Wang *et al. EURASIP Journal on Image and Video Processing* (2020) 2020:10
https://doi.org/10.1186/s13640-020-00497-4

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Minimal residual ordinal loss hashing with an adaptive optimization mechanism

Zhen Wang[1*], Fuzhen Sun[1], Longbo Zhang[1] and Pingping Liu[2]

Abstract

The binary coding technique has been widely used in approximate nearest neighbors (ANN) search tasks. Traditional hashing algorithms treat binary bits equally, which usually causes an ambiguous ranking. To solve this issue, we propose an innovative bitwise weight method dubbed minimal residual ordinal loss hashing (MROLH). Different from a two-step mechanism, MROLH simultaneously learns binary codes and bitwise weights by a feedback mechanism. When the algorithm converges, the binary codes and bitwise weights can be well adaptive to each other. Furthermore, we establish the ordinal relation preserving constraint based on quartic samples to enhance the power of preserving relative similarity. To decrease the training complexity, we utilize a tensor ordinal graph to represent quartic ordinal relation, and the original objective function is approximated by the one based on triplet samples. In this paper, we also assign different weight values to training samples. During the training procedure, the weight of each data is initialized to the same value, and we iteratively boost the weight of the data whose relative similarity is not well preserved. As a result, we can minimize the residual ordinal loss. Experimental results on three large-scale ANN search benchmark datasets, i.e., SIFT1M, GIST1M, and Cifar10, show that the proposed method MROLH achieves a superior ANN search performance in both the Hamming space and the weighted Hamming space over the sate-of-the-art approaches.

Keywords: Binary codes, Bitwise weights, Ordinal relation preserving, Joint optimization, Minimal residual loss

1 Introduction

The aim of hashing algorithms [1–6] is to learn the binary representations of data which can preserve their original similarity relationship in the Hamming space. Thus, hashing algorithms can retrieve the nearest neighbors of a query data according to Hamming distances. As the advantageous in storage and computation, hashing algorithms have recently been popular in various computer vision and artificial intelligence applications, e.g., image retrieval, object detection, multi-task learning, linear classifier training, and active learning.

We roughly divide existing hashing algorithms into either data-independent hashing or data-dependent ones. The data-independent hashing, such as locality-sensitive hashing (LSH) [7], randomly generates hashing functions, and it typically requires a long binary code or multi-hash tables to achieve satisfying performance. In contrast, the data-dependent hashing algorithms, such as BDMFH [8] and ARE [9], utilize machine learning mechanisms to learn similarity preserving binary codes. Bidirectional discrete matrix factorization hashing (BDMFH) [8] proposes to alternate two mutually promoted processes of learning binary codes from data and recovering data from the binary codes. To enforce the learned binary codes inheriting intrinsic structure from the original data, BDMFH designs an inverse factorization model. Angular reconstructive embeddings (ARE) method [9] learns binary codes by minimizing the reconstruction error between the cosine similarities computed by the original data and the binary embeddings. Usually, the data-dependent hashing can obtain an excellent approximate nearest neighbors (ANN) search performance with compact binary codes. Furthermore, according to the similarity preserving restriction, the data-dependent hashing can be divided

*Correspondence: zhwang@sdut.edu.cn
[1]School of Computer Science and Technology, Shandong University of Technology, Zibo, 255000 China
Full list of author information is available at the end of the article

into the absolute similarity preserving hashing [10, 11] and the relative similarity preserving hashing [6, 12]. The former ones emphasize that the Hamming distances of similar data pairs should be minimal enough, and they are proper for the semantic neighbor search task. The relative similarity preserving hashing demands that the ranking orders of data in different spaces should be consistent with each other. Thus, the relative similarity preserving hashing can achieve a better ANN search performance.

Traditional hashing algorithms treat each binary bit equally, which would cause an ambiguous ranking. For M-bit binary codes, there are C_M^m kinds of data sharing the same Hamming distance m to a query sample. To further explain this phenomenon, we give a simple example as in Fig. 1.

In Fig. 1, $H = \{h_1(x), h_2(x)\}$ represents a set of linear hashing functions, and it separately maps x_1, x_2, and x_3 to a 2-bit binary code. If the importance of each binary bit is considered to be equal, x_2 and x_3 have the same Hamming distance to x_1. As a result, x_2 and x_3 will be simultaneously returned when retrieving the nearest neighbors of x_1 in the Hamming space. However, the similarity degrees of (x_1, x_2) and (x_1, x_3) are different in the Euclidean space. To avoid such an ambiguous situation, the bitwise weight methods are proposed to assign different values to each binary bit. Thus, the similarity degree among the data pairs with the same Hamming distance can be distinguished by the weighted Hamming distances. In Fig. 1, according to the distribution of the query data x_1 and the hashing functions, a larger weight value is assigned to $h_2(x)$. As a result, the weighted Hamming distance of (x_1, x_2) is larger than that of (x_1, x_3). When retrieving the nearest neighbors of x_1, x_3 is firstly returned. As described above, in order to further distinguish the ranking orders of the data with the same Hamming distance to a query data, we should take the importance of bits into consideration. However, the bitwise weight methods, such as QaRank [13, 14], QsRank [15], WhRank [16], and QRank [17], just focus on learning bitwise weights by a two-step mechanism. In this setting, these methods firstly generate binary codes by an existing hashing method (e.g., LSH [7] and ITQ [10]), then generate bitwise weights according to the learnt codes. The two-stage schema causes the learning process of binary codes and bitwise weights to separate with each other, and their performances cannot be iteratively boosted.

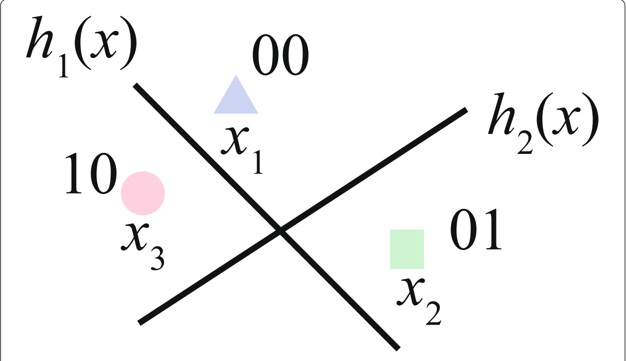

Fig. 1 The hashing functions $H = \{h_1(x), h_2(x)\}$ map the data to 2-bit binary code. Without considering the importance of binary bits, x_2 and x_3 share the same Hamming distance to x_1

In this paper, we propose a novel bitwise weight method dubbed minimal residual ordinal loss hashing (MROLH) and the flowchart is shown in Fig. 2. To enhance the power of preserving relative similarity, we define the ordinal relation preserving objective function based on quartic samples in (a). In (b), we transform the constraint and utilize a tensor ordinal graph to decrease the training time consuming. Unlike most hashing, we simultaneously learn the relative similarity preserving binary codes and bitwise weights with a feedback mechanism by steps (c), (e), and (f). During the iterative training process, we update the weights of the data whose relative similarity is not well preserved by steps (d) and (g), which can minimize the residual performance loss. We compare the proposed MROLH against various state-of-the-art hashing methods on three widely used benchmarks, SIFT1M [18], GIST1M [19], and Cifar10 [20]. Quantitative experiments demonstrate that our algorithm achieves the best ANN search performance in both the Hamming space and the weighted Hamming space.

The main contributions of this paper include:

1. In this paper, both binary codes and bitwise weights are demanded to preserve the original relative similarity of training data, and we establish the similarity preserving constraint based on quartic samples to enhance the power of preserving ordinal relation.

2. To decrease training time complexity, we embed the quartic ordinal relationship into a triplet one and utilize a tensor product graph to approximate the ordinal set.

3. During the iterative training process, we jointly learn binary codes and bitwise weights by a feedback mechanism to make them well adaptive to each other, and fix the problem of residual performance loss by boosting the weights of the data whose ordinal relation is not well preserved.

The rest of this paper is organized as follows: In Section 2, we briefly overview the relative similarity preserving hashing and the bitwise weight methods. Section 3 describes the proposed MROLH with three innovation measures. In Section 4, we show and analyze the comparative experiments on three large datasets. Finally, we conclude this paper in Section 5.

2 Related work

In this paper, we mainly focus on two issues: (a) How to preserve the original ordinal relation in the Hamming

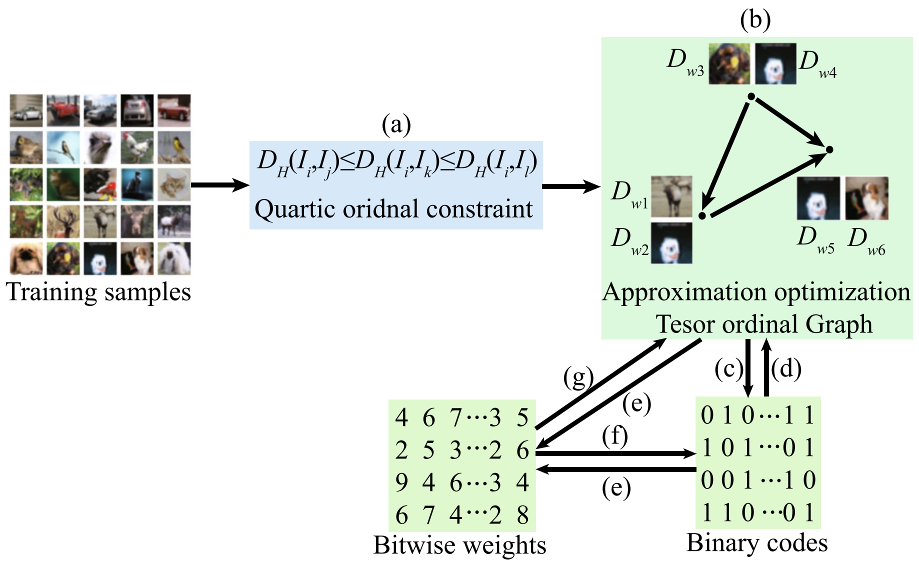

Fig. 2 The flowchart of the minimal residual ordinal loss hashing

space and the weighted Hamming space. (b) How to guarantee bitwise weights and binary codes are well adaptive to each other.

To solve problem (a), we demand binary codes and bitwise weights to preserve the relative similarity. However, almost of the existing relative similarity preserving restrictions are defined based on triplet samples, which has an inferior ANN search performance. Minimal loss hashing [21] defines a hing-like loss to penalize the similar (dissimilar) data pair with a large (small) Hamming distance, and it solves this issue by optimizing the convex-concave upper bound of the objective function using a perception-like learning procedure. Triplet loss hashing [22] and listwise supervision hashing [23] directly demand that the Hamming distance among similar data points should be minimal than that among dissimilar data points. Ordinal preserving hashing (OPH) [12] divides all training data into different clusters, and all cluster centers are involved in computing the performance loss. However, OPH demands the distribution of training samples should be uniform. Ordinal constraint hashing (OCH) [6] aims to minimize retrieval loss by preserving ordinal relations of ranking tuples in the Hamming space. As the number of ranking tuples is quadratic or cubic to the size of the training samples, it is difficult to build ranking tuples efficiently in a large-scale data set. To fix the above problem, OCH embeds in which the original quartic order relation can hold as the triplet order relation.

As Hamming distances are discrete integer values, many data pairs with different binary codes would share the same distance value which causes their relative similarity relationship hard to distinguish. To fix this issue, the bitwise weight methods propose to assign different weight values to each bit. QaRank [13, 14] learns bitwise weights by minimizing the intra-class distance while preserving the inter-class relationship computed based on original training samples. The bitwise weights in QsRank [15] are learned according to the probability of mapping training samples to specified codes, and it is well designed for PCA hashing. WhRank [16] takes the distribution of query samples into consideration, which can effectively distinguish the similarity relationship among data pairs with the same binary codes. The bitwise weights in QRank [17] relates to the discriminate ability of hashing functions and the distribution of query data. Most bitwise weight methods adopt a two-step mechanism, which firstly learns binary codes by a hashing method (such as LSH [7] or ITQ [10]), then generates bitwise weights according to the learnt binary codes. As a result, the retrieval results obtained by weighted Hamming distances cannot further feedback the procedure of learning binary codes, which causes binary codes and bitwise weights not well adaptive to each other as in problem (b).

3 Methods

For $x \in R^d$, we can map it into M-bit binary code $B = \{b_1, \cdots, b_M\}$ by the hash functions $H(x) = \{h_1(x), \cdots, h_M(x)\}$, and the mth bit b_m is calculated as $b_m(x) = \mathrm{sgn}(h_m(x))$. In this paper, $h_m(x)$ is a linear function.

Generally, Hamming distances are utilized to achieve an ANN search task. But, it usually causes an ambiguous ranking order [15–17]. To avoid this embarrassing situation, we learn the bitwise weights $W(x) = \{w_1(x), \cdots, w_M(x)\}$ of data x, and $w_m(x)$ represents the mth bit weight function.

In this paper, to ensure the hashing functions $H(x)$ and bitwise weight functions $W(x)$ have an excellent performance, we propose three innovation measures which are described in Sections 3.1, 3.2, and 3.3.

3.1 The ordinal relation preserving constraint based on quartic samples

As discussed in many previous works [6, 12], both the absolute similarity preserving hashing and the relative similarity preserving hashing based on triplet samples have a poor performance in retrieving approximate nearest neighbors. In contrast, we demand binary codes and bitwise weights should satisfy the ordinal relation preserving constraint defined based on quartic samples as in Eq. (1). It directly maximums the number of the data points whose ordinal relation is well preserved in set C.

$$\max \sum_{(x_i,x_j,x_k,x_l)\in C} I(D_H(x_i,x_j) \leq D_H(x_i,x_k) \leq D_H(x_i,x_l)) \quad (1)$$

(x_i, x_j, x_k, x_l) are the quartic samples which satisfy the ordinal relationship defined in the Euclidean space. $I(\cdot)$ is the judge function. It returns 1, if the condition is satisfied; otherwise, 0 is returned.

For the problem defined in Eq. (1), the primary question is how to construct the ordinal relation preserving set C. Generally, we can establish the set C by collecting similar data pairs and dissimilar ones. However, it is hard to define the similarity relationship. To fix this problem, we adopt a tensor product graph G to represent the ordinal relationship of quartic samples as below:

$$G = S \otimes DS \quad (2)$$

The definition of graph S is shown in Eq. (3), which utilizes the distance value to indicate the similarity relationship.

$$S(i,j) = \begin{cases} 0 & i = j \\ e^{\frac{-\|x_i - x_j\|_2^2}{2\sigma^2}} & \text{otherwise} \end{cases} \quad (3)$$

DS represents the dissimilar graph, and the value of $DS(i,j)$ is computed as in Eq. (4).

$$DS(i,j) = \frac{1}{S(i,j)} \quad (4)$$

$\otimes$ represents the Kronecker product of matrixes, then $G(ij,kl) = S(i,j) \cdot DS(k,l)$. As a result, the value in G can represent the similarity relationship of quartic samples as in Eq. (5).

$$\begin{cases} S(i,j) < S(k,l) & G(ij,kl) > 1 \\ S(i,j) > S(k,l) & G(ij,kl) < 1 \end{cases} \quad (5)$$

As described above, the ordinal relation preserving set C can be constructed according to the tensor ordinal graph G. But, for massive samples, the construction time complexity is relatively higher. So, we further transform the ordinal relation constraint as shown in Eq. (6).

$$\begin{aligned} &\sum_{\forall(x_i \neq x_j,x_k,x_l)} I\left(\left(\|x_i - x_j\|_2^2 - \|x_i - x_k\|_2^2\right)\right. \\ &\left.-\left(\|x_i - x_j\|_2^2 - \|x_i - x_l\|_2^2\right)\right) \\ &= \sum_{\forall(x_i \neq x_j,x_k,x_l)} I\left(\left(x_i^T x_j - x_i^T x_k\right)^2 - \left(x_i^T x_j - x_i^T x_l\right)^2\right) \\ &= \sum_{\forall(x_j,x_k,x_l)} I\left((x_j - x_l)^T M (x_j - x_l) - (x_j - x_k)^T M (x_j - x_k)\right) \end{aligned} \quad (6)$$

where $M = \sum_i x_i^T x_i$ is a positive semi-definite symmetrical matrix. So, it is convenient to use SVD to decompose into $Z \in R^{d_{\text{svd}} \times d}$ such that $M = Z^T \Lambda Z$. Then, a mapping function can be defined as $u_i = Zx_i \in R^{d_{\text{svd}}}$, and the ordinal relation constraint can be written as in Eq. (7).

$$\begin{aligned} &\sum_{\forall(u_j,u_k,u_l)} I\left((u_j - u_l)^T \Lambda (u_j - u_l) - (u_j - u_k)^T \Lambda\right) \\ &\leq \sum_{\forall(u_j,u_k,u_l)} I\left(\left\|\Lambda^{\frac{1}{2}}\right\|_2^2 \cdot \left(\|u_j - u_l\|_2^2 - \|u_j - u_k\|_2^2\right)\right) \\ &\propto \sum_{\forall(u_j,u_k,u_l)} I\left(\|u_j - u_l\|_2^2 - \|u_j - u_k\|_2^2\right) \end{aligned} \quad (7)$$

Finally, the objective function defined in Eq. (8) is utilized to learn binary codes. The set $\hat{C}$ can be easily constructed by selecting the elements whose values are minimal than 1 in G.

$$\min \sum_{(x_i,x_j,x_k)\in \hat{C}} I(D_H(x_i,x_j) \geq D_H(x_i,x_k)) \quad (8)$$

Similarly, the ordinal relation preserving restriction for bitwise weights is re-defined as in Eq. (9).

$$\min \sum_{(x_i,x_j,x_k)\in \hat{C}} I(D_{WH}(x_i,x_j) > D_{WH}(x_i,x_k)) \quad (9)$$

3.2 Minimal residual loss

For traditional algorithms, the weights of samples keep unchanged during training process. As a result, each hash function and bitwise weight just try to minimize the performance loss induced by its own, and the residual loss caused by their former ones are totally ignored. To fix up the above problem, we propose to iteratively boost the weights of the data whose similarity relationship is not well preserved.

Initially, we set the weight of each data as $\frac{1}{n}$ (n is the number of the training samples), and we utilize Eq. (10) to update their weights during the training process.

$$\pi_m^{r+1}(x_i) = \pi_m^r(x_i) \cdot \beta^{1-T(x_i)} \tag{10}$$

$$\beta = \frac{\xi_m^r}{1-\xi_m^r} \tag{11}$$

$\pi_m^r(x_i)$ is the weight of x_i for the mth hash function or bitwise weight function during the rth training procedure. $T(x_i)$ returns 0, if the similarity relationship among x_i and its nearest neighbors is preserved; otherwise, 1 is returned. The definition of ξ_m^r is shown in Eq. (12).

$$\xi_m^r = \sum_{i=1}^{n} \pi_m^r(x_i) T(x_i) \tag{12}$$

After introducing the data weights, we separately redefine the objective function for learning hash functions and bitwise weight functions as in Eqs. (13) and (14). $\pi_M^R(x_i)$ is the weight value of the samples when the algorithm converges.

$$\min \sum_{(x_i,x_j,x_k)\in\hat{C}} \pi_M^R(x_i) \cdot I(D_H(x_i,x_j) \geq D_H(x_i,x_k)) \tag{13}$$

$$\min \sum_{(x_i,x_j,x_k)\in\hat{C}} \pi_M^R(x_i) \cdot I(D_{WH}(x_i,x_j) > D_{WH}(x_i,x_k)) \tag{14}$$

3.3 Joint optimization

To make binary codes and bitwise weights well adaptive to each other, we propose a joint optimization mechanism, and the objective function is defined as in Eq. (15). During the training process, we iteratively optimize the parameters of hash functions and bitwise weight functions.

$$\begin{aligned}\Theta = \min \sum_{(x_i,x_j,x_k)\in\hat{C}} & \pi_M^R(x_i)[I(D_H(x_i,x_j) \geq D_H(x_i,x_k)) \\ & + I(D_{WH}(x_i,x_j) > D_{WH}(x_i,x_k))]\end{aligned} \tag{15}$$

In this paper, the sign function is utilized to generate discrete integer values, which makes the objective function become NP hard problem. To solve this issue, we adopt tanh(·) to approximate sign(·) function. Then, the binary code is re-defined as $B(x_i) = \tanh(V^T x_i)$. Thus, we can separately compute the Hamming distance and the weighted Hamming distance by Eqs. (16) and (17). M is the number of binary bits. $\odot$ is the bitwise product operation.

$$\begin{aligned}D_H(x_i,x_j) &= \frac{1}{2}\left(M - B^T(x_i) \cdot B(x_j)\right) \\ &= \frac{1}{2}\left(M - \sum_{m=1}^{M}\left[\tanh^T\left(v_m^T x_i\right) \cdot \tanh\left(v_m^T x_j\right)\right]\right)\end{aligned} \tag{16}$$

$$\begin{aligned}D_{WH}(x_i,x_j) &= \frac{1}{2}\left(M - (W(x_i)\odot B(x_i))^T \cdot B(x_j)\right) \\ &= \frac{1}{2}\left(M - \sum_{m=1}^{M}\left[w_m(x_i)\cdot \tanh^T\left(v_m^T x_i\right)\right.\right. \\ &\quad \left.\left.\cdot \tanh\left(v_m^T x_j\right)\right]\right)\end{aligned} \tag{17}$$

If we define $\phi(\hat{c})$ and $\phi(\tilde{c})$ as in Eqs. (18) and (19), the objective function can be rewritten as in Eq. (20).

$$\phi(\hat{c}) = \frac{1}{1+\exp(D_H(x_i,x_k) - D_H(x_i,x_j))} \tag{18}$$

$$\phi(\tilde{c}) = \frac{1}{1+\exp(D_{WH}(x_i,x_k) - D_{WH}(x_i,x_j))} \tag{19}$$

$$\Theta = \min \sum_{(x_i,x_j,x_i,x_k)\in\hat{C}} \pi_M^R(x_i)\cdot(\phi(\hat{c}) + \phi(\tilde{c})) \tag{20}$$

When learning the mth hash function during the rth training procedure, the partial derivation of the objective function is shown in Eq. (21).

$$\begin{aligned}\frac{\partial\Theta}{\partial v_m} &= \sum_{\hat{c}\in\hat{C}} \phi(\hat{c})(1-\phi(\hat{c}))\cdot\pi_m^r(x_i) \\ &\cdot\left\{\left[\frac{\partial D_H(x_i,x_k)}{\partial v_m} - \frac{\partial D_H(x_i,x_j)}{\partial v_m}\right]\right. \\ &\left.+\left[\frac{\partial D_{WH}(x_i,x_k)}{\partial v_m} - \frac{\partial D_{WH}(x_i,x_j)}{\partial v_m}\right]\right\}\end{aligned} \tag{21}$$

For the parameter v_m, the partial derivation of the Hamming distance function and the weighted Hamming distance function can be computed as in Eqs. (22) and (23).

$$\begin{aligned}\frac{\partial D_H(x_i,x_j)}{\partial v_m} &= -\frac{1}{2}\left\{x_i\cdot\left[\left(1-\tanh^2\left(v_m^T x_i\right)\right)\cdot\tanh\left(v_m^T x_j\right)\right]^T\right. \\ &\left.+x_j\cdot\left[\left(1-\tanh^2\left(v_m^T x_j\right)\right)\cdot\tanh\left(v_m^T x_i\right)\right]^T\right\}\end{aligned} \tag{22}$$

$$\begin{aligned}\frac{\partial D_{WH}(x_i,x_j)}{\partial v_m} &= -\frac{w_m(x_i)}{2}\left\{x_i\cdot\left[\left(1-\tanh^2\left(v_m^T x_i\right)\right)\right.\right. \\ &\left.\cdot\tanh\left(v_m^T x_j\right)\right]^T \\ &\left.+x_j\cdot\left[\left(1-\tanh^2\left(v_m^T x_j\right)\right)\cdot\tanh\left(v_m^T x_i\right)\right]^T\right\}\end{aligned} \tag{23}$$

As a result, the parameter v_m can be updated by Eq. (24) during the rth training procedure.

$$v_m = v_m - \eta\frac{\partial O}{\partial v_m} \tag{24}$$

Table 1 The *mAP*(%) values of the comparative experimental results in the Cifar10 dataset

NN	Bits	MROLH	MROLB	OCH	KMH_QRank	KMH	ITQ_QRank	ITQ_WhRank	ITQ	LSH_QRank	LSH_WhRank	LSH
10	32	12.26	12.14	11.61	11.25	8.56	7.25	5.1	4.08	4.15	4.03	2.68
	64	17.31	17.26	16.95	16.37	11.59	9.58	8.01	7.7	8.58	7.3	5.83
	128	21.47	21.35	21.02	20.73	15.12	15.60	13.95	13.27	14	9.99	9.36
100	32	12.08	11.97	11.38	11.02	8.31	7.04	4.92	3.81	4.02	3.76	2.47
	64	17.05	16.93	16.65	16.08	11.35	9.32	7.82	7.48	8.34	7.18	5.57
	128	21.07	20.98	20.71	20.45	14.86	15.38	13.61	13.04	13.75	9.67	9.03

Similarly, for the parameter w_m, the partial derivation of the objective function is shown in Eq. (25).

$$\frac{\partial\Theta}{\partial w_m}=\sum_{\tilde{c}\in\hat{C}}\phi(\tilde{c})(1-\phi(\tilde{c}))\cdot\pi_m^r(x_i)\cdot\frac{\partial D_{WH}(x_i,x_k,x_i,x_j)}{\partial w_m} \tag{25}$$

$$\frac{\partial D_{WH}(x_i,x_k,x_i,x_j)}{\partial w_m}=-\frac{1}{2}\cdot\tanh(v_m^T x_i)\cdot\left(\tanh\left(v_m^T x_k\right)-\tanh\left(v_m^T x_j\right)\right) \tag{26}$$

During the iterative training procedure, we can compute the value of w_m by Eq. (27).

$$w_m=w_m-\lambda\frac{\partial\Theta}{\partial w_m} \tag{27}$$

The iterative process for learning the hash functions and bitwise weight functions which can preserve the ordinal relation is described as in Algorithm 1.

Algorithm 1 Minimal Residual Ordinal Loss Hash

```
Input: Training data: $X$; Binary bits: $M$.
Output: Hash functions: $V = \{v_1,\cdots,v_M\}$, Bitwise weights functions: $W = \{w_1,\cdots,w_M\}$;
 1: Randomly initialize $V$ and $W$.
 2: Generate $n$ training samples $\{x_1,...,x_n\}$ using k-means method.
 3: Initialize the weight of each sample as $\frac{1}{n}$.
 4: for $m = 1 : M$ do
 5:     repeat
 6:         Compute $v_m$ according to Eq. (24).
 7:         Compute $w_m$ according to Eq. (27).
 8:         update the data weights according to Eq. (10).
 9:     until convergence
10:     for $i = 1 : n$ do
11:         Assign the weight of $x_i$ to $\pi_{m+1}^0(x_i)$.
12:     end for
13: end for
```

4 Results and discussion

In this section, we describe the ANN search comparative experiments.

4.1 Experimental setting

In this paper, we evaluate the comparative experiments on three large datasets SIFT1M [18], GIST1M [19], and Cifar10 [20], which are widely used in ANN search experiments. The SIFT1M dataset contains 1 million SIFT descriptors [24] with 128 dimensions, and 100,000 of them are considered as training samples. We also randomly select 10,000 features from SIFT1M dataset as query samples. In GIST1M dataset, there are 1 million 320-dimensional GIST descriptors [25], and we separately choose 50,000 and 10,000 data as training and query samples. The Cifar10 dataset contains 60,000 GIST features with 320 dimensions, and 50,000 samples are utilized as training dataset. Correspondingly, the number of query samples in Cifar10 dataset is 10,000.

The baseline methods include two kinds of algorithms: the binary code methods and bitwise weight methods. Locality-sensitive hashing (LSH) [7], iterative quantization hashing (ITQ) [10], and k-means hashing (KMH) [11] can generate the absolute similarity preserving binary codes. In contrast, ordinal constraint hashing (OCH) [6]

Table 2 The *mAP*(%) values of the comparative experimental results in the GIST1M dataset

NN	Bits	MROLH	MROLB	OCH	KMH_QRank	KMH	ITQ_QRank	ITQ_WhRank	ITQ	LSH_QRank	LSH_WhRank	LSH
10	32	4.35	4.26	4.03	3.87	2.35	2.42	1.97	1.67	1.33	1.09	0.89
	64	8.23	8.17	7.64	7.18	4.42	6.03	4.95	3.26	3.32	2.13	1.99
	128	11.14	11.06	10.53	10.22	6.79	8.75	6.39	4.27	6.83	4.86	2.75
100	32	3.98	3.93	3.75	3.61	2.17	2.25	1.68	1.43	1.08	0.85	0.67
	64	7.84	7.75	7.42	6.85	4.27	5.87	4.71	3.02	3.07	1.88	1.67
	128	10.53	10.48	10.27	10.03	6.54	8.57	6.13	4.01	6.58	4.62	2.56

Table 3 The *mAP*(%) values of the comparative experimental results in the SIFT1M dataset

NN	Bits	MROLH	MROLB	OCH	KMH_QRank	KMH	ITQ_QRank	ITQ_WhRank	ITQ	LSH_QRank	LSH_WhRank	LSH
10	32	6.13	6.02	5.62	5.05	4.38	4.99	3.17	3.07	3.74	2.81	2.24
	64	17.26	17.13	16.57	16.21	8.89	15.16	8.86	5.71	10.22	6.14	5.44
	128	31.34	31.06	30.76	20.06	10.17	29.86	18.15	7.94	25.15	15.92	7.47
100	32	5.69	5.53	5.37	4.82	3.98	4.73	2.93	2.84	3.48	2.67	2.03
	64	16.72	16.67	16.42	16.07	8.76	14.95	8.62	5.53	10.07	5.94	5.17
	128	30.89	30.75	30.48	18.35	8.92	29.78	17.89	7.75	24.89	15.71	7.18

aims to preserve the relative similarity in the Hamming space. QRank [17] and WhRank [16] assign different weights to each binary bit, which can be applied to further boost the ANN search performance of the binary code methods.

We use the criterion of *mAP* and *recall* to evaluate the ANN search performance. As defined in Eq. (28), *recall* represents the fraction of the positive data that are successfully returned. N_{positive} means the number of the positive data that are retrieved. N_{all} is the number of the true nearest neighbors.

$$\text{recall} = \frac{N_{\text{positive}}}{N_{\text{all}}} \tag{28}$$

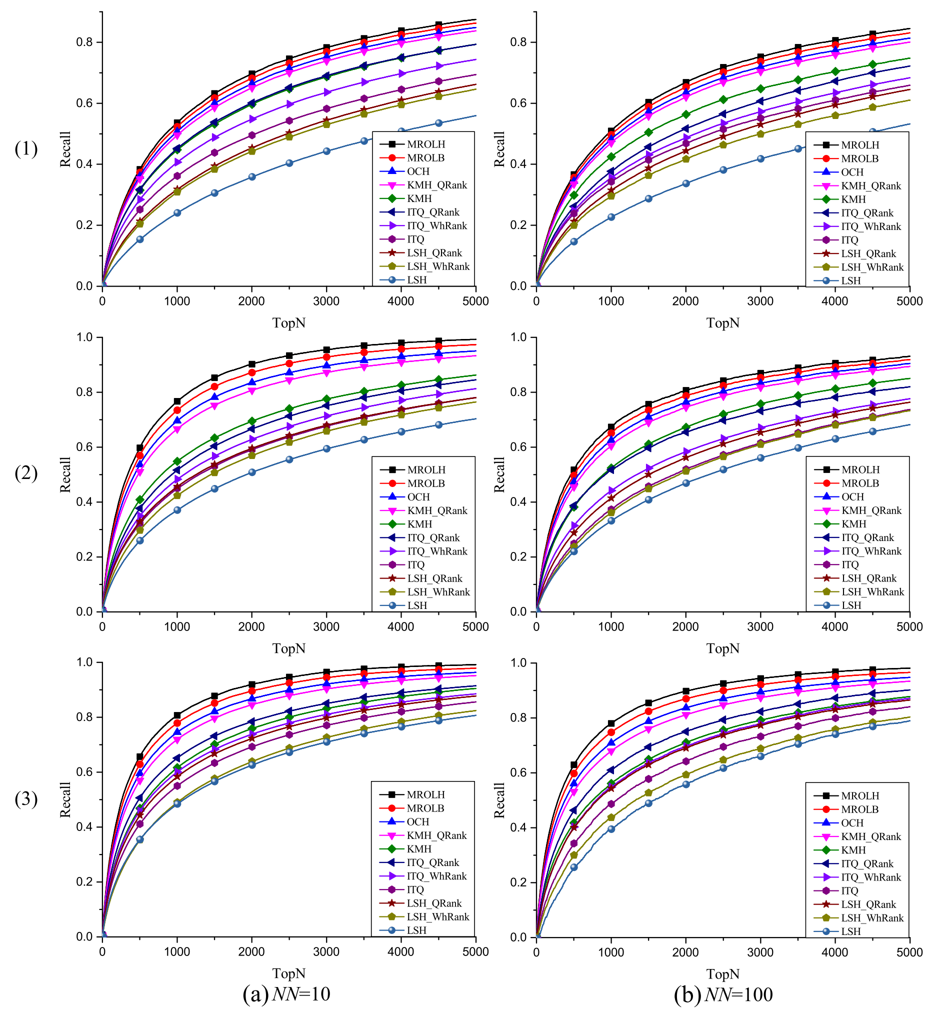

Fig. 3 The *recall* curves of the comparative experiments in the Cifar10 dataset. The data are separately mapped into 32-, 64-, and 128- bit binary codes in the (1), (2), and (3) rows. The number of the true nearest neighbors in columns **a** and **b** separately are 10 and 100

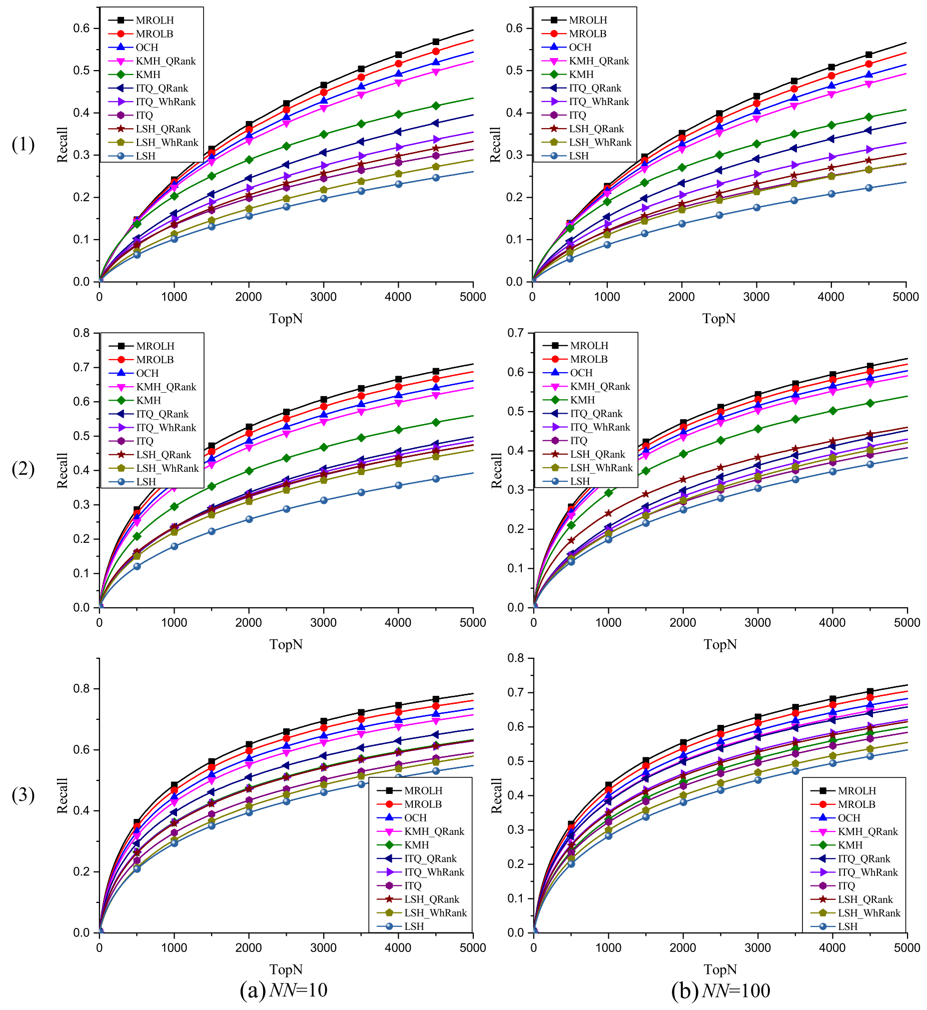

Fig. 4 The *recall* curves of the comparative experiments in the GIST1M dataset. The data are separately mapped into 32-, 64-, and 128- bit binary codes in the (1), (2), and (3) rows. The number of the true nearest neighbors in columns **a** and **b** separately are 10 and 100

The *recall* criterion cannot exactly express which position the ith positive data point locates in. To fix this problem, the criterion of *mAP* defined in Eq. (29) is adopted. Where $|Q|$ represents the number of query samples, K_i is the number of the ith query sample's ground truth. $rank(j)$ is the ranking position of the jth true positive sample in the retrieval results.

$$mAP = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{K_i}\sum_{j=1}^{K_i}\frac{j}{\text{rank}(j)} \tag{29}$$

4.2 Experimental results

In this section, the data are separately mapped into 32-, 64-, and 128-bit binary codes, and their corresponding bitwise weights are learnt.

The purpose of hashing algorithms is to guarantee the approximate nearest neighbors' retrieval results obtained in the Hamming space are identical to those in the Euclidean space. Therefore, we consider a data pair's Euclidean distance as its true similarity degree, and we separately define the 10 and 100 samples with smaller Euclidean distances to a query data as its ground truth in this paper. We show the experimental results in Tables 1, 2, and 3, and Figs. 3, 4, and 5. In the experimental results, MROLB represents the retrieval results obtained according to the binary codes, and MROLH utilizes the bitwise weights to further improve the ANN search performance of MROLB. From the experimental results, we know that MROLH and MROLB separately obtains the best ANN search performance in the Hamming space and the weighted Hamming space.

LSH [7] randomly generates hashing functions without training process, and its performance cannot be obviously

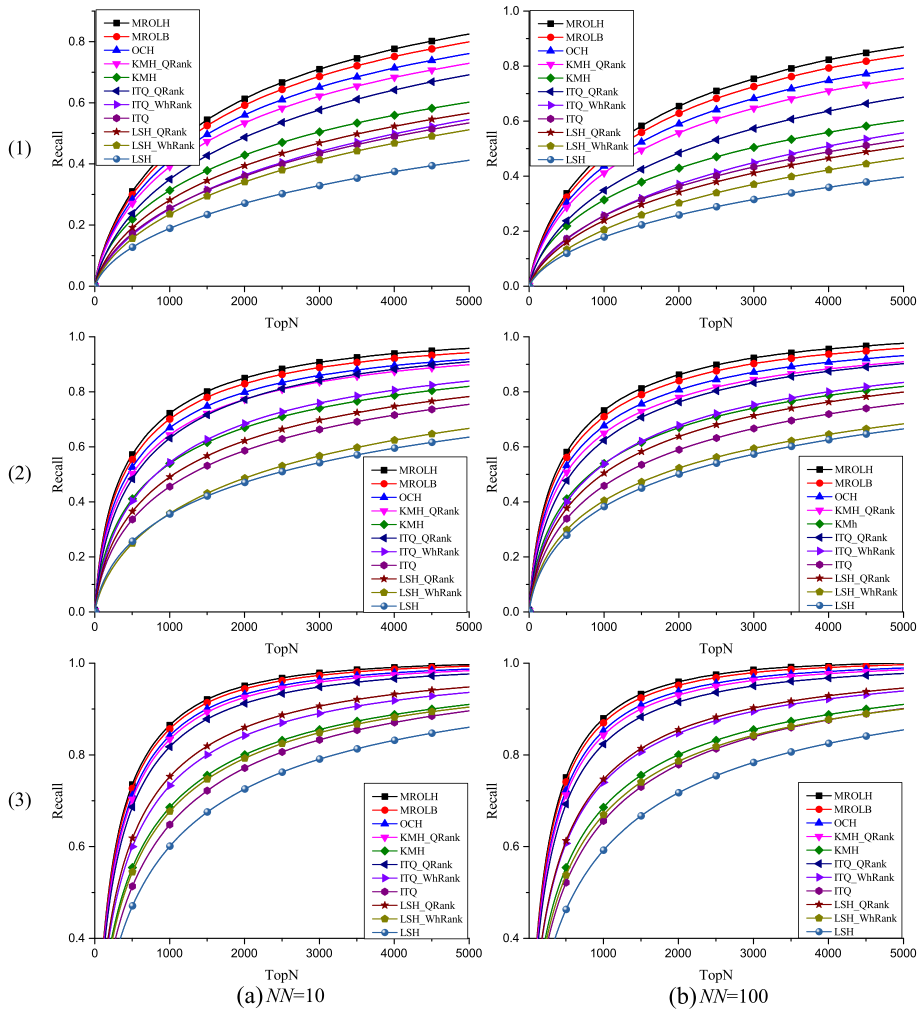

Fig. 5 The *recall* curves of the comparative experiments in the SIFT1M dataset. The data are separately mapped into 32-, 64-, and 128- bit binary codes in the (1), (2), and (3) rows. The number of the true nearest neighbors in columns **a** and **b** separately are 10 and 100

improved with the binary bits increasing. ITQ [10], KMH [11], and MROLB utilize a machine learning mechanism to generate compact binary codes which can achieve satisfying ANN search performance. ITQ [10] maps data points to the vertices of a hyper cubic. However, the vertices in ITQ [10] are fixed, and the encoding results are not adaptive to the data distribution. To fix this problem, KMH [11] learns encoding centers by simultaneously minimizing the quantization loss and the similarity loss. LSH [7], ITQ [10], and KMH [11] belong to the absolute similarity preserving hashing. In contrast, OCH [6] establishes an ordinal constraint to preserve the relative similarity among data points in the Hamming space. For the above hashing methods, the learning procedure of each binary bit is independent with each other, and the residual performance loss accumulated by former bits cannot be eliminated. To solve this problem, we propose to iteratively boost the weights of incorrectly encoded data during training process. Furthermore, we establish a ordinal relation preserving constraint based on quartic samples, which can obviously enhance the power of preserving relative similarity.

WhRank [16] can distinguish the similarity degree among the data pairs which have the same Hamming distance. Furthermore, the bitwise weights in QRank [17] and the proposed MROLH are sensitive to query data. As a result, for the data pairs with the same binary code,

Table 4 Comparison of time consumed (seconds) in the GIST1M dataset. We utilize 50,000 GIST to train hashing functions, and map 1 million GIST to 128-bit binary code during the online encoding procedure

	Our method	OCH	KMH	ITQ	LSH	WhRank	QRank
Online encoding	8.5	8.0	46.0	8.3	8.1	8.4	58.6
Offline training	1861	1572	1382	75	2	569	1183

Table 5 The objective value ($\times 10^3$) decreases as the number of iteration increases, and it converges when the number of iteration reaches 700

Iteration number	200	300	400	500	600	700	800	900	1000	1100
Θ	851.7	524.9	382.4	249.5	203.8	193.7	193.2	193.5	193.3	193.4

their similarity degree can be distinguished by QRank and MROLH. WhRank [16] and QRank [17] demand the bitwise weights should satisfy the absolute similarity preserving restriction, and utilize fixed binary codes to learn bitwise weights. Different from WhRank and QRank, we simultaneously learn the binary codes and bitwise weights by minimizing the ordinal relation preserving loss. As a result, MROLH can well preserve the relative similarity in both the Hamming space and the weighted Hamming space, and the ANN search performances can be iteratively boosted by the feedback mechanism.

4.3 The efficiency and convergence

An excellent hashing method should online encode a raw data efficiently and has a reasonable offline training time complexity [10]. Below, we separately discuss the time complexity of all compared methods.

For online encoding a query data as M-bit binary code, our algorithm, LSH [7], ITQ [10], OCH [6], and WhRank [16] need to compute the sign of the results projected by M linear functions, and they have the same time complexity of $O(M)$. Correspondingly, KMH [11] should compute and compare the distances between a query data and 2^M centers, and its time complexity is $O(2^M)$. QRank [17] firstly transforms a query data into anchor representation and computes its similarities to 2^M landmarks in $O(r + 2^M)$ and obtains query-adaptive weights by quadratic programming in polynomial time. Here, r represents the number of anchors.

For offline training stage, LSH [7] randomly generates M linear hashing functions with a constant time. QRank [17] represents 2^M landmarks using r anchors in $O(2^M r)$. The time complexity of WhRank [16] is $O(Mdk)$, and k represents the number of nearest neighbors. ITQ [10] iteratively optimizes a rotation matrix with a linear time complexity. In order to decrease training time complexity, OCH [6] and KMH [11] just select n ($n \ll N$) samples with d dimensions from all N data to join in their training procedure. For each iteration, KMH [11] computes and compares the distances between n data points and 2^M centers, and the time complexity is $O(2^M nd)$. In contrast, the overall training complexity of OCH [6] is $O(tMn^3d + nN)$, and t is the number of iteration. For our algorithm, the training process includes three stages, and we separately discuss their time complexity as below: Firstly, we adopt k-means algorithm to select n centers from N training samples, which needs to compare the distance relationship between each training data and all cluster centers. Therefore, the time complexity of the first stage is $O(Nnd)$. Secondly, we utilize a gradient descent algorithm to minimize the performance loss, and the time complexity mainly depends on the number of training groups. Initially, a training group contains quartic items, and its number is n^4. Actually, we project the original set to an approximation ordinal relation set established based on triplet elements, and the number of training groups reduces to n^3. In addition, to map d dimensional data to M-bit binary code, the hash functions with Md parameters are learnt. As a result, the time complexity of the second stage is $O(Mn^3d)$. Thirdly, to minimize the residual error, we update the weights of n training samples before learning each hashing function, and the time complexity of this stage is $O(Mn)$. As described above, the overall training time complexity of our method is $O(Nnd + Mn^3d + Mn)$.

To validate the above analysis, we separately test the efficiency of online encoding procedure and offline training process in the GIST1M dataset, and the time consumed is shown in Table 4.

Generally, we consider an algorithm to have converged when its objective value remains unchanged or changed a little. In this paper, we define the number of triplet elements whose ordinal relation is not well preserved as the objective value Θ. We conduct the convergence experiments in the GIST1M database, and the number of training samples is 50,000. As shown in Table 5, the objective value decreases as the iteration number increases. But, it changes a little after 700 iterations, and we consider the algorithm to have converged.

5 Conclusion

In this paper, we propose a novel hashing algorithm dubbed minimal residual ordinal loss hashing (MROLH). Different from tradition hashing algorithms, MROLH simultaneously learns binary codes and bitwise weights by a feedback mechanism. When the algorithm converges, the encoding results and bitwise weights are well adaptive to each other. In this paper, we aim to preserve the data pairs' original relative similarity in both the Hamming space and the weighted Hamming space. Furthermore, we establish the relative similarity preserving constraint based on quartic samples to obviously enhance the power of preserving ordinal relation. During the training process, we iteratively boost the weight of the data whose relative similarity is not preserved. Thus, the residual performance loss can be minimized during later training procedure. Extensive experiments on three benchmark datasets demonstrate that the proposed MROLH is superior to many existing stat-of-the-art approaches. In the future work, we will investigate to decrease the probability of an ambiguous ranking occurring at the top position of retrieval results.

Abbreviations
ITQ: Iterative quantization hash; KMH: k-means hash; LSH: Locality-sensitive hash; MROLH: Minimal residual ordinal loss hash; QRank: Query-sensitive ranking method; WhRank: Ranking based on weighted hamming distance

Acknowledgements
The authors would like to thank the editor and anonymous reviewers for their helpful comments and valuable suggestions.

Authors' contributions
All authors take part in the discussion of the work described in this paper. ZW, LZ, and PL conceived and designed the experiments. ZW performed the experiments. ZW, LZ, and FS analyzed the data. ZW, LZ, and FS wrote the paper. All authors read and approved the final version of the manuscript.

Funding
This work is funded by the Natural Science Foundation of Shandong Province of China (Grant No. ZR2018PF005), and the National Natural Science Foundation of China (Grant No. 61841602) .

Availability of data and materials
SIFT1M: [18]
GIST1M: [19]
Cifar10: [20]
Cifar10: http://www.cs.toronto.edu/~kriz/cifar.html
Please contact author for data requests.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Computer Science and Technology, Shandong University of Technology, Zibo, 255000 China. [2]School of Computer Science and Technology, Jilin University, Changchun, 130012 China.

Received: 4 June 2019 Accepted: 18 February 2020
Published online: 05 March 2020

References
1. X. Luo, P. Zhang, Z. Huang, L. Nie, X. Xu, Discrete hashing with multiple supervision. IEEE Trans. Image Process. **28**(6), 2962–2975 (2019)
2. M. Hu, Y. Yang, F. Shen, N. Xie, R. Hong, H.T. Shen, Collective reconstructive embeddings for cross-modal hashing. IEEE Trans. Image Process. **28**(6), 2770–2784 (2019)
3. Y. Cui, J. Jiang, Z. Lai, Z. Hu, W. Wong, Supervised discrete discriminant hashing for image retrieval. Pattern Recog. **78**, 79–90 (2018)
4. C. Yue, B. Liu, M. Long, J. Wang, in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Hashgan: deep learning to hash with pair conditional wasserstein gan (IEEE, Salt Lake City, 2018), pp. 1287–1296
5. C. Li, C. Deng, N. Li, W. Liu, X. Gao, D. Tao, in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Self-supervised adversarial hashing networks for cross-modal retrieval (IEEE, Salt Lake City, 2018), pp. 4242–4251
6. H. Liu, R. Ji, J. Wang, C. Shen, Ordinal constraint binary coding for approximate nearest neighbor search. IEEE Trans. Pattern. Anal. Mach. Intell. **41**(4), 941–955 (2019)
7. M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, in *Proceedings of Twentieth Annual Symposium on Computational Geometry*. Locality-sensitive hashing scheme based on p-stable distributions (ACM, Brooklyn, 2004), pp. 253–262
8. S. He, B. Wang, Z. Wang, Y. Yang, F. Shen, Z. Huang, H.T. Shen, Bidirectional discrete matrix factorization hashing for image search. IEEE Trans. Cybern., 1—12 (2019). https://ieeexplore.ieee.org/document/8863122
9. M. Hu, Y. Yang, F. Shen, N. Xie, H.T. Shen, Hashing with angular reconstructive embeddings. IEEE Trans. Image Process. **27**(2), 545–555 (2018)
10. Y. Gong, S. Lazebnik, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Iterative quantization: a procrustean approach to learning binary codes (IEEE, Colorado Springs, 2011), pp. 817–824
11. K. He, F. Wen, J. Sun, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. K-means hashing: an affinity-preserving quantization method for learning binary compact codes (IEEE, Portland, 2013), pp. 2938–2945
12. J. Wang, J. WANG, N. YU, S. Li, in *Proceedings of the 21st ACM International Conference on Multimedia*. Order preserving hashing for approximate nearest neighbor search (ACM, Barcelona, 2013), pp. 133–142
13. Y.G. Jiang, J. Wang, S.F. Chang, in *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*. Lost in binarization: query-adaptive ranking for similar image search with compact codes (ACM, Trento, 2011), pp. 16–1168
14. Y. G. Jiang, J. Wang, X. Xue, S. F. Chang, Query-adaptive image search with hash codes. IEEE Trans. Multimedia. **15**(2), 442–453 (2013)
15. H.Y. Shum, L. Zhang, X. Zhang, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Qsrank: query-sensitive hash code ranking for efficient neighbor search (IEEE, Providence, 2012), pp. 2058–2065
16. L. Zhang, Y. Zhang, J. Tang, K. Lu, Q. Tian, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Binary code ranking with weighted hamming distance (IEEE, Portland, 2013), pp. 1586–1593
17. T. Ji, X. Liu, C. Deng, L. Huang, B. Lang, in *Proceedings of the 22nd ACM International Conference on Multimedia*. Query-adaptive hash code ranking for fast nearest neighbor search (ACM, Orlando, 2014), pp. 1005–1008
18. H. Jegou, M. Douze, C. Schmid, Product quantization for nearest neighbor search. IEEE Trans. Pattern. Anal. Mach. Intell. **33**(1), 117–128 (2011)
19. J. Wang, S. Kumar, S.F. Chang, Semi-supervised hashing for large-scale search. IEEE Trans. Pattern. Anal. Mach. Intell. **33**(12), 2393–2406 (2012)
20. A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images. Computer Science Department, University of Toronto, Tech Rep (2009)
21. M. Norouzi, D.J. Fleet, in *Proceedings of the 28th International Conference on Machine Learning*. Minimal loss hashing for compact binary codes (ACM, Bellevue, 2011), pp. 353–360
22. M. Norouzi, D.M. Blei, R. Salakhutdinov, in *Proceedings of the Advances in Neural Information Processing Systems, Harrahs and Harveys, Lake Tahoe, USA*. Hamming distance metric learning (Curran Associates Inc, Lake Tahoe, 2012), pp. 1070–1078
23. J. Wang, W. Liu, A.X. Sun, Y.G. Jiang, in *Proceedings of the IEEE International Conference on Computer Vision*. Learning hash codes with listwise supervision (IEEE, Sydney, 2013), pp. 3032–3039
24. D. G. Lowe, Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
25. A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope. Int. J. Comput. Vis. **42**(3), 145–175 (2001)