *c* is moved to *c'*. As a result, the space structure is maintained

data into binary codes, the **b** and **c** are assigned to the vertex of the hypercube closest to them in terms of Euclidean distance, as shown in Fig. 3c. In the Hamming space, the distance of two points is approximated by Hamming distance. In Fig. 3c, the Hamming distance of **q** and **b** is equal to **q** and **c**. Therefore, the Hamming distance is somewhat ambiguous.

Based on the analysis above, we intend to assign weights to binary codes to keep the trend of Euclidean distance in the Hamming space. With the weights, quantized point **b** and **c** is moved to **b'** and **c'**. In this case, the weighted Hamming distance is more discriminative. Next, we describe how to calculate the weight for every binary bit.

### 3.2 Calculate the weight for binary code

The feature vector of query image **q** before quantization is denoted as  $Pq = [p_1^{(1)}, p_2^{(1)}, \dots, p_K^{(1)}]$ , its *K*-bit binary code is denoted as  $Bq = [b_1^{(1)}, b_2^{(1)}, \dots, b_K^{(1)}]$ . Given any image **I**, its *K*-dimensional feature vector before quantization and *K*-bit binary code after quantization are denoted as  $PI = [p_1^{(2)}, p_2^{(2)}, \dots, p_K^{(2)}]$  and  $BI = [b_1^{(2)}, b_2^{(2)}, \dots, b_K^{(2)}]$ . The Euclidean distance between  $Pq$  and  $PI$  is defined as

$$Ed(Pq, PI) = \|Pq - PI\|_2. \tag{2}$$

From (2), each dimensional of feature contributes to the Euclidean distance.

The Hamming distance between  $Bq$  and  $BI$  is defined as

$$Hd(Bq, BI) = \sum_{i=1}^K (|b_i^{(1)} - b_i^{(2)}|). \tag{3}$$

Different from Euclidean distance, the Hamming distance counts the number of bits with different binary value of two points. Therefore, Hamming distance has no relationship with the amplitude of Euclidean feature. For example, given three PCA projected data  $Pa = [-2, 10, 100, -200]$ ,  $Pb = [10, -2, 10, -2]$  and  $Pc = [100, 100, -20, -20]$ . The binary codes corresponding to them are  $Ba = [0, 1, 1, 0]$ ,  $Bb = [1, 0, 1, 0]$  and  $Bc = [1, 1, 0, 0]$ . The Euclidean distance between **a** and **b** is  $Ed(Pa, Pb) = 288$ , and between **a** and **c** is  $Ed(Pa, Pc) = 24804$ , that is to say  $Ed(Pa, Pb) < Ed(Pa, Pc)$ . However, the Hamming distance  $Hd(Ba, Bb)$  is equal to  $Hd(Ba, Bc)$ . Therefore, we intend to design weight for every binary code, thus the weighted Hamming distance can reflect the difference of Euclidean distance.

**Algorithm 1:** Distance-based weight calculation

**Input:**

Query image  $q$ , its PCA projection feature vector

$Pq = [p_1^{(1)}, p_2^{(1)}, \dots, p_K^{(1)}]$ its binary code

$Bq = \{b_1^{(1)}, \dots, b_K^{(1)}\}$  Any image  $I$ , its PCA projection feature vector

$PI = [p_1^{(2)}, p_2^{(2)}, \dots, p_K^{(2)}]$ its binary code  $BI = [b_1^{(2)}, \dots, b_K^{(2)}]$ .

**Output:**The weight  $\{w_i\} = \{w_1, \dots, w_K\}$  of  $I$  corresponding to  $BI$ .

- (1) Let  $S = \{S_1, S_2, \dots, S_m\}$  be the bits with same binary codes in  $Bq$  and  $BI$ .
- (2) Let  $D = \{D_1, D_2, \dots, D_n\}$  be the bits with different binary codes in  $Bq$  and  $BI$ .
- (3) Calculate constant parameter  $C$  in (6).
- (4) Calculate the weight  $\omega$  using (4).

Let  $S = \{s_1, s_2, \dots, s_m\}$  be the bits where  $b_{s_i}^{(1)} = b_{s_i}^{(2)}$ , and  $D = \{d_1, d_2, \dots, d_n\}$  be the bits where  $b_{d_i}^{(1)} \neq b_{d_i}^{(2)}$ , and  $m + n = K$ . Let  $\omega = \{\omega_1, \dots, \omega_K\}$  be the weights corresponding to  $BI$ .

Section 3.2 suberizes the algorithm of distance-based weight calculation. Given a query, the weight  $\omega$  in (4) is corresponding to the initial returned image, and is defined as:

$$w_i = \frac{\|p_i^{(1)} - p_i^{(2)}\|^2}{\sum_{j \in \{S \cup D\}} \|p_j^{(1)}\|^2} + C, i \in D, \tag{4}$$

where  $C$  is a constant, and is calculated as following:

$$C = \frac{1}{n} \frac{1}{\sum_{j \in \{S \cup D\}} \|p_j^{(1)}\|^2} \sum_{i \in S} \|p_i^{(1)} - p_i^{(2)}\|^2. \tag{5}$$

The weighted Hamming distance are calculated with  $\omega$  in (6), and the returned images are reranked based on the weighted Hamming distance.

$$\hat{d}(Hq, HI_i) = \sum_{k=1}^K (w_k |Hq^{(k)} - HI_i^{(k)}|). \tag{6}$$

**4 Reranking based on bit importance weight**

Distance-based weight calculation is suitable for PCA-like hashing method. For LSH-like or others, which don't satisfy the data distribution assumption, DWR result is unsatisfactory. In this section, we propose a reranking method based on bit importance weight which is suitable for all types of hashing methods.

For many hashing methods, for example, LSH, SH, ItQH, etc.,  $K$  hashing functions are independent with each other. Each hashing function generates one bit of binary code. Therefore,  $K$ -bit binary codes are also independent with each other. For these  $K$ -bit binary codes, we don't decide which binary is more important than the other. So, equal weight is assigned to each bit when calculating Hamming distance.

Actually, each hashing functions can be seen as a two-class classifier, the binary code is the label of class. During the classification, parts of images are classified as +1, corresponding to binary code 1; and others are -1, corresponding to binary code 0. For one hashing function, if most of similar images have the same label with the query, then this hashing

function is more efficient. Therefore, we may assign large weight to the binary code corresponding to this hashing function. In the next sections, we describe how to determine the importance of every bit of binary code.

### 4.1 Motivation

In this section, we compare the binary code of similar returned images and query image to determine which bit is more important. Table 1 shows an example about  $K$ -bit binary code of query image  $q$  and other five images  $\{I_i\}_{i=1}^5$ , whose Hamming distances with query are all equal to 2.  $NS$  is the number of images that have the same binary code with query in the given bit, and  $\omega = \{\omega_1, \dots, \omega_K\}$  is the weight of query.

From Table 1, We can find that in some bits more images share same hash code value with query image, e.g. the  $b4$  bit, but in some bits less images share same hash value with query, e.g. the  $b1$  bit. Since  $NS = 4$  in  $b4$  bit,  $NS = 3$  in  $b3$  bit,  $NS = 2$  in  $b2$  bit, and  $NS = 1$  in  $b1$  bit, so we think that  $b4$  bit is more important than  $b3$ ,  $b2$  and  $b1$ . Therefore,  $\omega_4 > \omega_3 > \omega_2 > \omega_1$ .

Ranking method based on bit importance is showed in Fig. 4. For the query image, feedback technique is applied to find similar images from the returned images. Firstly, we search image dataset based on Hamming distance and return images with small Hamming distance to the query. Then we choose images that are really similar to query, these images are used to design weight of hash code. In the next section, we describe the how to assign weight for every hashing bit.

### 4.2 Calculate the weight

Let query image be  $q$ , its top  $m$  similar images are selected using feedback technique according to the Hamming distance. These  $m(m \ll N)$  images can be shown in the concentric based on their Hamming distance, and the center of concentric are query image and images with Hamming distance equal to 0. The binary codes of  $m$  images are  $H = \{H_1, \dots, H_m\}$ , where  $H_i = \{H_i^{(1)}, H_i^{(2)}, \dots, H_i^{(K)}\}$ , the binary code of query image is  $H_q = \{H_q^{(1)}, \dots, H_q^{(K)}\}$  The weight of query image’s  $K$ -bit hash code is  $\omega = \{\omega_1, \dots, \omega_K\}$ , whose initial value is  $\omega_k = 1$ .

In this section, we compare  $H_i$  and  $H_q$  bit-by-bit. For the binary code of  $k^{th}$  bit  $H_i^{(k)}$ , if  $H_i^{(k)} = H_q^{(k)}$ , then  $\omega_k$  is increased, otherwise is decreased. We update  $\omega$  interactively, the iterative times are  $m$ . For the  $j$  iteration, the expression of  $\omega_i$  is as following:

$$\omega_k = \begin{cases} e\omega_k & \text{if } Hq^{(k)} \neq H_i^{(k)} \\ (1 + e)\omega_k & \text{otherwise} \end{cases}, 0 < e < 1, \tag{7}$$

**Table 1** Master codes and minor codes

bit	q	$I_1$	$I_2$	$I_3$	$I_4$	$I_5$	$NS$	$\omega$
b1	0	1	1	1	0	1	1	$\omega_1$
b2	1	0	1	0	0	1	2	$\omega_2$
b3	1	1	0	1	1	0	3	$\omega_3$
b4	0	0	0	0	1	0	4	$\omega_4$

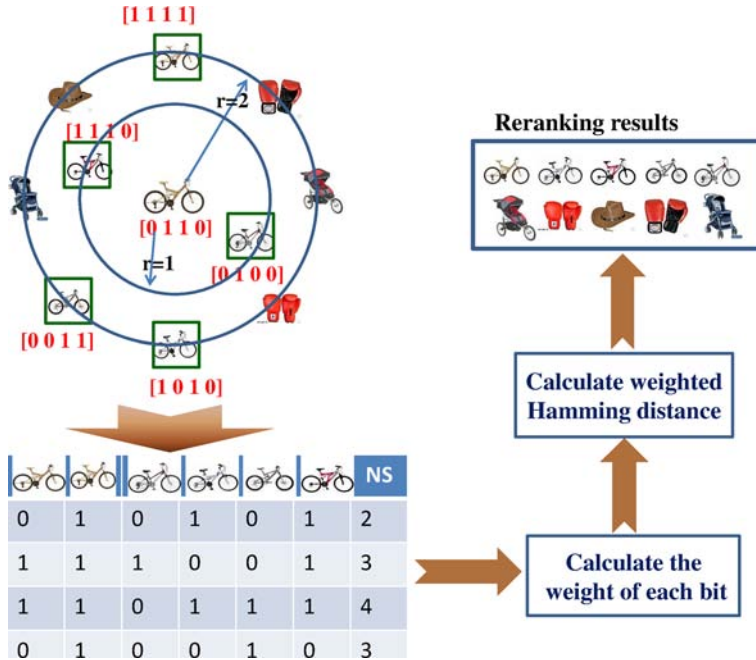


Fig. 4 Image reranking based on relevance weighted method

where  $\varepsilon$  is a parameter, whose value will be discussed in experiment section. The  $\omega$  is to wide the distance between important bits and minor bits. After  $m$  iterations, we obtain the weight  $\omega$  and use it to compute the weighted Hamming distance, and then sort the weighted Hamming distance to rerank the returned images in dataset. The calculation of weight based on bit importance is summarized in Algorithm 2.

**Algorithm 2:** Bit importance based weight calculation

**Input:**

The binary code of query  $Hq = \{hq^{(1)}, \dots, hq^{(K)}\}$ ; The top  $m$  similar images with query based on initial Hamming distance. The binary code of the  $i^{th}$  returned image is  $H_i = \{H_i^{(1)}, \dots, H_i^{(K)}\}$ ;

Parameter  $\varepsilon$ ;

**Output:** The weight  $\{\omega\} = \{\omega_1, \dots, \omega_K\}$  of  $Hq$ .

- (1) Initial the weight  $\{\omega_k\} = \{1, \dots, 1\}$
- (2) for  $i=1:m$
- (3) for  $k=1:K$
- (4) Whether the  $k^{th}$ -bit binary code of  $Hq$  and  $H_i$  is same?
  - if yes,  $\omega_k = (1 + \varepsilon)\omega_k$
  - otherwise,  $\omega_k = \varepsilon\omega_k$
- (5) end for
- (6) end for



## 5 Experiment results

To evaluate the performance of proposed methods, we conduct experiments on two widely used datasets : CIFAR10 and MNIST. To get the best results, we firstly conduct experiments to select the optimal parameters for each method. Then with the optimal parameters, we compare our methods with some art-of-state methods quantitatively and qualitatively.

### 5.1 Experimental setup

**Datasets:** CIFAR10 is an international public image dataset, which includes 60000 images of size of  $32 \times 32$ . These images contain ten categories with different natural scenery or objects. We randomly select 100 samples per category, 1000 images in total as test, and the remaining 59000 images are as training. In experiment, each image is represented as a 320-dimensional GIST feature.

MNIST is a manuscript dataset, which includes 70000 images of ten Arabic numerals. We randomly select 100 samples per category, 1000 images in total as the query and the remaining 69000 images as training. For each image, we use 784-dimensional pixel-value feature to describe the image.

**Baseline methods:** We use the binary code learnt with ITQ hashing method in the experiment. The bit length of the hash code are 16, 32, 48 and 64 respectively. With the hash code, the proposed DWR method and BIR method are compared with two image ranking methods:

1. Directly Hamming Reranking: Rank the images based on initial Hamming distance of the ITQ binary codes.
2. QsRank: The method firstly locates the hashing pool of query image to select the images to be reranked based on the first  $k$  bits of the hash code of query image. Then rerank the selected images based on all bits of the hash code. As for the consistency of experiments, we use all bits of hash code to select the image set to be reranked.

**Evaluation criteria:** We evaluate these methods quantitatively and qualitatively: (1) average precision of top  $n$  reranked images. (2) We show the top 20 images in the reranked results to reflect the performance of different methods visually.

### 5.2 Experimental results on CIFAR10 dataset

To get the best performance, we conduct experiments to determin the optimal parameters for QsRank and BIR. For different bits of hash code, the average reranking precision is showed in Fig. 5, when  $\varepsilon$  in QsRank changes from 0.1 to 1. We find the precision reaches the maximum when  $\varepsilon = 0.2$ . So in the following experiment, we set  $\varepsilon = 0.2$  in QsRank.

There are two parameters in BIR method:  $m$  and  $\varepsilon$ . We select the optimal values according to the reranking results. Figure 6 shows the precision of top 5 reranked images with different bit when  $\varepsilon$  changes between 0.1 and 1,  $m$  with value 10, 20, . . . . ., 60. Figure 6a is the result with 16-bit binary code. As it is shown, when  $m = 20$ , the precision reaches the maximum, and when  $\varepsilon = 0.2$  and  $\varepsilon = 0.3$ , the precision is much better than other  $\varepsilon$  value. Figure 6b and d show the results with 32-bit and 64-bit binary code. As is shown, the precision is the best when  $m = 30$ , and the precision changes little with  $\varepsilon$  changing. Figure 6c shows the case of 48-bit, the precision changes little with  $\varepsilon$  changing. Therefore, we set  $m = 30$  and  $\varepsilon = 0.3$  in BIR.

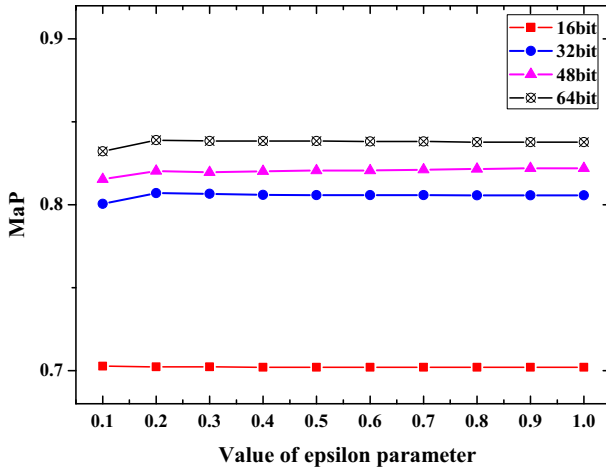


Fig. 5 QsRank method: Average precision for different epsilon

With the selected values for different parameters ( $\epsilon = 0.2$  in QsRank, and  $m = 30, \epsilon = 0.3$  in BIR), we compare these four reranking methods. The top

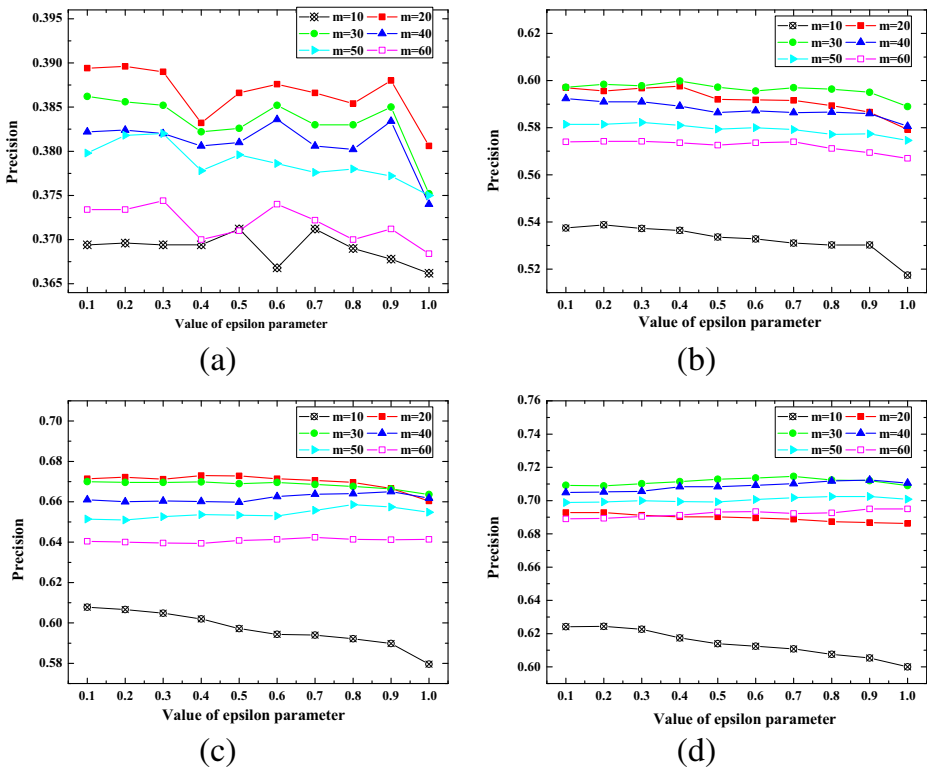
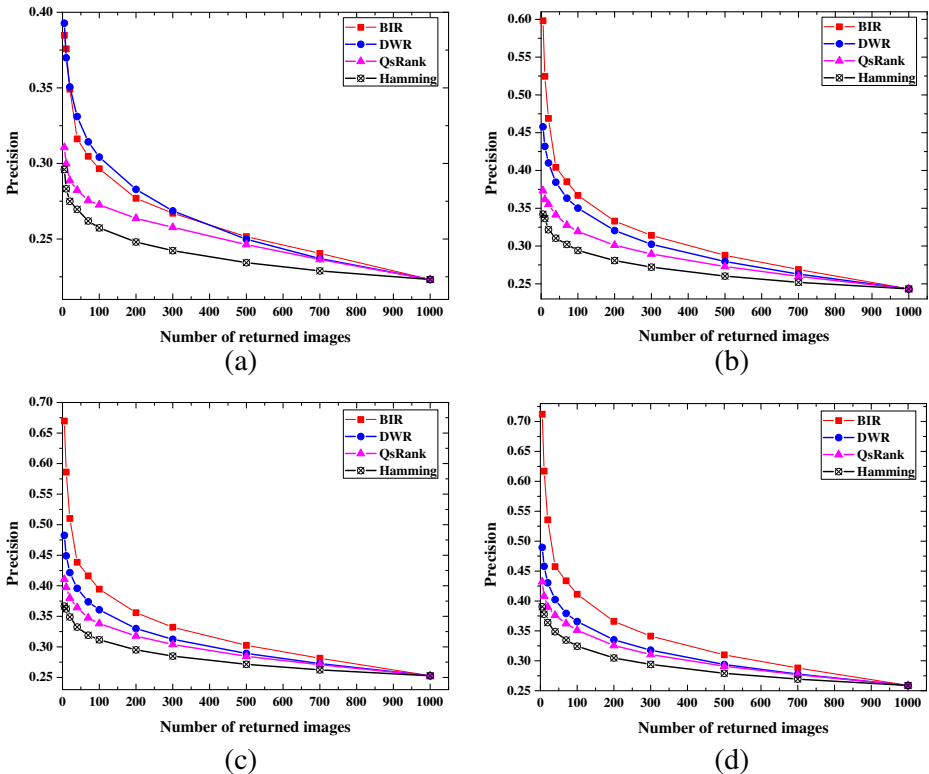


Fig. 6 For CIFAR10 dataset, the reranking precision of BIR method for different hashing bits with different  $m$  and  $\epsilon$ : a 16-bit; b 32-bit; c 48-bit; d 64-bit

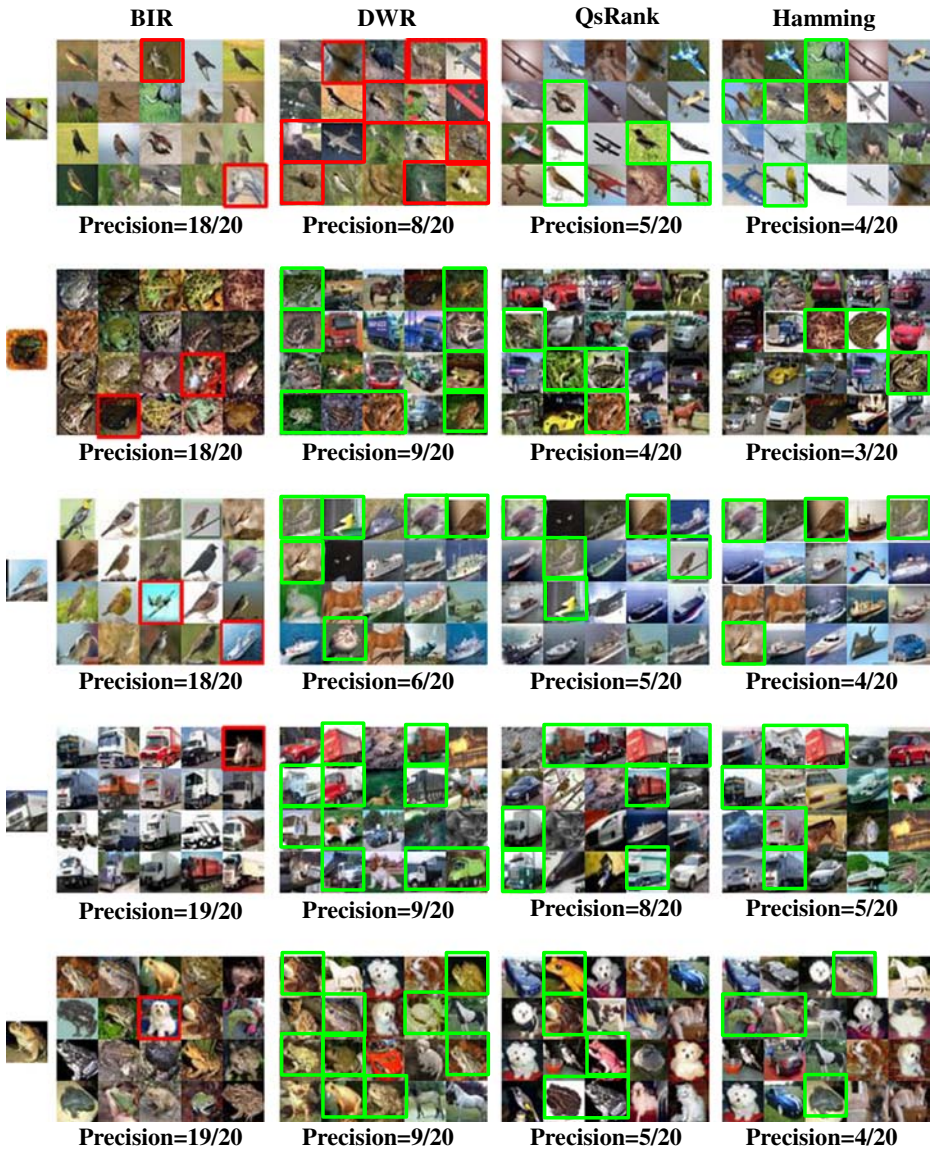


**Fig. 7** For CIFAR10 dataset, the reranking precision of 4 methods for the top 1000 images returned by Hamming distance: **a** 16-bit; **b** 32-bit; **c** 48-bit; **d** 64-bit

1000 images with small Hamming distance is chose to be reranked, and the reranking precision of top  $n$  images are calculated, as shown in Fig. 7. In Fig. 7a, BIR performs better than the other methods with 16-bit binary code. In detail, the precision of top 5 reranked images of BIR are 6 % higher than QsRank, 10 % higher than Directly Hamming Reranking. In Fig. 7b–d, BIR also performs the best, followed by DWR and QsRank. The precision of BIR in the top 5 reranked images with 32-bit is 23 % higher than QsRank and 25 % higher than Directly Hamming Reranking. Besides, BIR methods perperms better with the length of bits increasing. In 48-bit case, precision of BIR is 25 % higher than QsRank. In 64-bit case, BIR has absolute advantage over other methods.

In addition to evaluating quantitatively, we show some reranked results for query images in Fig. 8. The 5 query images come from categories of bird, frog and truck since the space is limited, and the top 20 reranked images are shown. In Fig. 8, the first column is query images, the second to fifth columns are results of BIR, DWR, QsRank and Hamming ranking method.

We sign the false classified images with red box. If there are many wrong classified images, we sign correct images with green box. In CIFAR10 dataset, BIR performs the best than other methods;



**Fig. 8** For CIFAR10 dataset, the reranking results of different methods for 64-bit hashing code

### 5.3 Experiment result of MNIST

The same with CIFAR10 dataset, We first need to set parameters of QsRank method. For the top 1000 returned images based on Hamming distance retrieval, the precision of top 5 reranked images using QsRank method with  $\epsilon = \{0.1, 0.2, 0.3, \dots, 5\}$ . Figure 9 shows the curve between  $\epsilon$  and the precision corresponding to different hash bits. As can be seen from the figure, in some case of a specific bit, when  $\epsilon < 1$ , precision grows fast with  $\epsilon$  increases; when  $\epsilon > 1$ , precision is relatively stable; When  $\epsilon$  is fixed, precision increases with hash

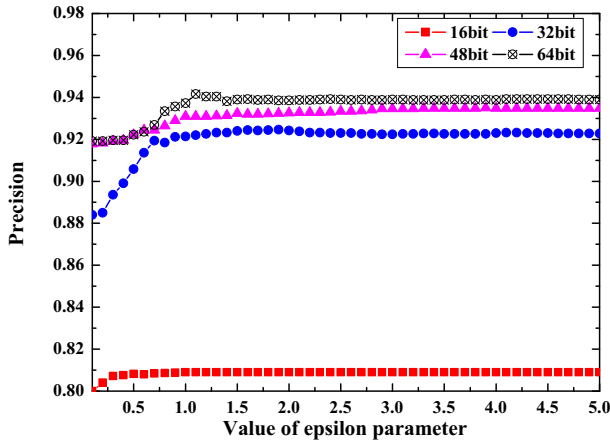
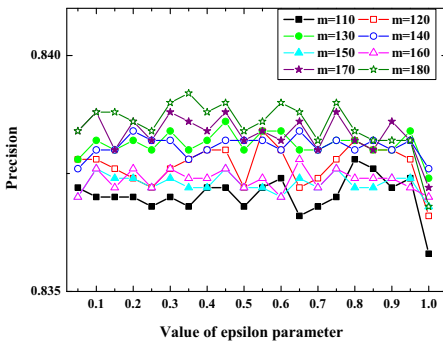
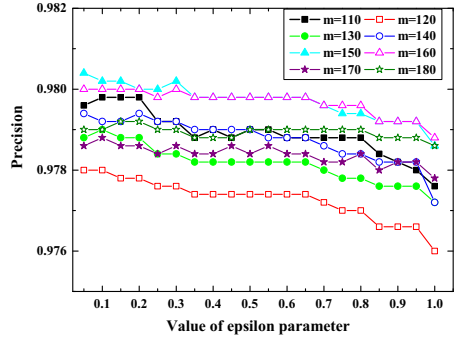


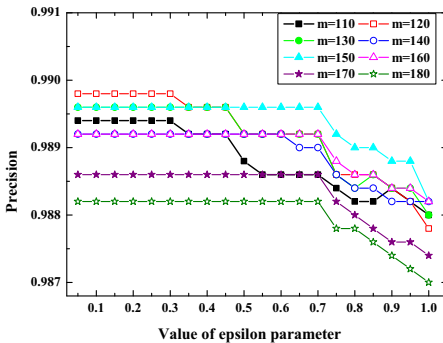
Fig. 9 Average precision of QsRank method with different epsilon in MNIST dataset



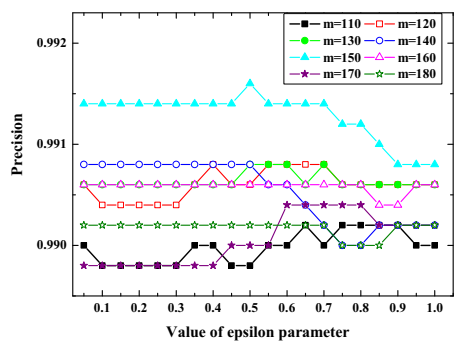
(a)



(b)



(c)



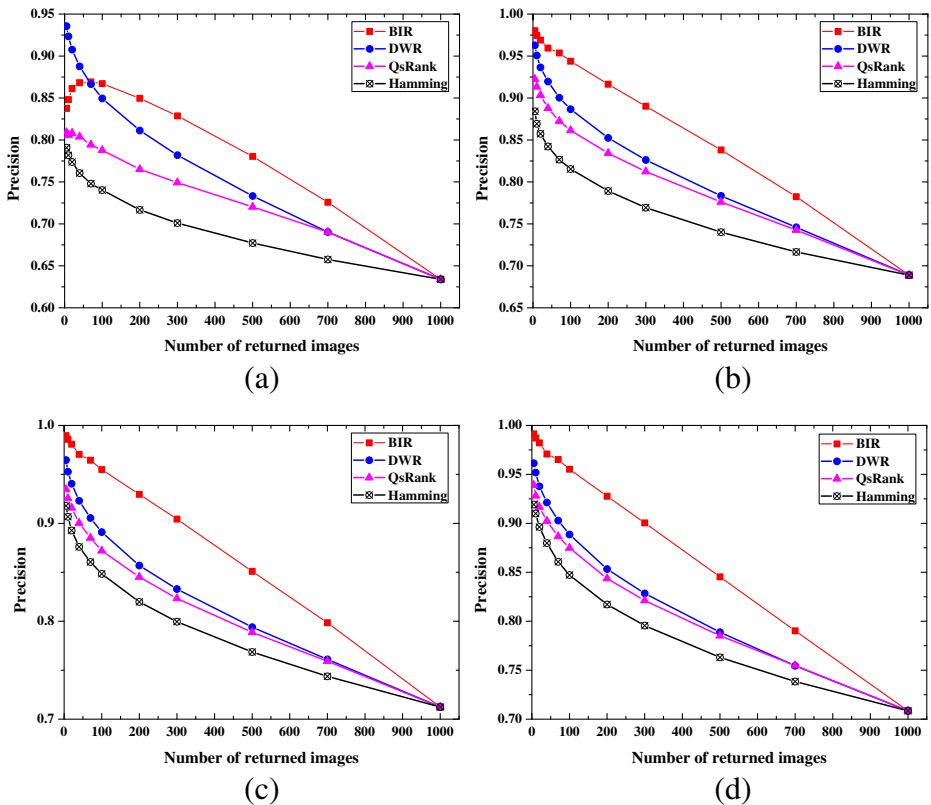
(d)

Fig. 10 For MNIST dataset, reranking precision of BIR method for different hashing bits with different  $m$  and  $\epsilon$ : a 16-bit; b 32-bit; c 48-bit; d 64-bit

bits increase. Especially, precision increases greatly when hash bits are greater than or equal to 32-bits compared with 16-bits. Considering the number of hash bits and the precision of each hash bits, we set  $\epsilon = 3$  in QsRank.

In MNIST dataset, parameters  $m$  and  $\epsilon$  in BIR method are also to be set. For the top 1000 returned images based on Hamming distance retrieval, the precision of top 5 reranked images with different values of  $m$  and  $\epsilon$  is calculated. Figure 10 shows the reranking precision of BIR with 16-bit, 32-bit, 48-bit and 64-bit in MNIST dataset.

In Fig. 10, the range of  $\epsilon$  is in 0.05, 0.1, 0.15, 0.2, ..., 0.95, 1, and the range of  $m$  is in 110, 120, ..., 180. Figure 10a shows the reranking results for the initial search results in 16-bit. The precision of the top 5 images changes little when  $m$  and  $\epsilon$  change. The precision is about 83.75 % with 0.5 % variation. Figure 10b shows the results for 32-bit, the reranking precision decreases with the increase of  $\epsilon$ . When  $\epsilon < 0.7$ , the precision decreases slowly but when  $0.7 < \epsilon < 1$ , it decreases fast. When  $m = \{150, 160\}$ , the precision is the highest. Figure 10c shows the results for 48-bit, the precision changes more smoothly compared with that of 32-bit. When  $m \geq 140$ , the precision changes little if  $\epsilon < 0.7$ , but it decreases with the increase of  $\epsilon$  between 0.7 and 1. When  $m = 150$ , the precision is highest for 48-bit. Figure 10d shows the result for 64-bit, in this case, the range of the reranking precision changes slightly with  $m$  and  $\epsilon$  changing (about 2 %). Based on the above four cases, we set  $m = 150, \epsilon = 0.1$  in the MNIST database Fig. 11.



**Fig. 11** For SIFT1M dataset, the reranking precision of 4 methods for the top 1000 images returned by Hamming distance: **a** 16-bit; **b** 32-bit; **c** 48-bit; **d** 64-bit

With the above parameters, we compare the reranking precision of BIR, DWR, QsRank and Hamming ranking. Figure 11 shows the average reranking precision. From the figures, in 16-bit case, the precision of BIR is the highest, which exceeds QsRank nearly 10 % and exceeds Hamming Ranking about 14 %. For 32-bit, 48-bit and 64-bit, BIR is better than the other three methods. Especially, when the number of returned images is is

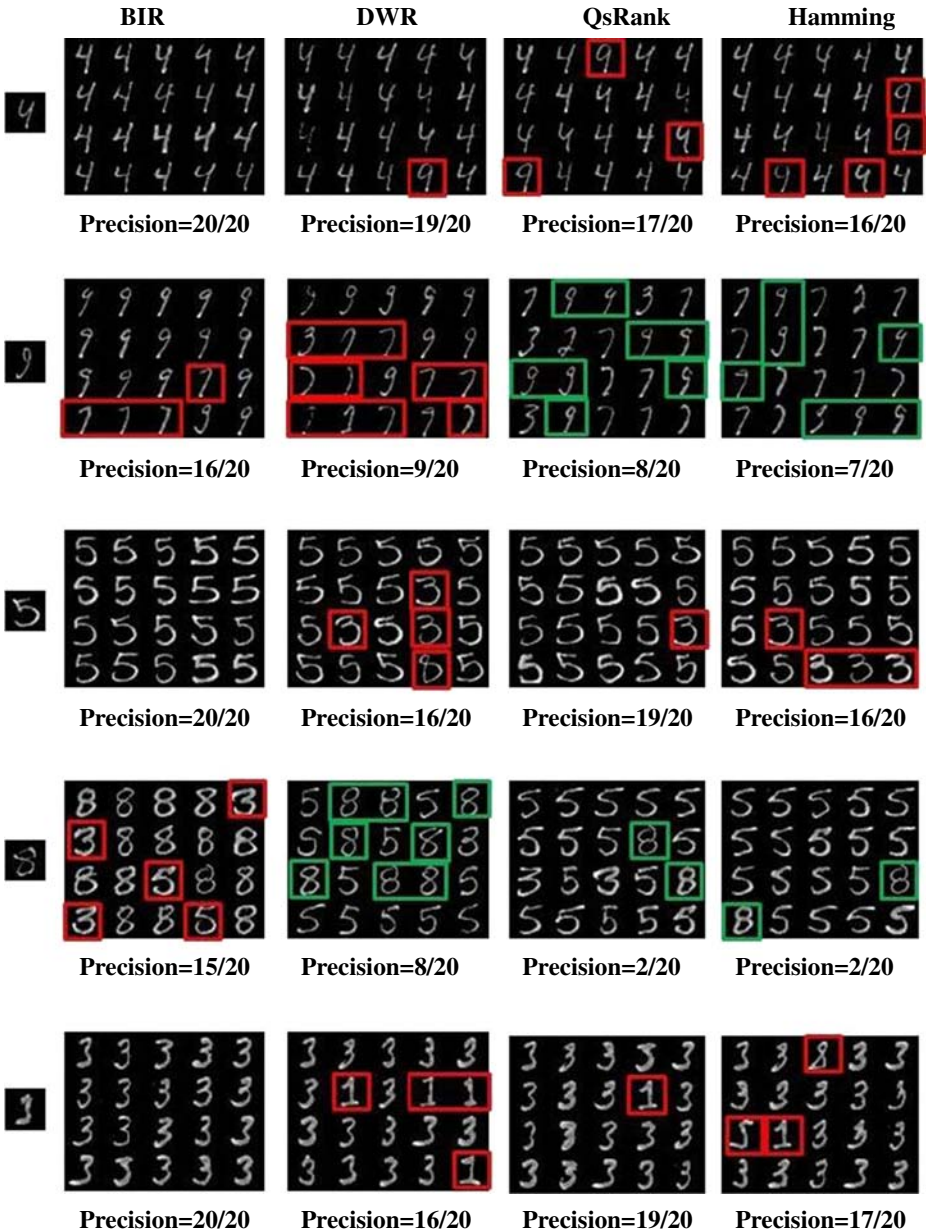


Fig. 12 For MNIST dataset, the reranking results of different methods for 64-bit hashing code

between 100 and 700, BIR method is absolutely ahead of the other methods. For 64-bit binary code, when the number of reranked images is less than 20, the precision of BIR exceeds 99 %.

As with CIFAR10, for part of query images, we apply four methods to rerank the Hamming returned images. The reranking results are showed in Fig. 12. Images in first column belong to digital number 4,9,5,8,3. The next four columns are the top 20 images reranked images of BIR, DWR, QsRank and Hamming methods. The false classified images are signed with red box if the number of false classified images are less. Oppositely, correct classified images are signed with green box.

The figure shows that the BIR, DWR and QsRank method all increase the precision of Hamming distance ranking at some extent. For the query images of digital number 4,5,3, the top 20 images are correct after reranking using BIR method. The precision of Hamming Ranking of these three queries are nearly 80 %. For query images of digital number 9 and 8, the precision is greatly (nearly 80 %) after reranking using BIR method, the precision can also increase by nearly 40 % using QsRank method.

## 6 Conclusion

Although image hashing has made great achievement, traditional Hamming distance performs poorly, since there are many results sharing the same Hamming distance to the query. In this paper, we propose two reranking methods to overcome this problem.

DWR method is suitable to PCA-like hashing methods. The motivation of DWR method is that, the weights are assigned to make weighted Hamming distance approach Euclidean distance. In the BIR method, high weights are assigned to important bits and small weights are assigned to less important bits. The initial returned images based on Hamming distance are as benchmark, we compare bit difference between returned images and query image to determine which bits are more importance. Compared with DWR, the weight of BIR is corresponding to query, and the weight of DWR is corresponding to images in dataset.

In the experiment section, we test the two proposed methods on two large-scale datasets: CIFAR10 and MNIST dataset. In general, BIR performances better than the other three methods in each dataset, DWR performances better than QsRank and Hamming Ranking, Hamming Ranking is the worst method. The experiments also verify the necessity of hash reranking.

**Acknowledgments** The work is partially supported by the Fundamental Research Funds for the Central Universities DUT14QY03.

## References

1. Andoni A, Indyk P (2006) Near-optimal hashing algorithms for near neighbor problem in high dimensions. In: IEEE Symposium on Foundations of Computer Science (FOCS)
2. Charikar M (2002) Similarity estimation techniques from rounding algorithms. In: ACM Symp. on theory of computing
3. Fu H, Kong X, Lu J (2013) Large-scale image retrieval based on boosting iterative quantization hashing with query-adaptive reranking. *Neurocomputing* 122(25):480C489
4. Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Proceedings of the 25th VLDB Conference, Edinburgh, Scotland



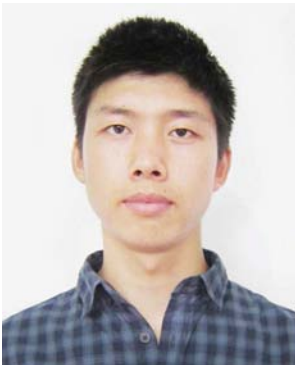
5. Gao Y, Wang M, Zha Z, Tian Q (2011) Less is more: efficient 3-d object retrieval with query view selection. *IEEE Trans Multimed* 13(5):1007–1018
6. Gao Y, Wang M, Tao D, Ji R (2012) 3-D object retrieval and recognition with hypergraph analysis. *IEEE Trans Image Process* 21(9):4290–4303
7. Gao Y, Wang M, Zha Z, Shen J (2013) Visual-textual joint relevance learning for tag-based social image search. *IEEE Trans Image Process* 22(1):363–376
8. Indyk P, Motwani R, 1998 Approximate nearest neighbors: towards removing the curse of dimensionality. In: 30th symposium on theory of computing
9. Gong Y, Lazebnik S (2011) Iterative quantization: A procrustean approach to learning binary codes. In: Proceedings of IEEE conference on computer vision and pattern recognition, Colorado Springs, USA, pp 817–824
10. Jiang Y, Wang J, Chang S (2011) Lost in binarization: query-adaptive ranking for similar image search with compact codes. In: Proceedings of ACM international conference on multimedia retrieval, Trento, Italy
11. Wang M, Yang K, Hua X, Zhang H (2010) Towards a relevant and diverse search of social images. *IEEE Trans Multimed* 12(8):829–842
12. Wang J, Kumar S, Chang S (2010) Sequential projection learning for hashing with compact codes. In: Proceedings of international conference on machine learning, Haifa, Israel, pp 1127–1134
13. Wang J, Kumar S, Chang S (2010) Semi-supervised hashing for scalable image retrieval. In Proceedings of IEEE conference on computer vision and pattern recognition, San Francisco, USA, pp 3424–3431
14. Wang M, Liu B, Hua X (2010) Accessible image search for color blindness. *ACM Trans Intell Syst Technol* 1(1):8:1–26
15. Xie X, Lu L, Jia M, Li H, Seide F, Ma W (2008) Mobile search with multimodal queries. In Proceedings of the IEEE 96(4):589–601
16. Xu H, Wang J, Li Z, Zeng G, Li S, Yu N (2011) Complementary Hashing for Approximate Nearest Neighbor Search. In Proceedings of IEEE International Conference on Computer Vision, Barcelona, Spain, PP 1631-1638
17. Weiss Y, Torralba A, Fergus R (2008) Spectral hashing. In: Proceedings of annual conference on neural information processing systems, Vancouver, Canada, pp 1753–1760
18. Zhang X, Zhang L, Shum HY (2012) Query-sensitive hash code ranking for efficient  $\epsilon$ -neighbor search. In: Proceedings of IEEE conference on computer vision and pattern recognition, providence, RI, USA, pp 2058–2065
19. Zhou J, Fu H, Kong X (2011) A balanced semi-supervised hashing method for CBIR. In: Proceedings of IEEE International Conference on Image Processing, Brussels, Belgium, pp 2481–2484



**Haiyan Fu** received her PhD degree from Dalian University of Technology, China, in 2014. From 2006 to now, she works as a Lecturer in School of Information and Communication Engineering at Dalian University of Technology, China. Her research interests are in the areas of image retrieval and computer vision.



**Xiangwei Kong** received her Ph.D. degree in Management Science and Engineering from Dalian University of Technology, China, in 2003. From 2006 to 2007, she was a visiting researcher in Department of Computer Science at Purdue University, USA. She is currently a professor in the School of Information and Communication Engineering at Dalian University of Technology, China. Her research interests include digital image processing and recognition, multimedia information security, digital media forensics, image retrieval and mining, multi-source information fusion, knowledge management and business intelligence.



**Zhenfan Wang** received his B.S. degree from Dalian University of Technology, China, in 2013. Currently, he is a master student in School of Information and Communication Engineering at Dalian University of Technology, China. His research interests are in the area of image retrieval.