Hindawi
Scientific Programming
Volume 2020, Article ID 2424381, 11 pages
https://doi.org/10.1155/2020/2424381

Research Article

AVBH: Asymmetric Learning to Hash with Variable Bit Encoding

Yanduo Ren, Jiangbo Qian, Yihong Dong, Yu Xin, and Huahui Chen

Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo, Zhejiang 315211, China

Correspondence should be addressed to Jiangbo Qian; qianjiangbo@nbu.edu.cn

Received 16 October 2019; Accepted 10 December 2019; Published 21 January 2020

Guest Editor: Aibo Song

Nearest neighbour search (NNS) is the core of large data retrieval. Learning to hash is an effective way to solve the problems by representing high-dimensional data into a compact binary code. However, existing learning to hash methods needs long bit encoding to ensure the accuracy of query, and long bit encoding brings large cost of storage, which severely restricts the long bit encoding in the application of big data. An asymmetric learning to hash with variable bit encoding algorithm (AVBH) is proposed to solve the problem. The AVBH hash algorithm uses two types of hash mapping functions to encode the dataset and the query set into different length bits. For datasets, the hash code frequencies of datasets after random Fourier feature encoding are statistically analysed. The hash code with high frequency is compressed into a longer coding representation, and the hash code with low frequency is compressed into a shorter coding representation. The query point is quantized to a long bit hash code and compared with the same length cascade concatenated data point. Experiments on public datasets show that the proposed algorithm effectively reduces the cost of storage and improves the accuracy of query.

1. Introduction

Given a query object/point **q** and a dataset **S**, the nearest neighbour search (NNS) [1–3] is to return the nearest neighbours in **S** to **q**. Nowadays, the NNS is widely used in many applications such as image retrieval, text classification, and recommendation systems. However, with the exponential growth of data scale and the disaster of the high data dimensionality, the NNS problem is now much more difficult to solve than before. Therefore, new efficient index structures and query algorithms for similarity searches have increasingly become the focus of research for the problem.

The hashing-based NNS methods [3–5] have attracted much attention. Generally, the hashing methods can project the original data with locality preserved to a low-dimensional Hamming space, i.e., binary codes [4–6]. The complexity of those methods is always in sublinear time. In addition, the hashing methods only need a simple bit operation to compute the similarity from Hamming encoding, which is very fast. As the high performance in large-scale data retrieval, hashing techniques have gained increasing interests in facilitating cross-view retrieval tasks [7, 8], online retrieval tasks [9], and metric learning tasks [10].

For large-scale data retrievals, the time and space costs are the two important issues. As we know, the accuracy of existing hash methods is limited by the length of hash encoding and usually requires a longer coding to get better accuracy. However, a long coding will increase the space cost, network communication overhead, and response time.

In order to solve this problem, a coding quantization mechanism [11] based on asymmetric hashing algorithm [12] was proposed. Different from the direct hash code comparison, by cascade concatenating the coding of the data point to the same encoding length of the query point, the coding storage cost of the dataset is reduced effectively and the accuracy of the result is ensured. However, this algorithm uses a unified compression method for all data, ignoring the effect of data distribution. Actually, the distribution of large-scale data is generally uneven. Hence, for most hashing algorithms, the frequency of quantization is also different. As we know, longer encoding can preserve most of the original information; however, it will bring high

cost and vice versa. A careful trade-off among accuracy, computing overhead, and space-saving needs to be studied. Intuitively, high-density data require longer length encoding to ensure that the original information is preserved as much as possible, while low-density data can use shorter length encoding and still preserve most of the original information. That is the idea behind our algorithm.

In this paper, an asymmetric learning to hash with variable bit encoding algorithm (AVBH) is proposed. The AVBH uses two types of hash mapping functions to quantify the dataset and the query set separately to encode the hash codes with different length bits. In particular, the frequency of dataset is calculated by random Fourier encoding, and then the random Fourier coding with high frequency is compressed into a longer hash code representation, and the random Fourier coding with low frequency is compressed into a shorter hash code representation.

The main contributions of this paper are as follows: (1) a variable bit encoding mechanism (named AVBH) based on hash code frequency compression is proposed, which makes the encoding space effectively used, and (2) the experiment shows that the AVBH can effectively reduce the storage cost and improve the query accuracy.

2. Preliminaries and Description

In this section, we review some basic knowledge of LSH (locality-sensitive hashing) [13–15], vector quantization [16], and product quantization [17] that is essential to our proposed technique.

2.1. Vector Quantization. Vector quantization (VQ) is a classical data compression technique, which compresses the original data into discrete vectors. For a vector $\mathbf{x}$ of n dimensions, formally, a VQ function f can specified as $f(\mathbf{x}) \in \mathbf{C} = \{\mathbf{c}_i, i = 1, 2, \ldots, k\}$, where $\mathbf{x}$ (with n dimensions) is an original data/vector, $\mathbf{C}$ is a pretrained code set, and $\mathbf{c}_i$ is a codeword in the codebook $\mathbf{C}$. The objective of a VQ function is to quantify the original real number vector to the nearest codeword with the lowest VQ loss. Here, the VQ loss of vector $\mathbf{x}$ is given by

$$E_{vq}(\mathbf{x}) = \min_{\mathbf{c}\in\mathbf{C}} \|\mathbf{x} - \mathbf{c}\|^2. \tag{1}$$

2.2. Product Quantization. Product quantization (PQ) is an optimization of vector quantization. Firstly, the feature space is divided into m mutually exclusive subspaces, and each subspace is then quantized separately using VQ. That is, the coding of each subspace forms a small codebook $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m$, and small codebooks form a large codebook $\mathbf{C}$ by the Cartesian product. In this method, a high-dimensional data can be decomposed into m low-dimensional spaces and can be processed in parallel. Suppose an object $\mathbf{x}$ is represented as a combination of m codewords $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_m$, the loss of the product quantization of vector $\mathbf{x}$ is given by

$$E_{pq} = \min_{\mathbf{c}\in\mathbf{C}} \|\mathbf{x} - \mathbf{c}\|^2 = \min_{\mathbf{c}\in\mathbf{C}} \left\| \mathbf{x} - \begin{bmatrix} \mathbf{c}_1^T \\ \mathbf{c}_2^T \\ \vdots \\ \mathbf{c}_m^T \end{bmatrix}^T \right\|^2. \tag{2}$$

2.3. Random Fourier Feature. Traditional dimensionality reduction methods, such as PCA, map the data to the independent feature space and compute the main independent features. This method ignores the nonlinear information of the sample distribution and cannot apply to the actual data well. Based on the feature mapping method of random Fourier feature (RFF), data are mapped to the characteristic space under the approximate kernel function, and the inner product of any two points under the feature space is approximated by their kernel function values. Compared with the PCA method, RFF can maximize the data distribution information and obtain the dimensional characteristic by reducing the dimension or raising the dimension. This kind of characteristic is suitable for the characteristic compression processing. SKLSH [18] is a kind of classical hashing algorithm based on RFF, which has a good experimental result under the long bit digit coding.

The length coding hash learning algorithm firstly maps the sample points from the original n dimension real space to the n dimension of the approximate kernel function feature space by RFF. Because of the convergence of RFF consistency, the kernel function similarity between any two sample points can be maintained.

Specifically, for two points $\mathbf{x}, \mathbf{y}$, the translation invariant kernel function [12] $K(\mathbf{x}, \mathbf{y}) = E(\Phi_{\mathbf{w},b}(\mathbf{x}), \Phi_{\mathbf{w},b}(\mathbf{y}))$ satisfies the following equation:

$$\begin{aligned} &K(\mathbf{x}, \mathbf{y}) = K(\mathbf{x} + \eta, \mathbf{y} + \eta) = K(\mathbf{x} - \mathbf{y}), \\ &K(\mathbf{x}, \mathbf{y}) \le K(\mathbf{x} - \mathbf{x}) = K(0) = 1, \end{aligned} \tag{3}$$

where $\Phi_{\mathbf{w},b} = \sqrt{2}\cos(\mathbf{w}^T\mathbf{x} + b)$, b satisfies the uniform distribution between $[0, 2\pi]$, $\mathbf{w}$ obeys the probability distribution $\mathbf{P}_K$ induced by the translation invariant kernel function, and η is a constant parameter.

Thus, the mapping from the n dimensional space to the feature space of the d dimensional approximation kernel function can be obtained by the following equation:

$$\Phi^d(\cdot) = \frac{1}{\sqrt{d}}\left(\Phi_{\mathbf{w}_1 b_1}(\cdot), \Phi_{\mathbf{w}_2 b_2}(\cdot), \ldots, \Phi_{\mathbf{w}_d b_d}(\cdot)\right), \tag{4}$$

where $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d$ is for the same-sense sampling subject to the probability distribution $\mathbf{P}_K$ and $b_1, b_2, \ldots, b_d$ obeys the same-distribution sampling which is uniformly distributed between the obedience $[0, 2\pi]$. When the translation invariant kernel function is a Gaussian kernel function, $K(\mathbf{x} - \mathbf{y}) = e^{-(\gamma/2)\|\mathbf{x}-\mathbf{y}\|^2}$, $\mathbf{P}_K$ is a Gaussian distribution, i.e., $\mathbf{P}_K \sim \text{Normal}(0, \gamma\mathbf{I}_{n\times n})$.

2.4. Orthogonal Procrustes Problem. An orthogonal Procrustes problem is to solve an orthogonal transforming matrix $\mathbf{O}$, so that $\mathbf{PO}$ is as close to $\mathbf{Q}$ as possible, i.e.,

$$\Gamma(\mathbf{O}) = \min_{\mathbf{O}} \|\mathbf{PO} - \mathbf{Q}\|_F, \tag{5}$$

where $\mathbf{O}^T\mathbf{O} = \mathbf{I}$. This formula is not easy to be solved directly, and it can be optimized by alternating optimization. Namely, the matrix $\mathbf{P}$ is first to be fixed, and the matrix $\mathbf{Q}$ is optimized to make the target function value reduced. Then the matrix $\mathbf{Q}$ is fixed, and the orthogonal transforming matrix $\mathbf{O}$ is optimized to make the target function value reduced.

3. Asymmetric Learning to Hash with Variable Bit Encoding

3.1. Algorithm Framework. For a general hash learning algorithm, the length of the hash code by learning is always fixed. AVBH uses the idea of asymmetric hashing algorithm, that is, the hash code for the dataset is short and unfixed, and the query point of the code is long and fixed. The steps of the AVBH hashing algorithm are shown in Figure 1, which mainly includes the dataset encoding steps ①–③ and the query point encoding step ④.

The dataset encoding section consists of two phases: random Fourier feature encoding (RFF encoding) and variable bit encoding (AVBH encoding). First, step ① uses the random Fourier feature (RFF) to map the dataset and get RFF encoding. After RFF coding, considering the difference of RFF coding frequency, the RFF coding frequency is sorted in step ②. According to the requirement, the original dataset can be divided into the subset by the RFF code as the length of $k_1, k_2, \ldots, k_L$ shown in the figure. As shown in step ③, the AVBH subset encoding of the length of $k_1, k_2, \ldots, k_L$ can be reproduced by duplicating $(n/k_1), (n/k_2), \ldots, (n/k_L)$ times sequentially and then the Hamming code of n dimension is formed.

In the query point encoding section, the query point quantization is encoded into RFF encoding of length n by step ④.

3.2. Objective Function. The target of the AVBH method is to get L groups of hash encoding with the length of $k_1, k_2, \ldots, k_L$ through the hash function $G(\mathbf{x})$, namely,

$$\mathbf{B}^{(l)} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N_l} \\ b_{21} & b_{22} & \cdots & b_{2N_l} \\ \vdots & \vdots & \ddots & \vdots \\ b_{k_l1} & b_{k_l2} & \cdots & b_{k_lN_l} \end{bmatrix}, \quad b_{ij} \in \{+1, -1\},\ i \in [1, 2, \ldots, k_l],\ j \in [1, 2, \ldots, N_l],\ l \in [1, 2, \ldots, L], \tag{6}$$

where $N = \sum_{l=1}^{L} N_l$, $n = \sum_{l=1}^{L} k_l$.

This divides the dataset $\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(L)}$ into subset according to the RFF encoding frequency. By cascading $(n/k_1), (n/k_2), \ldots, (n/k_L)$ times, respectively, we can get L group n bit hash code. For example,

$$\widetilde{\mathbf{B}}^{(l)} = \left[\left(\mathbf{B}^{(l)}\right)^T, \left(\mathbf{B}^{(l)}\right)^T, \ldots, \left(\mathbf{B}^{(l)}\right)^T\right]^T, \quad l \in [1, 2, \ldots, L]. \tag{7}$$

Then combine $\widetilde{\mathbf{B}}^{(1)}, \widetilde{\mathbf{B}}^{(2)}, \ldots, \widetilde{\mathbf{B}}^{(L)}$ to get a n bit long hash code of the entire dataset $\mathbf{B} \in \{+1, -1\}^{n \times N}$. The AVBH method calculates the similarity by calculating the Hamming distance between the hash code of the query point and the concatenated dataset during the query process. Therefore, for the dataset, we need to construct the hash mapping function, so that the L group hash code obtained with the length of $k_1, k_2, \ldots, k_L$, respectively, can preserve the original information as much as possible. Therefore, the AVBH method obtains the hash mapping function by the reconstruction error (8) between the minimum cascading encoding $\mathbf{B}$ and n dimension sample vector $\mathbf{Y}$:

$$\text{Loss}(\mathbf{B}, \mathbf{R}) = \min \|\mathbf{RB} - \mathbf{Y}\|_F^2, \tag{8}$$

where $\mathbf{R}$ is an orthogonal rotation $n \times n$ matrix, namely, $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$.

Combining the properties of associative matrices and the definition of F-norm of matrices, we can get the following equation:

$$\begin{aligned} \text{Loss}(\mathbf{B}, \mathbf{R}) &= \min \|\mathbf{RB} - \mathbf{Y}\|_F^2 \\ &= \min\left\{\text{tr}\left[(\mathbf{RB} - \mathbf{Y})^T(\mathbf{RB} - \mathbf{Y})\right]\right\} \\ &= \min\left\{\text{tr}\left[(\mathbf{RB})^T\mathbf{RB}\right] + \text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) - 2\text{tr}\left(\mathbf{RBY}^T\right)\right\}. \end{aligned} \tag{9}$$

As the unknown variable $\mathbf{B}, \mathbf{R}$ in formula (8) is the product relation, the expanded formula (9) contains two items of unknown variables, so it is difficult to solve. After further simplification, we can get the following formula:

$$\begin{aligned} \text{Loss}(\mathbf{B}, \mathbf{R}) &= \min \|\mathbf{RB} - \mathbf{Y}\|_F^2 \\ &= \min\left\{\text{tr}\left[(\mathbf{RB})^T\mathbf{RB}\right] + \text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) - 2\text{tr}\left(\mathbf{RBY}^T\right)\right\} \\ &= \min\left[\text{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{RB}\right) + \text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) - 2\text{tr}\left(\mathbf{RBY}^T\right)\right] \\ &= \min\left[\text{tr}\left(\mathbf{B}^T\mathbf{B}\right) + \text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) - 2\text{tr}\left(\mathbf{RBY}^T\right)\right]. \end{aligned} \tag{10}$$

As $\mathbf{B} \in \{+1, -1\}^{n \times N}$, it is easy to get $\text{tr}(\mathbf{B}^T\mathbf{B}) = nN$. As $\mathbf{R}^T\mathbf{R} = \mathbf{I}$, we can get that $\mathbf{Y}$ is unrelated to $\mathbf{B}$ and $\mathbf{R}$. As a result, $\text{tr}(\mathbf{Y}^T\mathbf{Y}) = \mathbf{c}$, where $\mathbf{c}$ is unrelated to $\mathbf{B}$. So formula (10) is simplified as follows:

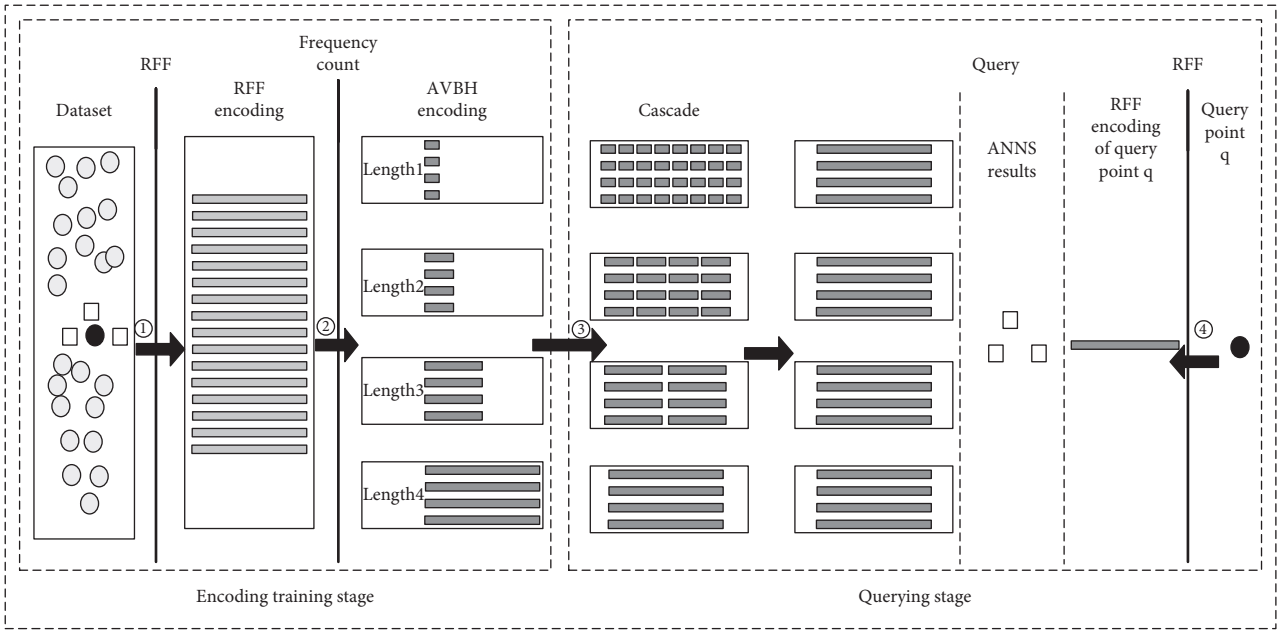

Figure 1: Algorithm framework of AVBH.

$$\begin{aligned}
\text{Loss}(\mathbf{B},\mathbf{R}) &= \min\|\mathbf{RB}-\mathbf{Y}\|_F^2 \\
&= \min\left[\text{tr}\left(\mathbf{B}^T\mathbf{B}\right)+\text{tr}\left(\mathbf{Y}^T\mathbf{Y}\right)-2\text{tr}\left(\mathbf{RBY}^T\right)\right] \\
&= \min\left[\text{tr}\left(\mathbf{B}^T\mathbf{B}\right)+\text{tr}\left(\mathbf{Y}^T\mathbf{RR}^T\mathbf{Y}\right)-2\text{tr}\left(\mathbf{RBY}^T\right)\right] \\
&= \min\left[nN+\mathbf{c}-2\text{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{Y}\right)\right] \\
&= \min\left\{\text{tr}\left(\mathbf{B}^T\mathbf{B}\right)+\text{tr}\left[\left(\mathbf{R}^T\mathbf{Y}\right)^T\mathbf{R}^T\mathbf{Y}\right]-2\text{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{Y}\right)\right\} \\
&= \min\left\{\text{tr}\left[\left(\mathbf{B}-\mathbf{R}^T\mathbf{Y}\right)^T\left(\mathbf{B}-\mathbf{R}^T\mathbf{Y}\right)\right]\right\} \\
&= \min\left\|\mathbf{B}-\mathbf{R}^T\mathbf{Y}\right\|_F^2.
\end{aligned} \tag{11}$$

Thus, minimizing the reconfiguration error (8) equals minimizing the quantization error (11).

The objective function of AVBH to encode the dataset is to minimize the reconstruction error of the concatenated encoding of the n bit by finding the orthogonal rotation matrix $\mathbf{R}$. In extreme cases of the dataset which is uniformly distributed, there is no significant difference in the frequency of the hash encoding of the dataset, and the AVBH method then degenerates into the ACH algorithm [16]. Compared with the ACH hashing algorithm, the AVBH hashing algorithm is more adaptable to the real data because it can adapt to the data of various distributions and the generalization ability is stronger.

3.3. Optimization Algorithm. The objective function (11) can be optimized by alternating optimization. Namely, the rotation matrix $\mathbf{R}$ is first to be fixed, and the encoding matrix $\mathbf{B}$ is optimized to make the target function value reduced. Then the encoding matrix $\mathbf{B}$ is fixed and the rotation matrix $\mathbf{R}$ is optimized to make the target function value reduced. In this way, the value of the target function decreases until it converges. The following is a discussion of how to tune and optimize the value of the target function.

(1) Fix the rotation matrix $\mathbf{R}$, and optimize the encoding matrix $\mathbf{B}$.

Given $\mathbf{V}=\mathbf{R}^T\mathbf{Y}$, $\mathbf{V}^{lm}$ is a matrix consisted of $(m-1)\times k+1$ row to $m\times k$ row, $(l-1)\times k+1$ column to $l\times k$ column of $\mathbf{V}$. From formula (11), we can get the following equation:

$$\begin{aligned}\mathrm{Loss}_1(\mathbf{B}) &= \min\left[nN + \mathbf{c} - 2\mathrm{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{Y}\right)\right] \\ &= \min\left\{nN + \mathbf{c} - 2\sum_{l=1}^{L}\mathrm{tr}\left[\left(\widetilde{\mathbf{B}}^{(l)}\right)^T\mathbf{V}^l\right]\right\} \\ &= \min\left\{nN + \mathbf{c} - 2\sum_{l=1}^{L}\mathrm{tr}\left\{\left[\left(\mathbf{B}^{(l)}\right)^T, \left(\mathbf{B}^{(l)}\right)^T, \ldots, \left(\mathbf{B}^{(l)}\right)^T\right]\begin{bmatrix}\mathbf{V}^{l1}\\ \mathbf{V}^{l2}\\ \vdots\\ \mathbf{V}^{l\left(n/k_l\right)}\end{bmatrix}\right\}\right\} \\ &= \min\left\{nN + \mathbf{c} - 2\sum_{l=1}^{L}\mathrm{tr}\left[\left(\mathbf{B}^{(l)}\right)^T\left(\sum_{m=1}^{\left(n/k_l\right)}\mathbf{V}^{lm}\right)\right]\right\}.\end{aligned} \tag{12}$$

As $n, N, \mathbf{c}$ are unrelated to $\mathbf{B}$, for a fixed $\mathbf{R}$, the problem of minimizing (12) is equal to the problem of maximizing the following formula:

$$\begin{aligned}\mathrm{Loss}_1(\mathbf{B}) &= 2\max\sum_{l=1}^{L}\mathrm{tr}\left[\left(\mathbf{B}^{(l)}\right)^T\left(\sum_{m=1}^{\left(n/k_l\right)}\mathbf{V}^{lm}\right)\right] \\ &= \max\sum_{l=1}^{L}\left[\sum_{i=1}^{\left(n/k_l\right)}\sum_{j=1}^{n_l}\mathbf{B}_{ij}^{(l)}\left(\sum_{m=1}^{\left(n/k_l\right)}\mathbf{V}_{ij}^{lm}\right)\right],\end{aligned} \tag{13}$$

As $\mathbf{B}_{ij}^{(l)} \in \{+1, -1\}$, optimal analytic solution of formula (13) is given by

$$\mathbf{B}_{ij}^{(l)} = \mathrm{sign}\left(\sum_{m=1}^{\left(n/k_l\right)}\mathbf{V}_{ij}^{lm}\right), \quad i \in [1, 2, \ldots, k_l],\ j \in [1, 2, \ldots, N_l],\ l \in [1, L]. \tag{14}$$

(2) Fix the encoding matrix $\mathbf{B}$, and optimize the rotation matrix $\mathbf{R}$.

Under $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$, the problem of minimizing formula (11) is equal to an Orthogonal Procrustes problem [9]. The optimal solution of such problems with $\mathbf{R}$ is as follows:

$$\begin{aligned}\mathrm{Loss}_2(\mathbf{R}) &= \min\left[nN + \mathbf{c} - 2\mathrm{tr}\left(\mathbf{B}^T\mathbf{R}^T\mathbf{Y}\right)\right] \\ &= \min\left[nN + \mathbf{c} - 2\mathrm{tr}\left(\mathbf{Y}\mathbf{B}^T\mathbf{R}^T\right)\right].\end{aligned} \tag{15}$$

Hence, the problem of optmizing $\mathbf{R}$ to get the minimum value of formula (15) is equal to the problem of maximizing the following formula:

$$\mathrm{Loss}_2(\mathbf{R}) = \max\left[\mathrm{tr}\left(\mathbf{Y}\mathbf{B}^T\mathbf{R}^T\right)\right]. \tag{16}$$

By calculating the SVD of $\mathbf{Y}\mathbf{B}^T$, we can get the following formula:

$$\mathbf{Y}\mathbf{B}^T = \mathbf{U}\mathbf{\Omega}\mathbf{C}^T, \tag{17}$$

where $\mathbf{U}$ is a matrix which consists of left singular value vector, $\mathbf{C}$ is a matrix which consists of right singular value vector, $\mathbf{\Omega}$ is a diagonal matrix which consists of corresponding singular value vectors, and the diagonal elements of which is $\mathbf{\Omega}_{ii} \geq 0$, $i \in [1, n]$.

By combining formulas (16) and (17), we can get the following equation:

$$\begin{aligned}\mathrm{Loss}_2(\mathbf{R}) &= \max\left[\mathrm{tr}\left(\mathbf{U}\mathbf{\Omega}\mathbf{C}^T\mathbf{R}^T\right)\right] \\ &= \max\left[\mathrm{tr}\left(\mathbf{U}\mathbf{\Omega}\left(\mathbf{R}\mathbf{C}\right)^T\right)\right] \\ &= \max\left[\mathrm{tr}\left(\left(\mathbf{R}\mathbf{C}\right)^T\mathbf{U}\mathbf{\Omega}\right)\right].\end{aligned} \tag{18}$$

Given $\mathbf{A} = (\mathbf{R}\mathbf{C})^T\mathbf{U}$, $\widetilde{\mathbf{R}} = \mathbf{R}\mathbf{C}$, $\mathbf{A}_{ii}$ is the diagonal element of $\mathbf{A}$, and $\widetilde{\mathbf{R}}_i$, $\mathbf{U}_i$, respectively, represent the $i-$ th row of $\widetilde{\mathbf{R}}$, $\mathbf{U}$. By Cauchy–Schwarz inequalities [11], the following equation is obtained:

$$\mathbf{A}_{ii} \leq \left\|\widetilde{\mathbf{R}}_i\right\|\left\|\mathbf{U}_i\right\| = 1, \quad \text{when } \widetilde{\mathbf{R}}_i = \mathbf{U}_i. \tag{19}$$

So formula (18) can be written as formula (20):

$$\mathrm{Loss}_2(\mathbf{R}) = \max\left[\mathrm{tr}\left(\mathbf{A}\mathbf{\Omega}\right)\right] = \max\sum_{i=1}^{n}\mathbf{A}_{ii}\mathbf{\Omega}_{ii}. \tag{20}$$

Combining formula (19), when $\widetilde{\mathbf{R}}_i = \mathbf{U}_i$, formula (20) takes the maximum value. For $\widetilde{\mathbf{R}}_i = \mathbf{U}_i$, we can get the following formula:

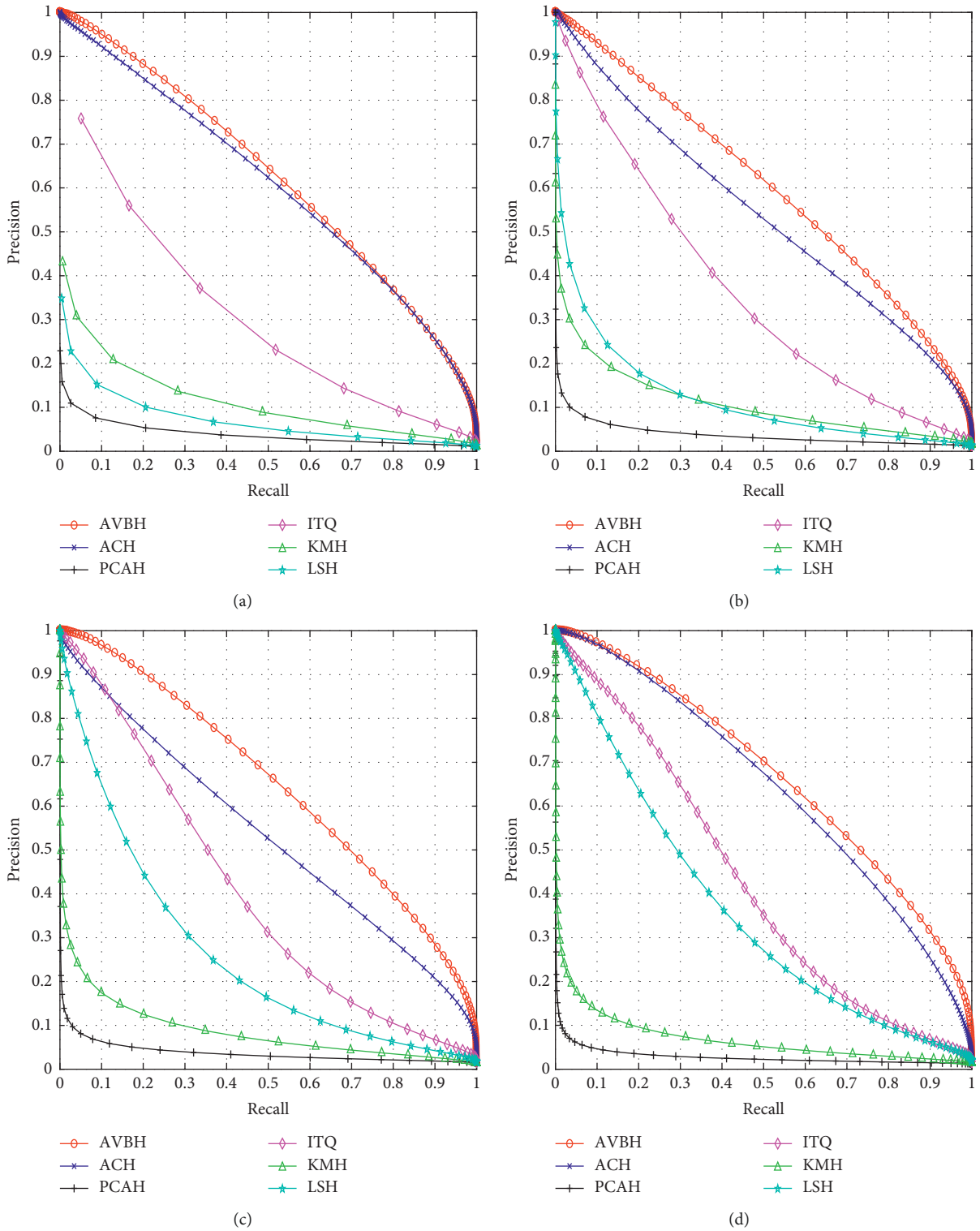

Figure 2: Comparison of the precision-recall experiment of different bits encoding under Cifar10: (a) Cifar10, 16 bit; (b) Cifar10, 32 bit; (c) Cifar10, 64 bit; (d) Cifar10, 128 bit.

$$\begin{aligned} \widetilde{\mathbf{R}}_i &= \mathbf{U}_i \\ \Longleftrightarrow \widetilde{\mathbf{R}} &= \mathbf{U} \\ \Longleftrightarrow \mathbf{RC} &= \mathbf{U} \\ \Longleftrightarrow \mathbf{RCC}^T &= \mathbf{UC}^T \\ \Longleftrightarrow \mathbf{R} &= \mathbf{UC}^T, \end{aligned} \tag{21}$$

when $\mathbf{R} = \mathbf{UC}^T$, formula (16) takes the maximum value, and formula (15) takes the minimum value. As a result, we can get the optimal result by formula (21).

3.4. Encoding Functions

3.4.1. Dataset Encoding. When the objective function value converges, we can get the mapping function $G(\mathbf{y}_1)$ of AVBH to the dataset according to formula (14), where $\mathbf{y}$ is the random Fourier feature (RFF) obtained by the mapping stage of the sample point $\mathbf{x}$:

$$\begin{aligned} G(\mathbf{y}) &= \left[g_1(\mathbf{y}), g_2(\mathbf{y}), \ldots, g_{k_L}(\mathbf{y})\right] \\ &= \operatorname{sign}\left(\sum_{m=1}^{(n/k_l)} \left(\mathbf{R}^T\right)^{lm} \mathbf{y}\right), \quad l \in [1, L]. \end{aligned} \tag{22}$$

3.4.2. Query Point Encoding. The optimal rotation matrix $\mathbf{R}$ can be obtained from the training process of dataset coding. For the data $\mathbf{q}$ in the query set, the main goal of encoding is to keep as much accurate information as possible, so the query set encoding does not need to be compressed and mapped to the hash code of the length bit. Combining formula (14), we can get the mapping function of AVBH to the query set:

$$F(\mathbf{q}) = \left[f_1(\mathbf{q}), f_2(\mathbf{q}), \ldots, f_n(\mathbf{q})\right] = \operatorname{sign}\left(\mathbf{R}^T\mathbf{q}\right). \tag{23}$$

3.5. Convergence Analysis of AVBH. According to the objective function (8), we can get the following formula:

$$\begin{aligned} \operatorname{Loss}(\mathbf{B}, \mathbf{R}) &= \min\left\|\mathbf{B} - \mathbf{R}^T\mathbf{Y}\right\|_F^2 \\ &= \min\left\|\mathbf{B} - \mathbf{D} - \left(\mathbf{R}^T\mathbf{Y} - \mathbf{D}\right)\right\|_F^2 \\ &\le \min\left(\|\mathbf{B} - \mathbf{D}\|_F^2 + \left\|\mathbf{R}^T\mathbf{Y} - \mathbf{D}\right\|_F^2\right) \\ &= \min\|\mathbf{B} - \mathbf{D}\|_F^2 + \min\left\|\mathbf{D} - \mathbf{R}^T\mathbf{Y}\right\|_F^2 \\ &= L_1(\mathbf{B}) + L_2(\mathbf{R}), \end{aligned} \tag{24}$$

where $\mathbf{D}$ is a $n \times N$ constant matrix and satisfies the following two conditions: (1) signs of each element, i.e., positive or negative, in $\mathbf{D}$ are the same as that in $\mathbf{R}^T\mathbf{Y}$, and (2) each element in $\mathbf{D}$ is not greater than the corresponding position element in $\mathbf{R}^T\mathbf{Y}$.

Therefore, the optimization goal for $\operatorname{Loss}(\mathbf{B}, \mathbf{R})$ is transformed into the two suboptimization problems, i.e., $\operatorname{Loss}_1(\mathbf{B})$ and $\operatorname{Loss}_2(\mathbf{R})$. Specifically, for the subproblem $\operatorname{Loss}_1(\mathbf{B})$, formula (14) gives the optimal solution. Therefore, it can be guaranteed that the updated value of formula (14) is less than or equal to the value obtained before. For the subproblem $\operatorname{Loss}_2(\mathbf{R})$, formula (21) gives the optimal solution. Therefore, it also can be guaranteed that the updated value of formula (21) is less than or equal to the value obtained before.

Combining the two parts, the combination of (14) and (21) can guarantee that the updated value is less than equal to the value obtained before the update. We can conclude the AVBH algorithm is convergence.

4. Experimental Results

The experimental computer has the Intel Core i5-2410M CPU and 8 GB DDR3 memory. We compared the performance of AVBH with that of several typical hashing methods: ACH [12], ITQ [19], KMH [20], PCAH [21], and LSH [13].

4.1. Experimental Datasets

4.1.1. CIFAR-10[1]. It is a set of 60,000-32×32 colour images in 10 categories with each category containing 6,000 images. In this experiment, 320-D gist features were extracted for each image in the dataset. We randomly selected 1,000 images as the test data and the remaining 59,000 as the training data. In the training data, the closest 50 data points (based on the Euclidean distance) from a test data point were regarded as its nearest neighbours.

4.1.2. SIFT[2]. It is a local SIFT feature set containing 1,000,000 128-D images. 100,000 of these sample images were randomly selected as the training data and 10,000 of other sample images as the test data.

4.1.3. GIST[3]. It is a global GIST feature set containing 1,000,000 960-D images. 500,000 of these sample images were randomly selected as the training data and 1,000 sample images as the test data.

4.2. Performance Evaluation. The performance of AVBH was evaluated mainly by the relationship between the accuracy of the query (precision) and the recall rate (recall). We define

$$\begin{aligned} \text{precision} &= \frac{\text{true positive results}}{\text{true positive results} + \text{false positive results}}, \\ \text{recall} &= \frac{\text{true positive results}}{\text{true positive results} + \text{false negative results}}. \end{aligned} \tag{25}$$

For the sake of fairness, the average encoding length of AVBH was set to be the encoding length of other methods under a given dataset.

Figures 2–4 show precision-recall curves for Euclidean neighbour retrieval for several methods on CIFAR-10, SIFT, and GIST with Euclidean neighbour ground truth.

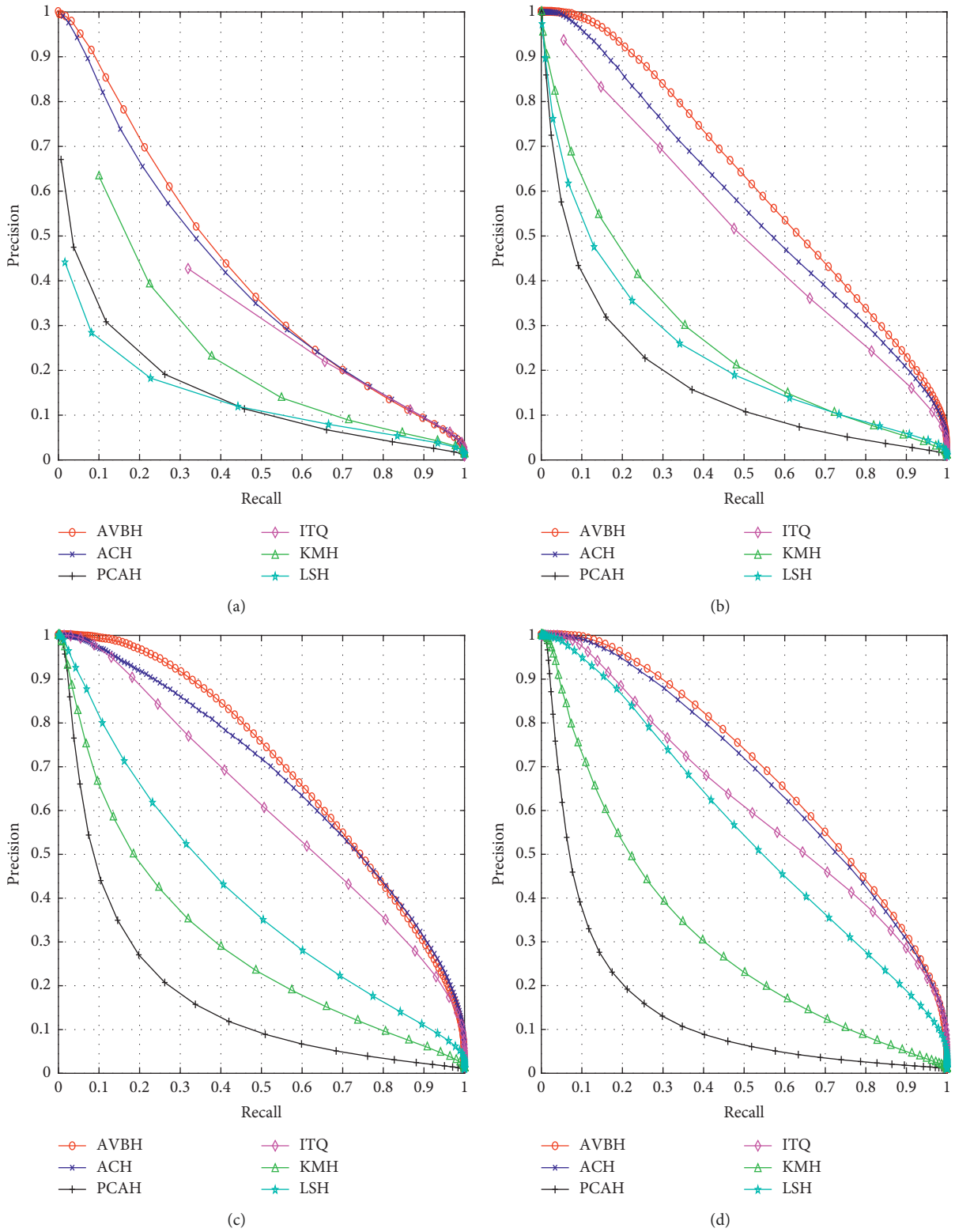

Figure 3: Comparison of the precision-recall experiment of different bits encoding under SIFT: (a) SIFT, 16 bit; (b) SIFT, 32 bit; (c) SIFT, 64 bit; (d) SIFT, 128 bit.

Our method, AVBH, can get a better precision performance on the four datasets. As the AVBH algorithm uses the variable bit code, the total code length is less than other algorithms. As a result, our method effectively reduces the cost of storage and improves the accuracy of query.

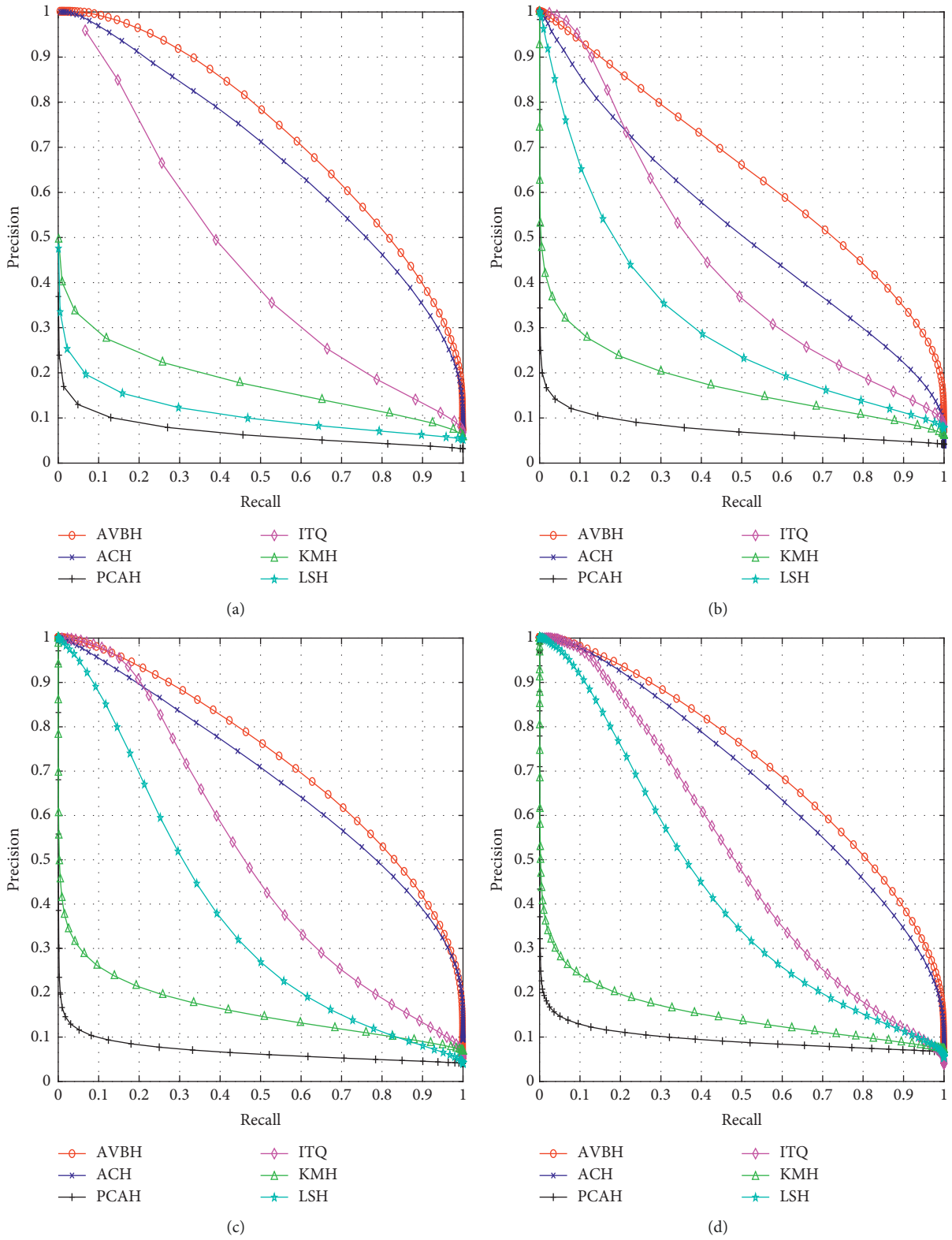

Figure 4: Comparison of the precision-recall experiment of different bits encoding under GIST: (a) GIST, 16 bit; (b) GIST, 32 bit; (c) GIST, 64 bit; (d) GIST, 128 bit.

5. Conclusion

In this paper, an asymmetric learning to hash with variable bit encoding algorithm was proposed. By the frequency statistics of the random Fourier feature encoding for the dataset, we compress high-frequency hash codes into longer encoding representations and low-frequency hash codes into shorter encoding representations. For a query data, we quantize to a long bit hash code and compare with the same length cascade concatenated data point to retrieve the nearest neighbours. This ensures that the original data information can be preserved as much as possible while the data are compressed, which maximizes the balance between coding compression and query performance. Experiments on open datasets show that the proposed algorithm effectively reduces the cost of storage and improves the accuracy of the query. As we use a two-stage algorithm framework for the hash codes generating, the training stage costs a lot of time. In future work, we will work on simplifying the training process.

Data Availability

The datasets for the experiment of this paper are as follows. (1) CIFAR-10 (available at http://www.cs.toronto.edu/~kriz/cifar.html): it is a set of 60,000-32×32 colour images in 10 categories with each category containing 6,000 images. In this experiment, 320-D gist features were extracted for each image in the dataset. We randomly selected 1,000 images as the test data and the remaining 59,000 as the training data. In the training data, the closest 50 data points (based on the Euclidean distance) from a test data point were regarded as its nearest neighbours. (2) SIFT (available at http://corpus-texmex.irisa.fr): it is a local SIFT feature set containing 1,000,000 128-D images. 100,000 of these sample images were randomly selected as the training data and 10,000 of other sample images as the test data. (3) GIST (available at http://corpus-texmex.irisa.fr): it is a global GIST feature set containing 1,000,000 960-D images. 500,000 of these sample images were randomly selected as the training data and 1,000 sample images as the test data.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by Zhejiang NSF (Grant nos. LZ20F020001 and LY20F020009) and China NSF (Grant nos. 61472194 and 61572266) as well as programs sponsored by K. C. Wong Magna Fund in Ningbo University.

References

[1] C. S. Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, Anchorage, AK, USA, June 2008.

[2] F. Shen, X. Yan, L. Li, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 40, no. 12, pp. 3034–3044, 2018.

[3] X. Qu, W. Yi, T. Wang, S. Wang, L. Xiao, and Z. Liu, "Mixed-integer linear programming models for teaching assistant assignment and extensions," *Scientific Programming*, vol. 2017, Article ID 9057947, 7 pages, 2017.

[4] Z. Zhang, Q. Zou, Q. Wang, Y. Lin, and Q. Li, "Instance similarity deep hashing for multi-label image retrieval," 2018, https://arxiv.org/abs/1803.02987v1.

[5] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 437–451, 2018.

[6] S. Berchtold, D. A. Keim, and H. P. Kriegel, "The X-tree: an index structure for high- dimensional data," in *Proceedings of the 22nd International Conference on Very Large Data Bases*, pp. 28–39, Mumbai, India, 1996.

[7] X. Shen, W. Liu, I. W. Tsang, Q.-S. Sun, and Y.-S. Ong, "Multilabel prediction via cross-view search," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4324–4338, 2018.

[8] X. Shen, F. Shen, Q.-S. Sun, Y. Yang, Y.-H. Yuan, and H. T. Shen, "Semi-paired discrete hashing: learning latent hash codes for semi-paired cross-view retrieval," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4275–4288, 2016.

[9] L. K. Huang, Q. Yang, and W. S. Zheng, "Online hashing," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 29, no. 6, pp. 2309–2322, 2017.

[10] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing," in *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010*, IEEE, San Francisco, CA, USA, June 2010.

[11] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik, "Asymmetric distances for binary embeddings," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 36, no. 1, pp. 33–47, 2014.

[12] Y. Lv, W. W. Y. Ng, Z. Zeng, D. S. Yeung, and P. P. K. Chan, "Asymmetric cyclical hashing for large scale image retrieval," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1225–1235, 2015.

[13] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*, vol. 8, no. 2, pp. 518–529, Edinburgh, Scotland, 1999.

[14] J. Qian, Q. Zhu, and H. Chen, "Multi-granularity locality-sensitive bloom filter," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3500–3514, 2015.

[15] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3893–3903, 2018.

[16] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[17] W. Kong and W.-J. Li, "Isotropic hashing," in *Advances in Neural Information Processing Systems*, pp. 1655–1663, Curran Associates Inc., New York, NY, USA, 2012.

[18] M. Raginsky, "Locality-sensitive binary codes from shift-invariant kernels," *Advances in Neural Information Processing*

Systems, pp. 1509–1517, Curran Associates Inc., New York, NY, USA, 2009.

[19] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[20] K. He, F. Wen, and J. Sun, "K-means hashing: an affinity-preserving quantization method for learning binary compact codes," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, June 2013.

[21] X. Yu, X. Zhang, B. Liu, L. Zhong, and D. N. Metaxas, "Large scale medical image search via unsupervised PCA hashing," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 393–398, IEEE, Portland, OR, USA, June 2013.