algorithms

MDPI

Article

Top Position Sensitive Ordinal Relation Preserving Bitwise Weight for Image Retrieval

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

[1] School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China; sunfuzhen@sdut.edu.cn (F.S.); zhanglb@sdut.edu.cn (L.Z.); wanglei0511@sdut.edu.cn (L.W.)

[2] School of Computer Science and Technology, Jilin University, Changchun 130000, China; liupp@jlu.edu.cn

* Correspondence: zhwang@sdut.edu.cn

Received: 16 December 2019; Accepted: 3 January 2020; Published: 6 January 2020

Abstract: In recent years, binary coding methods have become increasingly popular for tasks of searching approximate nearest neighbors (ANNs). High-dimensional data can be quantized into binary codes to give an efficient similarity approximation via a Hamming distance. However, most of existing schemes consider the importance of each binary bit as the same and treat training samples at different positions equally, which causes many data pairs to share the same Hamming distance and a larger retrieval loss at the top position. To handle these problems, we propose a novel method dubbed by the top-position-sensitive ordinal-relation-preserving bitwise weight (TORBW) method. The core idea is to penalize data points without preserving an ordinal relation at the top position of a ranking list more than those at the bottom and assign different weight values to their binary bits according to the distribution of query data. Specifically, we design an iterative optimization mechanism to simultaneously learn binary codes and bitwise weights, which makes their learning processes related to each other. When the iterative procedure converges, the binary codes and bitwise weights are effectively adapted to each other. To reduce the training complexity, we relax the discrete constraints of both the binary codes and the indicator function. Furthermore, we pretrain a tensor ordinal graph to decrease the time consumption of computing a relative similarity relationship among data points. Experimental results on three large-scale ANN search benchmark datasets, i.e., SIFT1M, GIST1M, and Cifar10, show that the proposed TORBW method can achieve superior performance over state-of-the-art approaches.

Keywords: image retrieval; binary code; hash algorithm; bitwise weights; top-rank-sensitive; ordinal-relation-preserving

1. Introduction

With the rapid development of massive image collections, it has been challenging to search for visually relevant images effectively and efficiently [1,2]. In contrast to commonly used methods that exhaustively search for the most similar images in one high-dimensional space, hashing methods map floating point data into binary codes and achieve tasks of searching approximate nearest neighbors (ANNs) using Hamming distances. Therefore, hashing methods can accelerate ANN search procedures and save on storage. Recently, hashing methods have been applied in the area of computer vision and machine learning [3–6].

The pioneering method, locality-sensitive hashing (LSH) [7], randomly generates linear hashing functions and computes binary codes based on projection signs. The learning process is independent of training samples, and the performance cannot obviously improve as the number of binary bits increases [8]. To fix the above problem, many data-dependent hashing algorithms have been proposed to preserve training samples' similarity relationships in Hamming spaces. Generally speaking,

according to the similarity-preserving restriction, we roughly divide data-dependent hashing algorithms into pairwise similarity-preserving hashing [8–12] and ordinal relation-preserving ones [1,13].

The former kind of hashing algorithms, such as k-means hashing (KMH) [9], iterative quantitation (ITQ) [10], anchor graph hashing (AGH) [8], minimal loss hashing (MLH) [11], and semantic hashing [12], aim to preserve the original similarity relationship of a data pair in a Hamming space. AGH [8] first learns training centers by the k-means algorithm. Then, it utilizes a spectral graph to represent the neighbor information and computes binary codes by a graph cut mechanism. ITQ [10] maps similar data points into the same vertex of a cubic and assigns them the same binary code. KMH [9] learns binary codes by minimizing quantization loss and similarity loss to make encoding results well fit the distribution of training data. In semantic hashing [12], Hamming distances among data points with the same label should be small enough; otherwise, their Hamming distances should be larger than the threshold. MLH [11] aims to penalize similar (or dissimilar) points when they have large (small) Hamming distance values to assign by a hinge-like loss function.

The above methods have shown promising performance for searching semantic neighbors, but ANN search tasks mainly focuses on an ordinal relation among data points. To fulfill this requirement, ordinal-relation-preserving hashing, such as triplet loss hashing (TLH) [14], ordinal constraint hashing (OCH) [1], and order-preserving hashing (OPH) [13], were proposed to preserve data points' Euclidean ranking orders in a Hamming space. OPH [13] divides all data points into different clusters and demands the ranking orders of cluster centers should be consistent in different spaces. TLH [14] defines a relative-similarity-preserving restriction based on triplet elements and demands the Hamming distance of similar data pairs should be smaller than that of dissimilar data pairs. Wang [15] also adopted a similarity-preserving restriction in [14] to learn binary codes. Recently, deep learning mechanisms have been adopted to dramatically improve the performance of binary coding techniques [16–19]. However, high performance has been coupled with high computational and storage overhead. As declared in ITQ [10], binary coding methods should be efficient and scalable for encoding new high-dimensional data.

The hash algorithms mentioned above treat all binary bits equally, and the Hamming distances are discrete integer values. Therefore, many data points with different binary codes would share the same Hamming distance to a query data, and their ranking orders are tied in Hamming spaces. To solve this problem, bitwise weight methods assign different weight values to binary bits and utilize weighted Hamming distances to resort retrieval results. QaRank [20,21] demands that query-adaptive weights should minimize intra-class similarities and maintain inter-class relations among the top similar images. QsRank [22] directly considers a floating point query sample and a target nearest neighbor radius as inputs, and it is designed only for principal component analysis hashing. In WhRank [23], bitwise weights are adaptive to training samples and sensitive to query data. QRank [24] learns query-adaptive weights by exploiting the discriminative power of each hash function and their complements.

Although the aforementioned hash methods and bitwise weights methods have shown their efficacy for ANN search tasks, we would argue that most of them treat data points at different ranking positions equally. Furthermore, few of them explore learning hash functions and bitwise weights functions simultaneously. To this end, we propose a novel top-position-sensitive ordinal-relation-preserving bitwise weight (TORBW) method, as shown in Figure 1. We iteratively learn binary codes and bitwise weights and demand them to satisfy the top-position-sensitive relative-similarity-preserving restriction. For the ANN search task, we utilize weighted Hamming distances to distinguish data pairs that share the same similarity degree. The main contributions of this work are concluded as follows:

(1) When learning binary codes and bitwise weights, we propose to penalize data points without preserving an ordinal relation at the top position of a ranking list more than those at the bottom. This measure helps to reduce the probability of chaos ranking occurring at the top position.

(2) Unlike a general two-step mechanism, we simultaneously learn binary codes and bitwise weights by an iterative mechanism, and they can feedback into each other. When the mechanism converges, the binary codes and bitwise weights are effectively adapted to each other.

(3) A tensor ordinal graph (TOG) is precomputed to represent a relative similarity relationship of any two data pairs, which can effectively reduce the training complexity.

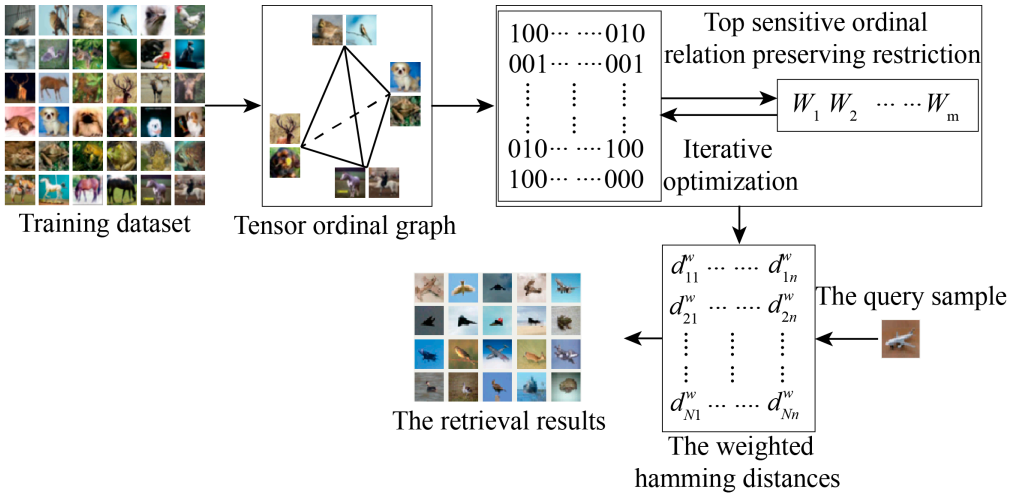

Figure 1. Flow chart of the proposed top-position-sensitive ordinal-relation-preserving bitwise weight (TORBW) method. A tensor ordinal graph is constructed to approximate a similarity relationship of any two data pairs, and we establish a top-position-sensitive ordinal-relation-preserving restriction to improve the performance. We adopt an iterative optimization mechanism to simultaneously learn binary codes and bitwise weights. During the retrieval of approximate nearest neighbors, we utilize weighted hamming distances to resort a chaos ranking list in a Hamming space.

The rest of this paper is organized as follows: Section 2 describes the proposed TORBW method and the adopted iterative optimization mechanism. In Section 3, we show and discuss experimental results. Finally, we conclude this paper in Section 4.

2. Method

2.1. Top-Position-Sensitive Ordinal-Relation-Preserving Restriction

We define $X \in R^{d \times N}$ as a training dataset, which includes N samples, and $x_i \in R^{d \times 1}$ is the i-th data point. Our aim is to simultaneously learn a set of hash functions and bitwise weight functions. Thus, we can map x_i into r-bit binary code $b_i = \{h_1(x_i), \cdots, h_r(x_i)\}$ and generate its bitwise weights $[w_1(x_i), \cdots, w_r(x_i)]$. Here, $h_c(x_i) = sgn({u_c}^T x_i + b)$ represents the c-th hash function, and $sgn(\cdot)$ is the sign function. If $u_c \leftarrow [u_c\ b]$ and $x_i \leftarrow [x_i\ 1]$, we can rewrite the c-th hash function as $h_c(x_i) = \mathrm{sgn}\ ({u_c}^T x_i)$. Similarly, we define the c-th bit weight function as $w_c(x_i) = {v_c}^T x_i$, and it is sensitive to query sample x_i.

In this paper, we achieve an ANN search task in two steps [20–23]. First, we retrieve nearest neighbors by Hamming distances as in most commonly used hash algorithms [1,9,11–13]. Then, for data points that share the same Hamming distance to a query sample, the weighted Hamming distance $d_{wh}(\cdot,\cdot)$ is utilized to resort their ranking orders, as shown in Equation (1):

$$d_{wh}(x_i, x_j) = \sum_{c=1}^{r} w_c(x_j) \cdot \left(b_c(x_i) \otimes b_c(x_j)\right). \tag{1}$$

Therefore, to guarantee the performance of an ANN search task, the data points' ordinal relation should be well preserved in both a Hamming space and a weighted Hamming space. Generally, existing ordinal-relation-preserving restrictions treat each data point at different ranking positions

equally [1,13]. In contrast, evaluation standards of ANN search tasks, such as mean average precision (*mAP*), pay more attention to samples at the top position [25]. To achieve the above requirement, we define a top-position-sensitive ordinal-relation-preserving restriction as in Equation (2), which can effectively optimize the precision at the top position:

$$L(x_i, x_j) = \log(1 + R_E(x_i, x_j) + R_H(x_i, x_j)), \tag{2}$$

where R_E and R_H represent the position of x_j at the ranking lists of x_i's nearest neighbors in different spaces. For the loss function defined in Equation (2), its first-order derivation is larger than zero, and the second-order derivation is smaller than zero. Obviously, $L(x_i, x_j)$ is a monotonic nondecreasing function, and its growth ration drastically decreases as the ranking order increases. Therefore, the loss function $L(x_i, x_j)$ can penalize more of the samples without preserving its original ranking orders at the top position than those at the bottom.

As defined in Equation (3), $R_E(x_i, x_j)$ represents the ranking order of x_j to x_i in a Euclidean space:

$$R_E(x_i, x_j) = \sum_{k=1}^{n} I(d(x_i, x_k) < d(x_i, x_j)), \tag{3}$$

where $d(\cdot,\cdot)$ represents the Euclidean distance. As shown in Equation (4), $R_H(x_i, x_j)$ calculates the number of dissimilar samples with closer positions than x_j in a Hamming space:

$$R_H(x_i, x_j) = \sum_{l=1}^{n} I(d_{wh}(x_i, x_l) < d_{wh}(x_i, x_j)), \tag{4}$$

where $d_{wh}(\cdot,\cdot)$ returns the weighted Hamming distance and R_H is computed on the basis of R_E. The first question is how to compute the ranking list R_E. The naive method of forming triplet data (x_i, x_j, x_k) is to select a similar data pair (x_i, x_k) and a dissimilar data pair (x_i, x_j) with a Euclidean distance. However, this kind of representation needs to randomly select triple tuples and compare the similarity degree among all data points side by side, which is time-consuming and needs costly memory. To avoid the above situation, we learn a TOG to represent a relative similarity relationship of any two data pairs in advance [1].

Given a dataset X, we first construct an affinity graph S, as shown in Equation (5):

$$S(i, j) = \begin{cases} 0, & i = j \\ e^{-\|x_i - x_j\|_2^2 / 2\sigma^2}, & otherwise \end{cases}, \tag{5}$$

where the value in S represents the similarity degree of a data pair and is computed according to a Euclidean distance. Next, we further define a dissimilar graph DS, and $DS(i, j) = 1/S(i, j)$,

We can formulate a TOG as shown in Equation (6):

$$G = S \otimes DS, \tag{6}$$

where $\bigotimes$ is defined as $G(ij, kl) = S(i, j) \cdot DS(k, l)$. Therefore, each entry of G relates to two data pairs, and the relative similarity relationship of any two data pairs (i, j, i, k) can be represented through the TOG, as shown in Equation (7):

$$\begin{cases} \delta_{ij} < \delta_{ik}, & G(ij, ik) > 1 \\ \delta_{ij} > \delta_{ik}, & G(ij, ik) \leq 1 \end{cases}, \tag{7}$$

where δ_{ij} represents the similarity degree of data pair (x_i, x_j). Thus, we can form $R_E(x_i, x_j)$ by directly counting elements, of which values are smaller than 1 in the column ij. $R_H(x_i, x_j)$ can be generated by selecting triplet elements with $d_{wh}(x_i, x_j) > d_{wh}(x_i, x_k)$ in the set $R_E(x_i, x_j)$.

2.2. Relaxation and Iterative Optimization

In this section, we describe the process of learning hash and bitwise weight functions by minimizing loss function $L(\cdot,\cdot)$. Different from the two-step mechanism, we design an iterative optimization mechanism that simultaneously learns parameters of hash and bitwise weight functions [9,10].

As binary codes and indicator functions are discrete integer values, directly optimizing loss function $L(\cdot,\cdot)$ becomes difficult. To fix this problem, we adopt a relaxation mechanism.

First, we utilize $tanh(\cdot)$ instead of the encoding function as shown in Equation (8), and the weighted Hamming distance is rewritten as in Equation (9):

$$h_c(x) = \mathrm{sgn}({u_c}^T x) \approx \tanh({u_c}^T x), \tag{8}$$

$$d_{wh}(x_i, x_j) = \frac{1}{2}\left(\sum_{c=1}^{r} w_c(x_j) - \sum_{c=1}^{r} w_c(x_j) \cdot h_c(x_i) \cdot h_c(x_j)\right). \tag{9}$$

Secondly, we employ the sigmoid function to approximate the indicator function as in shown in Equation (10):

$$I(d_{wh}(x_i, x_k) \leq d_{wh}(x_i, x_j)) \approx G(z) = \frac{1}{1 + \exp(-z)}. \tag{10}$$

The definition of z is shown in Equation (11):

$$z = d_{wh}(x_i, x_k) - d_{wh}(x_i, x_j). \tag{11}$$

After the above relaxation procedure, we utilize a stochastic gradient descent algorithm to optimize the objective function in Equation (12):

$$O = \sum_{i,j} L(x_i, x_j) = \sum_{i,j} \log(1 + R_E(x_i, x_j) + R_H(x_i, x_j)). \tag{12}$$

For the parameter of the c-th hash function u_c, the partial derivation of the objective function O is shown in Equation (13):

$$\begin{aligned} \frac{\partial O}{\partial u_c} &= \frac{R_H(x_i,x_j)}{1+R_E(x_i,x_j)+R_H(x_i,x_j)} \cdot \frac{\partial R_H(x_i,x_j)}{\partial u_c} \\ &= \frac{R_H(x_i,x_j)}{1+R_E(x_i,x_j)+R_H(x_i,x_j)} \cdot \frac{\partial \sum_k G(d_{wh}(x_i,x_k) - d_{wh}(x_i,x_j))}{\partial u_c} \\ &= \frac{R_H(x_i,x_j)}{1+R_E(x_i,x_j)+R_H(x_i,x_j)} \cdot \sum_k \left(G(z)(1-G(z))\right) \cdot \left[\frac{\partial d_{wh}(x_i,x_k)}{\partial u_c} - \frac{\partial d_{wh}(x_i,x_j)}{\partial u_c}\right] \end{aligned} \tag{13}$$

The partial derivation of the weighted Hamming distance is shown in Equation (14):

$$\frac{\partial d_{wh}(x_i, x_k)}{\partial u_c} = \frac{w_c(x_k)}{2} \cdot [(1 - h_c^2(x_i)) \circ h_c(x_k) + (1 - h_c^2(x_k)) \circ h_c(x_i)], \tag{14}$$

where $\circ$ represents the element-wise product. As a result, we can update the value of the parameter u_c shown in Equation (15):

$$u_c = u_c - \lambda_1 \frac{\partial O}{\partial u_c}. \tag{15}$$

For v_c of the c-th bitwise weight function, the partial derivation of the objective function is shown in Equation (16):

$$\begin{aligned}\frac{\partial O}{\partial v_c} &= \frac{R_H(x_i,x_j)}{1+R_E(x_i,x_j)+R_H(x_i,x_j)} \cdot \frac{\partial R_H(x_i,x_j)}{\partial v_c} \\ &= \frac{R_H(x_i,x_j)}{1+R_E(x_i,x_j)+R_H(x_i,x_j)} \cdot \frac{\partial \sum_k G(d_{wh}(x_i,x_k)-d_{wh}(x_i,x_j))}{\partial v_c} \\ &= \frac{R_H(x_i,x_j)}{1+R_E(x_i,x_j)+R_H(x_i,x_j)} \cdot \sum_k (G(z)(1-G(z))) \cdot [\frac{\partial d_{wh}(x_i,x_k)}{\partial v_c} - \frac{\partial d_{wh}(x_i,x_j)}{\partial v_c}] \\ &= \frac{R_H(x_i,x_j)}{1+R_E(x_i,x_j)+R_H(x_i,x_j)} \cdot \sum_k (G(z)(1-G(z))) \cdot [\frac{x_k}{2}(1-h_c(x_i)h_c(x_k)) - \frac{x_j}{2}(1-h_c(x_i)h_c(x_j))]\end{aligned} \tag{16}$$

As a result, we can update the parameter v_c shown in Equation (17):

$$v_c = v_c - \lambda_2 \frac{\partial O}{\partial v_c}. \tag{17}$$

The details of the proposed TORBW method are shown in Algorithm 1.

Algorithm 1 TORBW

Input: Data set X={x_1,$\cdots$, x_n}, the number of binary bits r, and parameters λ_1 and λ_2.
Output: The coefficients of hash functions (u_1, $\cdots$, u_r) and bitwise weight functions (v_1, $\cdots$, v_r).

```
Choose training samples from X by the k-means algorithm;
Compute an affinity graph S and a dissimilar graph DS by Equation (5);
Construct a tensor ordinal graph G by Equation (6);
Generate an ordinal relations set of triplet elements (x_i, x_j, x_k) by G
for c=1:r
  repeat
    Compute the partial derivation by Equation (13);
    Update the value of u_c by Equation (15);
    Compute the partial derivation by Equation (16);
    Update the value of v_c by Equation (17);
  until convergence or reaching the maximum iteration number
end for.
```

3. Experimental Results and Discussion

In this section, we first introduce three publicly available datasets for ANN search experiments. Then, we describe the compared methods and the ANN search performance evaluation metrics. Finally, we compare the proposed TORBW method against several state-of-the-art hashing algorithms and bitwise weight methods.

3.1. Database and Setup

We conduct ANN search experiments on three datasets, i.e., SIFT1M [26], GIST1M [27], and Cifar10 [28] datasets. SIFT1M [26] consists of one million SIFT [29] descriptors with 128 dimensions. In SIFT1M, we randomly select 100,000 samples as a training dataset, and 10,000 data points are used as query samples. GIST1M [27] includes one million GIST [30] descriptors that have 320 dimensions, and we respectively select 100,000 data points as a training data and query samples. In Cifar10 [28], there are six thousand tiny images, which are described as 320-dimensional GIST descriptors [30]. We utilize all image descriptors in Cifar10 as training samples, and one thousand samples are considered as query data points.

3.2. Compared Methods and Evaluation Metrics

To prove that the proposed TORBW method can achieve excellent ANN search performance, we compare the TORBW method against five hashing algorithms and two bitwise weight methods. Among them, LSH [7], AGH [8], KMH [9], Top-RSBC [25], and OCH [1] aim to generate excellent

binary codes. WhRank [23] and QRank [24] learn bitwise weights to resort the tied ranking orders, and we use them to further boost the ANN search performances of LSH [7], AGH [8], and KMH [9].

In this experiment, we use the criterions *recall* [31,32] and *mAP* to evaluate the ANN search performances. *recall* represents the fraction of positive data that are successfully returned as defined in Equation (18).

$$recall = \frac{N_{positive}}{N_{all}}, \tag{18}$$

where $N_{positive}$ means the number of positive data that are retrieved and N_{all} is the number of the true nearest neighbors. We further use *mAP* to exactly express which position the *i*-th positive data point locates in, and the definition is shown in Equation (19):

$$mAP = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{K_i}\sum_{j=1}^{K_i}\frac{j}{rank(j)}, \tag{19}$$

where $|Q|$ represents the number of query samples, Ki is the number of the i-th query sample's ground truth and $rank(j)$ is the ranking position of the j-th true positive sample in the retrieval results.

3.3. Experimental Results

In this section, we first encode floating data into 32-, 64-, and 128-bit binary codes by the hash methods (TORBW, LSH [7], AGH [8], KMH [9], Top-RSBC [25], and OCH [1]) and achieve ANN search tasks according to Hamming distances. Then, TORBW, WhRank [23], and QRank [24] assign different weight values to each binary bit and utilize the weighted Hamming distances to resort the tied ranking orders. The ANN search experimental results are shown in Tables 1–3 and Figures 2–4. We separately define the ground truth as 10 and 100 nearest neighbors in the Euclidean space.

Table 1. The *mAP* (%) values of the comparative experiments on the GIST1M dataset.

Ground Truth		**10**			**100**	
Binary Bits	**32**	**64**	**128**	**32**	**64**	**128**
TORBW	4.25	7.86	10.72	3.92	7.65	10.53
OCH	4.03	7.64	10.53	3.75	7.42	10.27
Top-RSBC	3.95	7.42	10.48	3.71	7.26	10.17
KMH_QRank	3.87	7.18	10.22	3.61	6.85	10.03
KMH	2.35	4.42	6.79	2.17	4.27	6.54
AGH_QRank	2.38	5.96	8.63	2.16	5.82	8.47
AGH_WhRank	1.93	4.89	6.31	1.62	4.63	6.08
AGH	1.62	3.21	4.19	1.35	3.01	3.96
LSH_QRank	1.33	3.32	6.83	1.08	3.07	6.58
LSH_WhRank	1.09	2.13	4.86	0.85	1.88	4.62
LSH	0.89	1.99	2.75	0.67	1.67	2.56

Table 2. The *mAP* (%) values of the comparative experiments on the Cifar10 dataset.

Ground Truth		**10**			**100**	
Binary Bits	**32**	**64**	**128**	**32**	**64**	**128**
TORBW	11.84	17.12	21.26	11.52	16.81	20.96
OCH	11.61	16.95	21.02	11.38	16.65	20.71
Top-RSBC	11.46	16.58	20.94	11.24	16.37	20.63
KMH_QRank	11.25	16.37	20.73	11.02	16.08	20.45
KMH	8.56	11.59	15.12	8.31	11.35	14.86
AGH_QRank	7.22	9.46	15.53	6.94	9.25	15.27
AGH_WhRank	5.06	7.94	13.92	4.85	7.76	13.53
AGH	4.02	7.68	13.21	3.72	7.45	12.98
LSH_QRank	4.15	8.58	14.00	4.02	8.34	13.75
LSH_WhRank	4.03	7.30	9.99	3.76	7.18	9.67
LSH	2.68	5.83	9.36	2.47	5.57	9.03

Table 3. The *mAP* (%) values of the comparative experiments on the SIFT1M dataset.

Ground Truth		10			100	
Binary Bits	32	64	128	32	64	128
TORBW	5.83	16.82	21.05	5.54	16.71	31.59
OCH	5.62	16.57	20.76	5.37	16.42	30.48
Top-RSBC	5.43	16.38	20.43	5.06	16.21	25.37
KMH_QRank	5.05	16.21	20.06	4.82	16.07	18.35
KMH	4.38	8.89	10.17	3.98	8.76	8.92
AGH_QRank	4.87	15.03	29.74	4.67	14.86	29.64
AGH_WhRank	3.12	8.73	18.02	2.92	8.53	17.75
AGH	2.93	5.64	7.85	2.81	5.46	7.62
LSH_QRank	3.74	10.22	25.15	3.48	10.07	24.89
LSH_WhRank	2.81	6.14	15.92	2.67	5.94	15.71
LSH	2.24	5.44	7.47	2.03	5.17	7.18

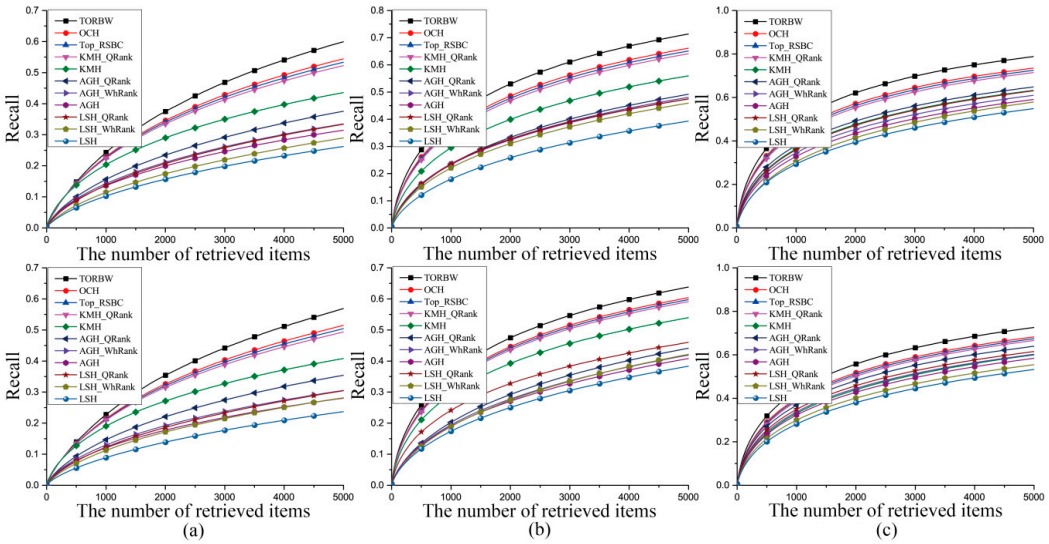

Figure 2. The *recall* curves of the comparative experiments on the GIST1M dataset mapped into 32-bit (**a**), 64-bit (**b**), and 128-bit (**c**) binary codes. The numbers of the true nearest neighbors in the first row and the second row are 10 and 100, respectively.

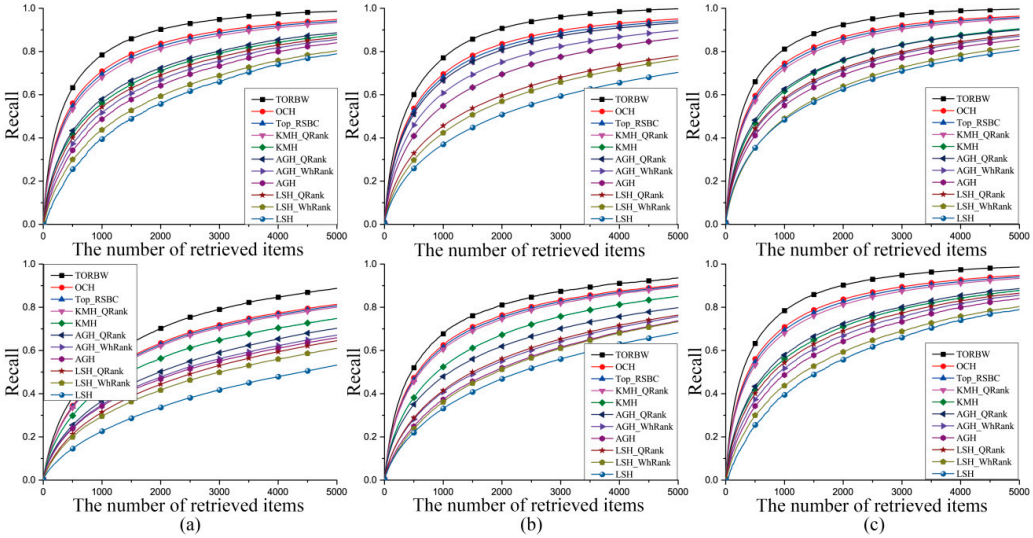

Figure 3. The *recall* curves of the comparative experiments on the Cifar10 dataset mapped into 32-bit (**a**), 64-bit (**b**), and 128-bit (**c**) binary codes. The numbers of the true nearest neighbors in the first row and the second row are 10 and 100, respectively.

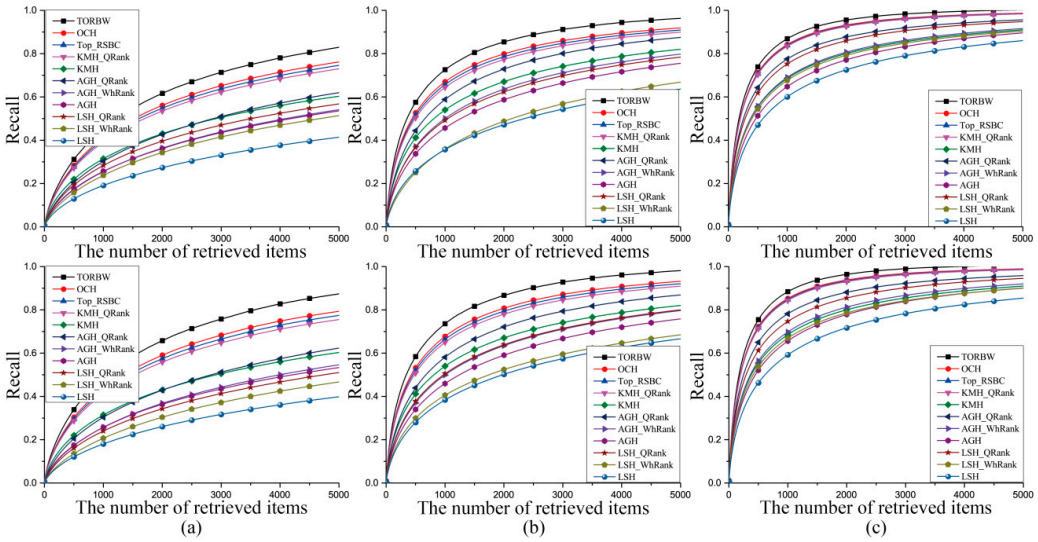

Figure 4. The *recall* curves of the comparative experiments on the SIFT1M dataset mapped into 32-bit (**a**), 64-bit (**b**), and 128-bit (**c**) binary codes. The numbers of the true nearest neighbors in the first row and the second row are 10 and 100, respectively.

We perform ANOVA tests by matlab function "anovan1()" and show the experimental results in Table 4. From the experimental results, we know that the improvements made by our algorithm are significant.

Table 4. ANOVA tests on three widely benchmark datasets.

	SIFT1M	**GIST1M**	**Cifar10**
p value	6.57×10^{-8}	1.76×10^{-5}	1.9×10^{-3}

In LSH [7], the learning process is independent from training samples, and the ANN search performance cannot evidently improve as the number of binary bits increases [8]. Thus, to obtain a satisfying ANN search performance, LSH needs a long binary code or multiple hash tables. In contrast, data-dependent algorithms, such as TORBW, OCH [1], Top-RSBC [25], KMH [9], and AGH [8], utilize a kind of machine learning mechanism to learn hashing functions, which can minimize the ANN search performance loss on a training database. The data-dependent hashing algorithms can achieve a better ANN search performance with compact binary codes. In this paper, the maximum number of binary bits is only 128, and the data-dependent hashing algorithms are superior to LSH. AGH [8] generates centers with the k-means algorithm and learns binary codes with the spectral graph cut mechanism. However, AGH demands that the distribution of training samples should be uniform [10]. In practice, the databases used in this paper do not obey a uniform distribution. KMH [9] learns encoding centers, which can minimize both quantization loss and similarity loss. As a result, the performance of KMH is better than that of AGH. AGH and KMH aim to preserve a pointwise similarity relationship. Top-RSBC defines a pairwise similarity relationship restriction for learning hashing functions, which can further enhance the power of preserving a similarity relationship among samples. The ANN search task emphasizes preserving an ordinal relation. To fulfill this task, TORBW and OCH [1] generate binary codes, which can preserve an original ordinal relation among data points in a Hamming space. Thus, TORBW and OCH have a superior performance over AGH, KMH, and LSH. As described above, the binary coding methods, OCH [1], Top-RSBC [25], and KMH [9], have shown efficacy for ANN search tasks, but they consider the importance of each binary bit equally. Therefore, many data with different binary codes would share the same Hamming distance to a query sample, and the tied ranking orders that exist in their retrieval results would result in inferior performance. To avoid the above ambiguous situation, TORBW, WhRank [23], and QRank [24] assign different weight values

to each binary bit, and they use the weighted Hamming distance to resort the retrieval results. As a result, bitwise weight methods can effectively improve the ANN search performances of binary coding methods. The bitwise weights in TORBW and QRank [24] are particularly sensitive to query data points, and they have a better performance. The aforementioned methods including WhRank [23], QRank [24], OCH [1], KMH [9], ITQ [10], and LSH [7] treat all data points equally. However, evaluation criterions of ANN search tasks pay more attention at the top position of the retrieval results. To satisfy this requirement, TORBW and Top-RSBC [25] penalize mistakes at the top position more than those at the bottom. Therefore, TORBW and Top-RSBC can well preserve a similarity relationship among data points at the top position. As described above, TORBW can effectively preserve ordinal relations and utilizes query-sensitive bitwise weights to reduce the tied ranking orders occurring at the top position. Finally, TORBW achieves an excellent ANN search performance.

4. Conclusions

In this paper, we propose a novel method dubbed as the TORBW method, which simultaneously generates hash functions and bitwise weight functions. To guarantee the performance of an ANN search task, we demand the preservation of an original ordinal relation among data points in both a Hamming space and a weighted Hamming space. We utilize a TOG to represent a relative similarity relationship between any two data pairs, which can reduce the complexity of constructing a training dataset. Different from a two-step mechanism, which separately learns binary codes and bitwise weights, TORBW adopts an iterative mechanism to simultaneously optimize them. When the algorithm converges, binary codes and bitwise weights are effectively adapted to each other. Generally, training samples are considered to be equal. In contrast, we assign a large weight value to training samples at the top position during the learning process. As a result, it can effectively reduce the probability of tied ranking orders occurring at the top position of a ranking list. Extensive experiments on three benchmark datasets demonstrate that TORBW has a better ANN search performance than many existing state-of-the-art binary coding methods and bitwise weight methods. In this paper, we adopt linear hashing functions to map data into binary codes. However, the practical dataset may be linearly inseparable. In future work, we will employ a kernel formulation for target hashing functions to further enhance the performance of preserving ordinal relations.

Author Contributions: Conceptualization, Z.W. and L.Z.; methodology, F.S.; software, L.W.; validation, Z.W., F.S., and P.L.; formal analysis, Z.W.; investigation, P.L.; resources, F.S.; data curation, L.Z.; writing of the original draft preparation, Z.W.; writing of review and editing, P.L.; visualization, L.W.; supervision, F.S.; project administration, L.Z.; funding acquisition, Z.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Shandong Province of China (grant number: ZR2018PF005) and the National Natural Science Foundation of China (grant number: 61841602).

Acknowledgments: The authors express their gratitude to the institutions that supported this research: Shandong University of Technology (SDUT) and Jilin University (JLU).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Liu, H.; Ji, R.; Wu, Y.; Huang, F. Ordinal constrained binary code learning for nearest neighbor search. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2019; Volume 41, pp. 941–955.
2. Luo, X.; Zhang, P.F.; Huang, Z.; Nie, L.Q.; Xu, X.S. Discrete hashing with multiple supervision. *IEEE Trans. Image Process.* **2019**, *28*, 2962–2975. [CrossRef] [PubMed]
3. Wu, G.S.; Han, J.G.; Guo, Y.C.; Liu, L.; Ding, G.; Ni, Q.; Shao, L. Unsupervised deep video hashing via balanced code for large-scale video retrieval. *IEEE Trans. Image Process.* **2019**, *28*, 1993–2007. [CrossRef] [PubMed]
4. Chen, Z.D.; Li, C.X.; Luo, X.; Nie, L.; Zhang, W.; Xu, X.S. Scratch: A scalable discrete matrix factorization hashing framework for cross-modal retrieval. *IEEE Trans. Circ. Syst. Video* **2019**. [CrossRef]

5. Ding, K.; Yang, Z.; Wang, Y.; Liu, Y. An improved perceptual hash algorithm based on u-net for the authentication of high-resolution remote sensing image. *Appl. Sci.* **2019**, *9*, 2972. [CrossRef]
6. Yang, H.; Yin, J.; Jiang, M. Perceptual image hashing using latent low-rank representation and uniform lbp. *Appl. Sci.* **2018**, *8*, 317. [CrossRef]
7. Datar, M.; Immorlica, N.; Indyk, P.; Mirrokni, V.S. Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the Annual Symposium on Computational Geometry, Brooklyn, NY, USA, 8–11 June 2004.
8. Liu, W.; Wang, J.; Kumar, S.; Chang, S.F. Hashing with graphs. In Proceedings of the International Conference on Machine Learning, Bellevue, WA, USA, 28 June–2 July 2011.
9. He, K.; Wen, F.; Sun, J. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013.
10. Gong, Y.; Lazebnik, S. Iterative quantization: A procrustean approach to learning binary codes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011.
11. Norouzi, M.; Fleet, D.J. Minimal loss hashing for compact binary codes. In Proceedings of the International Conference on Machine Learning, Bellevue, WA, USA, 28 June–2 July 2011.
12. Salakhutdinov, R.; Hinton, G. Semantic hashing. *Int. J. Approx. Reason.* **2009**, *50*, 969–978. [CrossRef]
13. Wang, J.; Wang, J.; Yu, N.; Li, S. Order preserving hashing for approximate nearest neighbor search. In Proceedings of the ACM International Conference on Multimedia, Barcelona, Spain, 21–25 October 2013.
14. Norouzi, M.; Blei, D.M.; Salakhutdinov, R. Hamming distance metric learning. In Proceedings of the Advances in Neural Information Processing Systems, Harrahs and Harveys, Lake Tahoe, Stateline, NV, USA, 3–6 December 2012.
15. Wang, J.; Liu, W.; Sun, A.X.; Jiang, Y.G. Learning hash codes with listwise supervision. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013.
16. Dizaji, G.K.; Zheng, F.; Nourabadi, S.N.; Yang, Y.; Deng, C.; Huang, H. Unsupervised deep generative adversarial hashing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.
17. Shen, F.; Xu, Y.; Liu, L. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE Trans. Pattern Anal.* **2018**, *40*, 3034–3044. [CrossRef] [PubMed]
18. Zhang, H.; Liu, L.; Long, Y.; Shao, L. Unsupervised deep hashing with pseudo labels for scalable image retrieval. *IEEE Trans. Image Process.* **2018**, *27*, 1626–1638. [CrossRef] [PubMed]
19. Chen, Z.; Yuan, X.; Lu, J.; Tian, Q.; Zhou, J. Deep hashing via discrepancy minimization. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.
20. Jiang, Y.G.; Wang, J.; Chang, S.F. Lost in Binarization: Query-adaptive ranking for similar image search with compact codes. In Proceedings of the ACM International Conference on Multimedia Retrieval, Trento, Italy, 18–20 April 2011.
21. Jiang, Y.G.; Wang, J.; Xue, X.; Chang, S.F. Query-adaptive image search with hash codes. *IEEE Trans. Multimed.* **2013**, *15*, 442–453. [CrossRef]
22. Shum, H.Y.; Zhang, L.; Zhang, X. QsRank: Query-sensitive hash code ranking for efficient ϵ-neighbor search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012.
23. Zhang, L.; Zhang, Y.; Tang, J.; Lu, K.; Tian, Q. Binary code ranking with weighted hamming distance. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013.
24. Ji, T.; Liu, X.; Deng, C.; Huang, L.; Lang, B. Query-adaptive hash code ranking for fast nearest neighbor search. In Proceedings of the ACM International Conference on Multimedia, New York, NY, USA, 10–16 October 2014.
25. Song, D.; Liu, W.; Ji, R.; Meyer, D.A.; Smith, J.R. Top rank supervised binary coding for visual search. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
26. Jegou, H.; Douze, M.; Schmid, C. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal.* **2011**, *33*, 117–128. [CrossRef] [PubMed]

27. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Computer Science Department, University of Toronto: Toronto, ON, Canada, 2009.
28. Wang, J.; Kumar, S.; Chang, S.F. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal.* **2012**, *34*, 2393–2406. [CrossRef] [PubMed]
29. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
30. Oliv, A.; Torralba, A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **2001**, *42*, 145–175. [CrossRef]
31. Le, N.Q.; Ho, Q.T.; Qu, Y.Y. Incorporating deep learning with convolutional neural networks and position specific scoring matrices for identifying electron transport proteins. *J. Comput. Chem.* **2017**, *38*, 2000–2006. [CrossRef] [PubMed]
32. Le, N.Q.K.; Huynh, T.T.; Yapp, E.K.Y.; Yeh, H.Y. Identification of clathrin proteins by incorporating hyperparameter optimization in deep learning and PSSM profiles. *Comput. Methods Programs Biomed.* **2019**, *177*, 81–88. [CrossRef] [PubMed]