Neurocomputing 171 (2016) 866–877

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Adaptive bit allocation product quantization

Qin-Zhen Guo [a,*], Zhi Zeng [a], Shuwu Zhang [a], Guixuan Zhang [a], Yuan Zhang [b]

[a] Institute of Automation, Chinese Academy of Sciences, 95 Zhongguancun East Road, 100190 Beijing, China
[b] Samsung Electronics, 12a TaiYangGong Middle Road, 100028 Beijing, China

A R T I C L E I N F O

Article history:
Received 9 April 2015
Received in revised form
11 July 2015
Accepted 11 July 2015
Communicated by Shaoting Zhang
Available online 8 August 2015

Keywords:
Vector quantization
Product quantization
Adaptive bit allocation
Approximate nearest neighbor search
Image retrieval

A B S T R A C T

Product quantization (PQ) is a popular vector quantization method for approximate nearest neighbor search. The key idea of PQ is to decompose the original data space into the Cartesian product of some low-dimensional subspaces and then every subspace is quantized separately with the same number of codewords. However, the performance of PQ depends largely on the distribution of the original data. If the energies of subspaces are extremely unbalanced, PQ will achieve bad results. In this paper, we propose an adaptive bit allocation product quantization (BAPQ) method to deal with the problem of unbalanced energies of subspaces in PQ. In BAPQ, we adaptively allocate different numbers of codewords (or bits) to subspaces to quantize data for minimizing the total quantization distortion. The essence of our method is to find the optimal bit allocation scheme. To this end, we formulate an objective function about minimizing quantization distortion with respect to bit allocation scheme and adopt a greedy algorithm to find the near-optimal solution. BAPQ can achieve lower quantization distortion than PQ and optimized product quantization (OPQ). Besides, both bias and variance of difference between the true distance and the BAPQ's estimated distance are reduced from those of PQ and OPQ. Extensive experiments have verified the superiority of BAPQ over state-of-the-art approaches.

1. Introduction

Approximate nearest neighbor (ANN) search has been widely used to avoid excessive computational and memory cost of nearest neighbor (NN) search in many computer vision problems, like image retrieval [1], image classification [2], 3D reconstruction [3] and other related application areas. The key idea of ANN is to find the nearest neighbors with high probability. Nowadays, several ANN approaches have been developed including tree-based methods, hashing-based methods and vector quantization.

The tree-based methods [4,14] usually recursively partition the data space to implement an efficient search for low-dimensional data. However, for high-dimensional data, in the worst case the tree-based methods can degenerate to exhaustive search.

The hashing-based methods [5,6,15–20,23,24] which map the data into Hamming space have been popular approaches to ANN search. The similarity between two vectors is measured by the Hamming distance of their hashing codes. Hashing-based methods have two advantages. One is that the needed storage is largely reduced. The other is that the Hamming distance between two codes can be computed efficiently by XOR operator followed by bit count. But due to the thick boundary of Hamming space, these methods usually cannot achieve ideal results [7].

Vector quantization (VQ) [8] is an effective and efficient method for ANN search. These methods quantize the data by codewords in order to reduce the cardinality of the data space. Among the VQ methods, product quantization (PQ) [9] is designed to decompose the original data space into the Cartesian product of several low-dimensional subspaces and each subspace is quantized separately. The distance between two vectors can be computed by the sum of the distances between their subvectors in the same subspace.

However, PQ considers every subspace having the same energy and uses the same number of codewords to quantize every subspace, which is unreasonable since the energies of subspaces may be not identical, especially when the data is principal component analysis (PCA) projected. The subspaces formed by dimensions with larger variance carry more energy. If we use the same number of codewords as that in subspaces with more energy to quantize data in subspaces with less energy, it will be redundant. And the quantization distortion will be expanded if we use the same number of codewords as that in subspaces with less energy to quantize the data in subspaces with more energy.

Motivated by the above considerations, in this paper, we propose an adaptive bit allocation product quantization (BAPQ) approach for ANN search. The key idea of BAPQ is that we adaptively allocate different numbers of codewords (or bits) to subspaces to minimize

* Corresponding author. Tel.: +86 18810461353.
E-mail addresses: qinzhen.guo@ia.ac.cn (Q.-Z. Guo), zhi.zeng@ia.ac.cn (Z. Zeng), shuwu.zhang@ia.ac.cn (S. Zhang), guixuan.zhang@ia.ac.cn (G. Zhang), zhang.yuan09@gmail.com (Y. Zhang).

http://dx.doi.org/10.1016/j.neucom.2015.07.062
0925-2312/© 2015 Elsevier B.V. All rights reserved.

the quantization distortion. (Since the number of bits needed to index the codewords is positively related to the number of codewords, in the following, we use both terms to indicate the number of codewords, which is also called the size of codebooks.) In BAPQ, subspaces with more energy will be allocated with more bits while subspaces with less energy will be allocated with fewer bits. BAPQ can achieve lower quantization distortion than PQ and its improved version [10] and give more accurate distance estimation. Extensive experiments have demonstrated the superiority of our method. The rest of the paper is organized as follows. In Section 2, we briefly review the related work to product quantization. In Section 3, we describe the details of our BAPQ method. Experimental results are presented in Section 4. The paper is concluded in Section 5.

2. Related work

2.1. Vector quantization

Given a set of n vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, the purpose of vector quantization (VQ) is to reduce the cardinality of data, especially when the data is real-valued. In VQ, the original data is represented by the reconstruction values. The reconstruction values c_i are called codewords, which form the codebook $C = \{c_1, \ldots, c_k\}$ where k is the size of the codebook. The codewords in C can be index by $l = \log_2 k$ bits. In the following discussion of the paper, we assume that the total bit length l is fixed. That means the size of the codebook C is fixed to $k = 2^l$.

The quantization distortion of VQ is

$$E = \sum_{i=1}^{n} \|\boldsymbol{x}_i - c(\boldsymbol{x}_i)\|^2 \tag{1}$$

$\|\boldsymbol{x}-\boldsymbol{y}\|$ denotes the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. $c(\boldsymbol{x}_i)$ is the codeword of $\boldsymbol{x}_i$.

In order to minimize the quantization distortion, there are two properties known as the Lloyd optimality conditions to be satisfied. First, a vector $\boldsymbol{x}$ must be quantized to its nearest codeword. That is,

$$c(\boldsymbol{x}) = \underset{c_j}{\operatorname{argmin}} \|\boldsymbol{x} - c_j\| \tag{2}$$

Thus the whole data space will be partitioned by C into k Voronoi cells. The second condition is that, a codeword must be the expectation of the vectors lying in the same Voronoi cell. In practice, we use Lloyd's algorithm, known as k-means, to find a near-optimal codebook by iteratively assigning the vectors to codewords and re-estimating the codewords from the assigned vectors.

2.2. Product quantization

In vector quantization, if we use $l = 64$ bits to encode the data, the size of the codebook k will be 2^{64}. It is impossible to use k-means to find the codebook in terms of both time complexity and memory usage. Product quantization (PQ) [9] is proposed to handle this issue. The basic idea of PQ is that the original data is decomposed into m subspaces of dimension $q = d/m$.

$$\overbrace{x_1, \ldots, x_q}^{x^1}, \ldots, \overbrace{x_{d-q+1}, \ldots x_d}^{x^m} \tag{3}$$

$\boldsymbol{x}^i$ is the ith subvector formed by from the $(i{*}q - q + 1)$th dimension to the $(i{*}q)$th dimension of $\boldsymbol{x}$.

In each subspace, PQ uses k-means to find $2^{l/m}$ codewords which are indexed by l/m bits to form the codebook of each subspace. The codebook of the ith subspace is $C^i = \{c^i_1, \ldots, c^i_{k'}\}$, where $k' = 2^{l/m}$ is the size of codebooks in subspaces. The codebook of the whole data space C will be the Cartesian product of all the subspaces i.e. $C = C^1 \times C^2 \times \cdots \times C^m$. In each subspace, the subvector of $\boldsymbol{x}$ is quantized by the codebook in corresponding subspace,

$$c^i(x^i) = \underset{c^i_j}{\operatorname{argmin}} \|x^i - c^i_j\| \tag{4}$$

where $c^i(x^i)$ is the codeword of $\boldsymbol{x}^i$ in the ith subspace. Then the whole vector $x = (x^1, \ldots, x^m)$ can be quantized to $(c^1(x^1), \ldots, c^m(x^m))$ and encoded by the concatenation of the index of the codeword that quantizes $\boldsymbol{x}$ in each subspace, i.e. $I(x) = (I(c^1(x^1)), \ldots, I(c^m(x^m)))$, which is a $m{*}\log_2 k' (= l)$ bits binary code. $I(c^i_j)$ is the index of codeword c^i_j in the ith subspace. The memory usage of the whole codebook and the assignment complexity are largely reduced from that of k-means (see Table 1).

For computing the distance between the vector $\boldsymbol{x}$ in database and the query $\boldsymbol{y}$, there are two types of distance for PQ: symmetric distance computation (SDC) and Asymmetric distance computation (ADC), which are defined as follows:

$$d_{SDC} = \sqrt{\sum_{i=1}^{m} \|c^i(x^i) - c^i(y^i)\|^2} \tag{5}$$

$$d_{ADC} = \sqrt{\sum_{i=1}^{m} \|c^i(x^i) - y^i\|^2} \tag{6}$$

For SDC, both $\boldsymbol{x}$ and $\boldsymbol{y}$ are quantized while for ADC only the vectors in database are quantized. Actually, d_{SDC} and d_{ADC} can be computed fast by looking up table if we precompute $d(c^i(x^i), c^i(y^i))$ and $d(c^i(x^i), y^i)$ in each subspace prior to the search. Besides, when computing the distances, we do not need to read the original database into the memory. We only need to read the codebook of each subspace and the index into the memory, which largely reduces the storage.

Table 1
Memory usage of the codebook and assignment complexity for different quantization methods.

	Memory usage	Assignment complexity
k-means	kd	kd
PQ/OPQ	$k'd$	$k'd$
TC	$\sum_{i=1;b_i>0}^{d} 2^{b_i}$	$\sum_{i=1;b_i>0}^{d} 2^{b_i}$
BAPQ	$\sum_{i=1;l_i>0}^{m} 2^{l_i} \cdot \frac{d}{m}$	$\sum_{i=1;l_i>0}^{m} 2^{l_i} \cdot \frac{d}{m}$

2.3. Optimized product quantization

In PQ, the problem of optimal space decomposition is not handled and the performance of PQ depends largely on the distribution of data. If the energies of subspaces are extremely unbalanced, PQ will achieve bad results. Optimized product quantization (OPQ) [10] addresses the problem of optimal space decomposition in PQ and minimizes quantization distortion with respect to the space decomposition and the codebooks.

$$\begin{aligned} &\min_{\boldsymbol{R}, C^1, \ldots, C^m} \sum_{i=1}^{n} \|\boldsymbol{x}_i - c(\boldsymbol{x}_i)\|^2 \\ &\text{s.t. } c \in C = \left\{ c \,\middle|\, Rc \in C^1 \times \cdots \times C^m,\ \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} = \boldsymbol{I} \right\} \end{aligned} \tag{7}$$

where $\boldsymbol{R}$ is an orthogonal matrix and $\boldsymbol{I}$ is an identity matrix.

In OPQ, the authors optimize the projection matrix and codebooks by minimizing the quantization distortion with two solutions: a non-parametric solution and a parametric solution. The details of the two methods can be found in [10]. For the parametric solution, the authors propose a simple Eigenvalue Allocation method. In Eigenvalue Allocation, original vectors firstly are projected by PCA and the eigenvalues λ_i are sorted with descending order $\lambda_1 \geq \cdots \geq \lambda_d$. Then the largest eigenvalues are sequentially picked up and allocated to the subspace which does not reach the max-size of d/m and has the minimum product of the eigenvalues in it. Re-ordering the eigenvectors according to the orders of the corresponding eigenvalues will result in the projection matrix $\boldsymbol{R}$. After obtaining $\boldsymbol{R}$, we can perform k-means in each subspace of the transformed space to get the codebook and then do the same following procedure of PQ. The energies of subspaces in OPQ are more balanced than that in PQ and OPQ achieves the state-of-the-art performance.

2.4. Transform coding

Transform coding (TC) based vector quantization [21] is a novel quantization method. In TC, to minimize the quantization distortion, different numbers of bits are allocated to different dimensions. Assuming that each dimension is identically distributed after normalizing the variance and that the per-dimension quantization distortion functions are identical, the optimal bit allocation is achieved when

$$b_i \sim \log_2 \sigma_i \tag{8}$$

where b_i is the bit number allocated to the ith dimension and σ_i is the standard deviation of the ith dimension [22]. To reduce statistical dependence among the dimensions, TC uses PCA to transform the data before bit allocation. For TC, there are also two types of distance computation: ADC and SDC. Details can be seen in [21].

2.5. Adaptive bit allocation hashing

Adaptive bit allocation hashing (ABAH) [17] is a hashing method for approximate nearest neighbor search. In [17], different numbers of bits are adaptively allocated to dimensions.

$$b_p = \begin{cases} \left\lfloor l \cdot \dfrac{v_p}{\sum_{i=p}^{d} v_i} + 0.5 \right\rfloor & p = 1 \\ \left\lfloor \left(l - \sum_{i=1}^{p-1} b_i\right) \cdot \dfrac{v_p}{\sum_{i=p}^{d} v_i} + 0.5 \right\rfloor & p \geq 2 \end{cases} \tag{9}$$

where b_p and v_p are the bit number and variance of the pth dimension respectively.

To preserve the neighborhood structure of the Euclidean space, important dimensions in Euclidean space should be also important in Hamming space and ABAH allocates more bits to important dimensions to achieve that. In ABAH, the purpose of bit allocation is, in Hamming space, to preserve the importance of dimensions in Euclidean space rather than minimize the quantization distortion. Our BAPQ method is a vector quantization method following PQ (see Section 3). In BAPQ, the original space is decomposed into the Cartesian product of several low-dimensional subspaces and each subspace is quantized separately. To minimize the quantization distortion, we use more codewords to quantize the subspaces with more energy. The purpose of bit allocation of BAPQ is to minimize the quantization distortion. Besides, the bit allocation in ABAH is heuristic. While in BAPQ, we obtain the optimal (near-optimal) bit allocation in term of minimizing the quantization distortion.

3. Proposed method

In both PQ and OPQ, the size of codebook in each subspace is the same and each subspace is allocated with the same number of bits (l/m bits for one subspace). However, since the energies of subspaces of the data are not identical, allocating the same number of bits to each subspace to perform PQ will lead to poor results.

In this paper, we propose an adaptive bit allocation product quantization (BAPQ) method to allocate different numbers of bits to subspaces to minimize the quantization distortion. We use PCA to preprocess the data to find the principal components. (We just use PCA to rotate the data rather than reduce dimensionality.) Please note that the discussion of BAPQ below is based on PCA projected data.

The basic idea of our method is that firstly we decompose the whole space into several subspaces. Then by minimizing the quantization distortion of the training data, we obtain a near-optimal bit allocation scheme to allocate different numbers of bits to subspaces. According to the near-optimal bit allocation scheme, we compute the codebook of each subspace by k-means. Finally, we use the codebook of each subspace to quantize the data in corresponding subspace separately.

3.1. Motivation

Quantization distortion has been identified to be closely related to the search accuracy [10]. The purpose of our method is still to minimize the quantization distortion in Eq. (1). As stated above, it is unpractical to directly use k-means to find the 2^l codewords, especially for a large value of l. We follow the idea of PQ to use product quantizers to obtain the codebook. That is, we decompose the whole space into m subspaces and compute the codebook of each subspace separately. The whole codebook C will be the Cartesian product of the subspaces' codebooks i.e. $C = C^1 \times C^2 \times \cdots \times C^m$, where C^j is the codebook of the jth subspace, $1 \leq j \leq m$.

Then the whole quantization distortion in Eq. (1) will be

$$\begin{aligned} E &= \sum_{i=1}^{n} \|x_i - c(x_i)\|^2 \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} (x_i^j - c^j(x_i))^2 \\ &= \sum_{j=1}^{m} \sum_{i=1}^{n} (x_i^j - c^j(x_i))^2 \end{aligned} \tag{10}$$

$\sum_{i=1}^{i} (x_i^j - c^j(x_i))^2$ can be seen as the quantization distortion in the jth subspace. And the total quantization distortion of the whole data is the sum of the distortions of all the subspaces.

The quantization distortion of each subspace is related to the size of the codebook of corresponding subspace. The larger the size of the codebook is, the less the quantization distortion we can obtain. Our method is to adaptively control the size of the codebook of each subspace to minimize the total quantization distortion.

3.2. Adaptive bit allocation product quantization

For the jth subspace of the data, assume that l_j bits are allocated to this subspace. Then the quantization distortion of the jth subspace is

$$E_j = \sum_{i=1}^{n} (x_i^j - c^j(x_i^j))^2 \tag{11}$$

where $c^j(x_i^j)$ is the codeword in C^j that quantizes x_i^j. The size of C^j is 2^{l_j}. Please note that $c^j(x_i^j)$ in Eq. (11) is the codeword that quantizes x_i^j in the jth subspace while $c^j(x_i)$ in Eq. (10) is the jth subvecor of the codeword that quantizes x_i in the whole space. However, since we use the Cartesian product of codebooks of subspaces to construct the codebook of the whole data, these two terms are the same in our method.

Then, the total quantization distortion will be

$$E = \sum_{j=1}^{m} E_j \tag{12}$$

and we can minimize the following objective function Eq. (13) to get the optimal bit allocation scheme:

$$(l_1, \ldots, l_m) = \arg \min_{(l_1,\ldots,l_m)} \sum_{j=1}^{m} E_j$$
$$\text{s.t.} \sum_{j=1}^{m} l_j = l \tag{13}$$

where l is the total bit length. Directly optimizing Eq. (13) is intractable due to the implicit relation between the bit allocation scheme $(l_1, \ldots, l_m)$ and $\boldsymbol{E}$. Using brute force to find the optimal solution is also unpractical because the complexity is $\binom{m+l-1}{m-1}$. In this paper, we employ a greedy algorithm to find the near-optimal solution.

We initiate $(l_1,\ldots,l_m)=(0,\ldots,0)$. Our objective is to assign l bits to m subspaces. For one of the l bits, firstly we tentatively assign this bit to the first subspace, i.e. $(l_1+1,\ldots,l_m)=(1,\ldots,0)$. We use k-means to obtain the codebook of each subspace and compute the quantization distortion of each subspace by Eq. (11) and the total quantization distortion E^1 by Eq. (12). Then we tentatively assign this bit to the second subspace, i.e. $(l_1,l_2+1,\ldots,l_m)=(0,1,\ldots,0)$ and compute the total quantization distortion E^2. After testing all the m subspaces, we get m total quantization distortion, $E^1,\ldots,E^m$. Finally, we assign this bit to the subspace with the minimum total quantization distortion. We will obtain the near-optimal bit allocation after we process all the l bits. Algorithm 1 presents the procedure of our greedy algorithm. Actually, we do not need to compute all the m subspaces' quantization distortion in Eq. (12) when assigning one bit to subspaces. In Step 7 of Algorithm 1, we can save E_j for fast computation in the following loops. In practice, if the bit length of one subspace is big, the size of the subcodesbooks in this subspace will be large. We can limit the maximum bits of each subspace to avoid this problem.

As a byproduct, the codebooks of subspaces also can be obtained in Algorithm 1. After obtaining the codebooks of subspaces, we quantize the subvector of x_i with the codebook of corresponding subspace by Eq. (4). Then $x_i = (x_i^1, \ldots, x_i^m)$ can be quantized to $x_i = (c^1(x_i^1), \ldots, c^m(x_i^m))$.

For the query process, there are also two types of distance estimators same as Eqs. (5) and (6) in PQ. To compute the approximate Euclidean distance, our method only need m additions, which is the same as PQ. In practice, we do not need m additions. The reason is that, for subspaces whose bit length is zero (since we use PCA to preprocess the data, there will be many subspaces allocated with zero bit), those subspaces will be quantized by the mean of corresponding subspaces and thus all the vectors in database will have the same value in those subspaces. No matter for SDC or ADC, subspaces with zero bit do not affect the ranking. So we do not need to compute all the m subspaces. We only need to compute subspaces whose bit length is bigger than zero. And, even, we do not need to store the codebooks of those subspaces and also do not need to compute the distance look up tables for those subspaces prior to search.

Algorithm 1. Adaptive bit allocation

```
1. Input: a training set $\{\boldsymbol{x_i}\}$, total bit length $l$.
2. Output: the optimal bit allocation scheme $(l_1,\ldots,l_m)$ and
   codebooks $C^j$ for $1 \le j \le m$.
3. Initialize: $(l_1,\ldots,l_m)=(0,\ldots,0)$.
4. For $i=1$ to $l$ do
5.     For $j=1$ to $m$ do
6.         $l_j=l_j+1$;
7.         According to the present bit allocation scheme $(l_1,\ldots,l_m)$, calculate the codebook $C^j_{new}$ by $k$-means and distortion
           $E_j$ of the $j$th subspace by Eq. (11); then calculate total
           quantization distortion $E^j$ by Eq. (12).
8.         $l_j=l_j-1$;
9.     End for
10.    Find the minimum $E^j$ for $1 \le j \le m$, i.e. $p=\min_j\{E^j\}$;
11.    $l_p=l_p+1$, $C^p=C^p_{new}$.
12. End for.
```

3.3. Discussion

The PQ-based methods are faced with the unbalanced energies of subspaces. The original PQ approach uses the random dimension combination to balance the energies of subspaces. OPQ minimizes the lower bound of product quantizer to find an orthogonal matrix to rotate the data. BAPQ deals with the problem in another angle. We fully use the unbalance of subspaces and adaptively allocate different numbers of bits to them. In BAPQ, subspaces with more energy will be allocated with more bits. Theoretically, it is hard to compare the quantization distortion of BAPQ to that of PQ/OPQ. Experimentally, our method can achieve lower quantization distortion than PQ and OPQ, which can be seen in Table 5.

Compared with PQ and OPQ, the memory cost of our BAPQ can be far less and the assignment complexity can be much lower. Since the data is rotated by PCA in BAPQ, the energy will be aggregated in the principal components, which can make us group less dimensions to form one subspace than PQ and allocate different numbers of bits to subspaces. The memory cost comparisons among PQ, OPQ, TC and BAPQ are presented in Table 1.

Fig. 1 illustrates the empirical probability distribution function of the difference between the true distance and the asymmetric distance estimated by PCA-PQ, RPQ, PQ, OPQ, TC and our BAPQ. We randomly sample 10,000 pairs of vectors from the GIST1M960 dataset. Table 2 shows both the bias and variance of all the compared methods. Since our method can adaptively allocate different numbers of bits to subspaces to minimize the quantization distortion, both the bias and variance of BAPQ are lower than those of PQ.

4. Experiments

In this section we compare our method with the state-of-the-art approaches for approximate nearest neighbor search task and image retrieval task.

4.1. Datasets

We compare our method with the state-of-the-art on the following public datasets.

- GIST1M960 [9]: A set of one million 960 dimensional GIST descriptors.
- GIST1M512: A set of one million 512 dimensional GIST descriptors which are extracted from FLICKR1M [11].
- SIFT1M [9]: A set of one million 128 dimensional SIFT descriptors.
- UKB [25]: The University of Kentucky Benchmark (UKB) contains 10,200 images of 2550 objects, 4 pictures correspond to each object, taken from different angles. The size of all the images is 640×480.

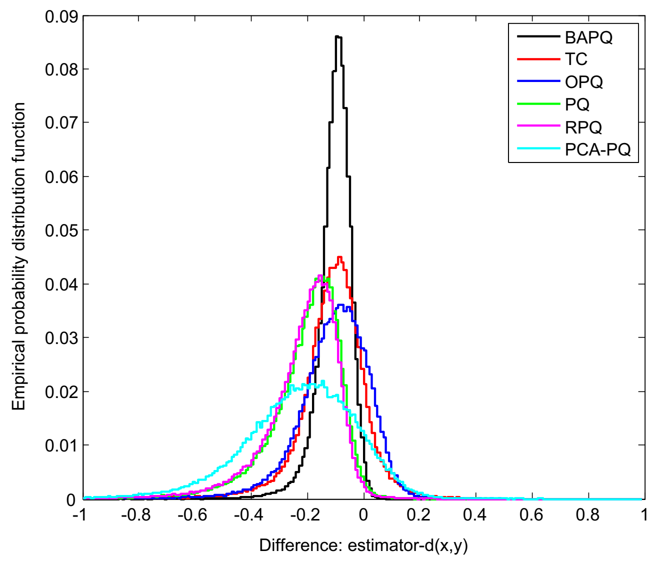


Fig. 1. The figure shows the empirical distribution of difference between the true distances and the estimated asymmetric distance from our BAPQ and other compared methods.

Table 2
Comparison of bias and variance of the difference between the true distance and the estimated asymmetric distance.

	BAPQ	TC	OPQ	PQ	RPQ	PCA-PQ
Bias	−0.1024	−0.1102	−0.1052	−0.2080	−0.2195	−0.2285
Variance	0.0035	0.0160	0.0168	0.0198	0.0208	0.0442

We follow the protocol in [17]. For all the experiments on GIST1M960, GIST1M512 and SIFT1M, we use 1000 queries that do not have any overlap with the dataset. The ground truth is defined by the 1000 nearest neighbors computed by the exhaustive, linear scan based on the Euclidean distance. The setting of 1000 nearest neighbors defined as ground truth is also used in [12] though PQ and OPQ use the 100 nearest neighbors as ground truth. However, actually, no matter 100 or 1000 nearest neighbors defined as ground truth, the comparisons among the compared methods are almost unchanged (see Section 4.5), which is also pointed out in [10]. The performance is measured by recall and mean Average Precision (mAP). For UKB, we represent each image with a 512 dimensional GIST descriptor. Each image is used in turn as query to search through the whole set. The accuracy is measured in terms of the number of relevant images retrieved in the top 4, i.e. $4 \times$ precision@4. For learning purpose on UKB, we use an independent image set (10,000 images randomly chosen from FLICKR1M) to learn the parameters.

4.2. Compared methods

To evaluate our method, we compare the following methods:

- PCA-PQ: For comparison, the original data is rotated by PCA.
- PQ: The dimensions of the original data are randomly ordered [9].
- RPQ: The original data is preprocessed by PCA and then rotated by a random orthogonal matrix [13].
- OPQ: The optimized product quantization with parametric solution [10].
- TC: A method that allocates different numbers of bits to dimensions [21].

Table 3
Comparison of BAPQ with different value of q using ADC on GIST1M960 in term of 1000-NN mAP.

Bits	16	32	64	128
$q=1$	0.0409	0.1382	0.2887	0.4691
$q=2$	0.0484	0.1666	0.3220	0.5144
$q=4$	0.0619	0.1909	0.3602	0.5543

Table 4
Bit allocation of BAPQ on GIST1M960.

Bits	Bit allocation
16	{8, 5 ,3, 0, …,0}
32	{11, 7, 6, 4, 4, 0, …,0}
64	{13, 9, 8, 7, 6, 5, 4, 4, 3, 3, 2, 0, …,0}
128	{15, 12, 11, 9, 8, 8, 7, 7, 6, 6, 6, 5, 5, 5, 4, 4, 4, 3, 3, 0, …,0}

Table 5
Comparison of quantization distortion on GIST1M960.

Bits	16	32	64	128
PCA-PQ	1.1941	1.1841	1.1545	1.0849
RPQ	1.1936	1.1824	1.1504	1.0738
PQ	1.1933	1.1747	1.1392	1.0678
OPQ	1.1915	1.1464	0.9481	0.5571
OPQ_BA	1.1417	1.0380	0.8024	0.5355
TC	1.1558	0.9868	0.7854	0.6010
BAPQ	1.0873	0.9021	0.7153	0.5279

Table 6
Comparison in term of 1000-NN mAP on GIST1M960.

Bits	16	32	64	128
OPQ	0.0282	0.0687	0.1313	0.4039
OPQ_BA	0.0484	0.1126	0.2438	0.5418
BAPQ	0.0619	0.1909	0.3602	0.5543

- BAPQ: Our BAPQ method that allocates different numbers of bits to subspaces.

Since the hashing based approximate nearest neighbor methods achieve inferior results to PQ based methods [10,12], we do not redundantly compare hashing methods with PQ based methods. We randomly sample 100,000 vectors from the database to train the parameters in the training stage for all the compared methods. We apply 100 iterations in the k-means clustering for all the methods. For PCA-PQ, PQ, RPQ, and OPQ, we assign 8 bits to each subspace as suggested in [9] which will result in 256 centers in each subspace and the number of subspaces m is $l/8$, where l is the total bit length.

We use an exhaustive search strategy in the query stage. That is we compute the Euclidean distances between the query and all the vectors in the database and vectors having the smallest N distances are returned as the results. In this paper, we do not combine our method with any non-exhaustive method like inverted files [9] since this is not the focus of this paper.

In our BAPQ method, we group q dimensions to form one subspace and therefore the number of subspaces is $m=d/q$. We observe that $q=4$ gives reasonable trade-off between effectiveness and efficiency. Experimental results with different value of q are given in Table 3. As we can see, the larger q is, the better the

Table 7
Comparison in term of Recall@1000 on GIST1M960.

Bits	16	32	64	128
OPQ	0.1238	0.2025	0.2997	0.5326
OPQ_BA	0.1678	0.2626	0.4004	0.6407
BAPQ	0.1988	0.3584	0.5051	0.6502

Table 8
Comparison of memory usage (floats) of the codebook on GIST1M960.

Bits	16	32	64	128
PQ/OPQ	245,760	245,760	245,760	245,760
TC	34	66	140	276
BAPQ	1184	9088	36,944	162,176

performance is. When $q=d$, BAPQ is equivalent to performing k-means on the whole data. We also test $q=8$ and find that it achieves better results than $q=4$. The mAP improvement of $q=8$ over that of $q=4$ is 30.3% for 16 bits and 14.8% for 32 bits when using ADC. But the training time of $q=8$ is much longer than that of $q=4$. Besides, the memory usage and assignment complexity of $q=8$ will be far more than that of $q=4$. Taking both effectiveness and efficiency into consideration, we set $q=4$ in the experiments.

In Table 4, we present the bit allocation of BAPQ on GIST1M960. As we can see, more bits are allocated to subspaces with more energy while fewer bits are allocated to subspaces with less energy.

4.3. Distortion and memory usage

As stated in [10], the quantization distortion is tightly related to the ANN search accuracy. The quantization distortion comparison is presented in Table 5. We can see that our BAPQ method achieves

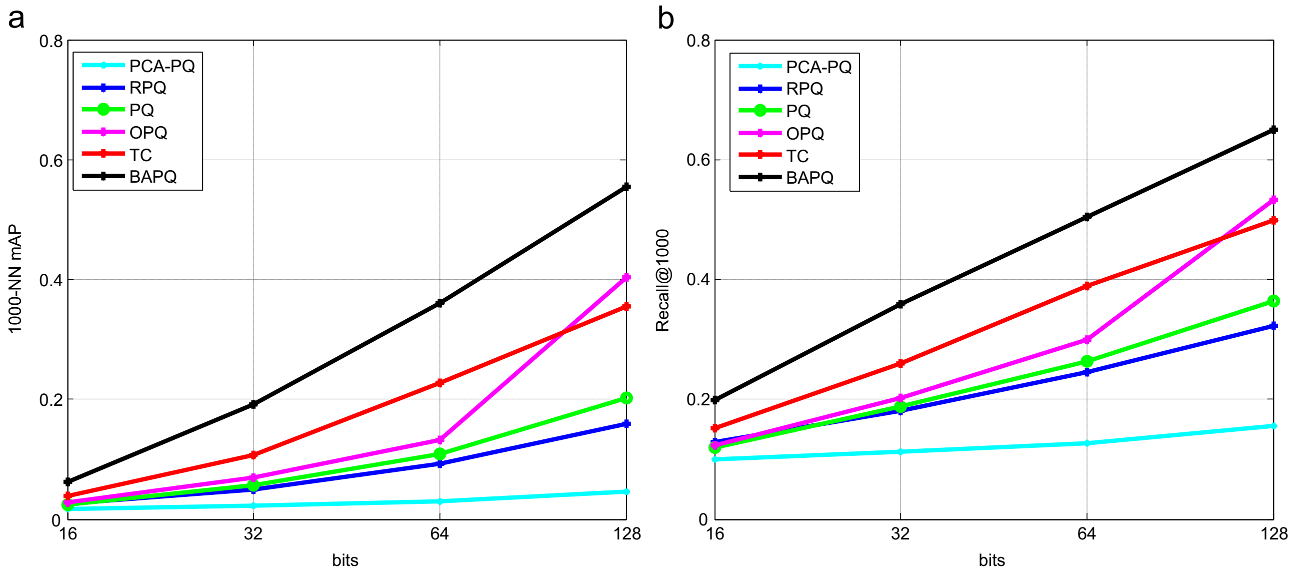


Fig. 2. Comparison of BAPQ to state-of-the-art methods with ADC on GIST1M960 in terms of 1000-NN mAP and recall@1000: (a) 1000-NN mAP on GIST1M960 and (b) recall@1000 on GIST1M960.

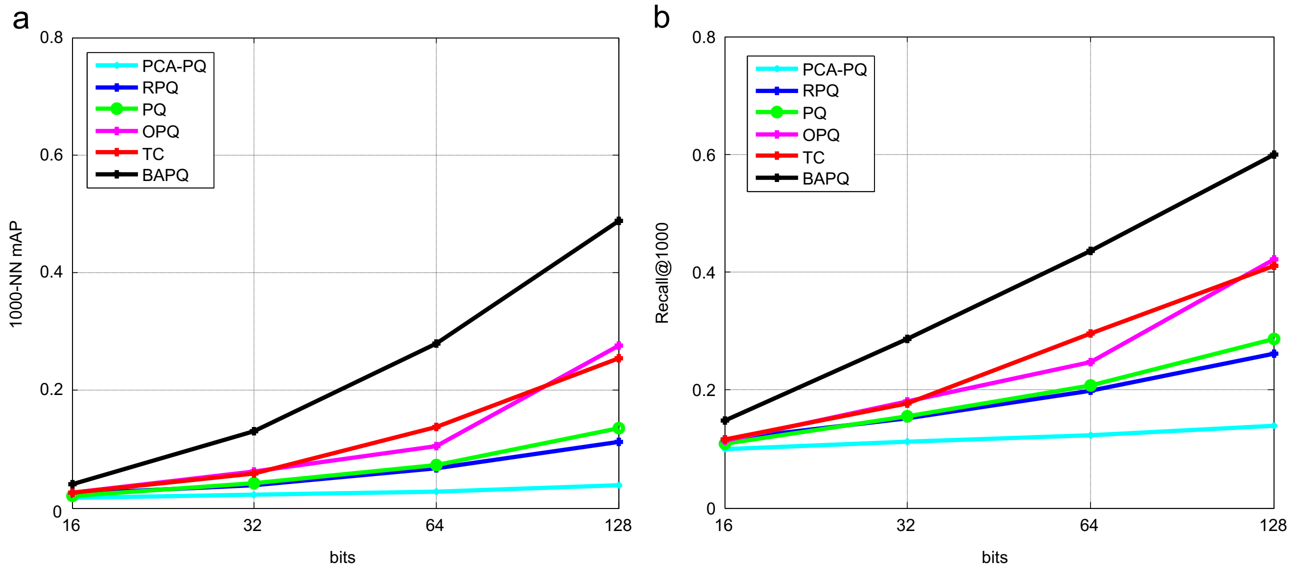


Fig. 3. Comparison of BAPQ to state-of-the-art methods with SDC on GIST1M960 in terms of 1000-NN mAP and recall@1000: (a) 1000-NN mAP on GIST1M960 and (b) recall@1000 on GIST1M960.

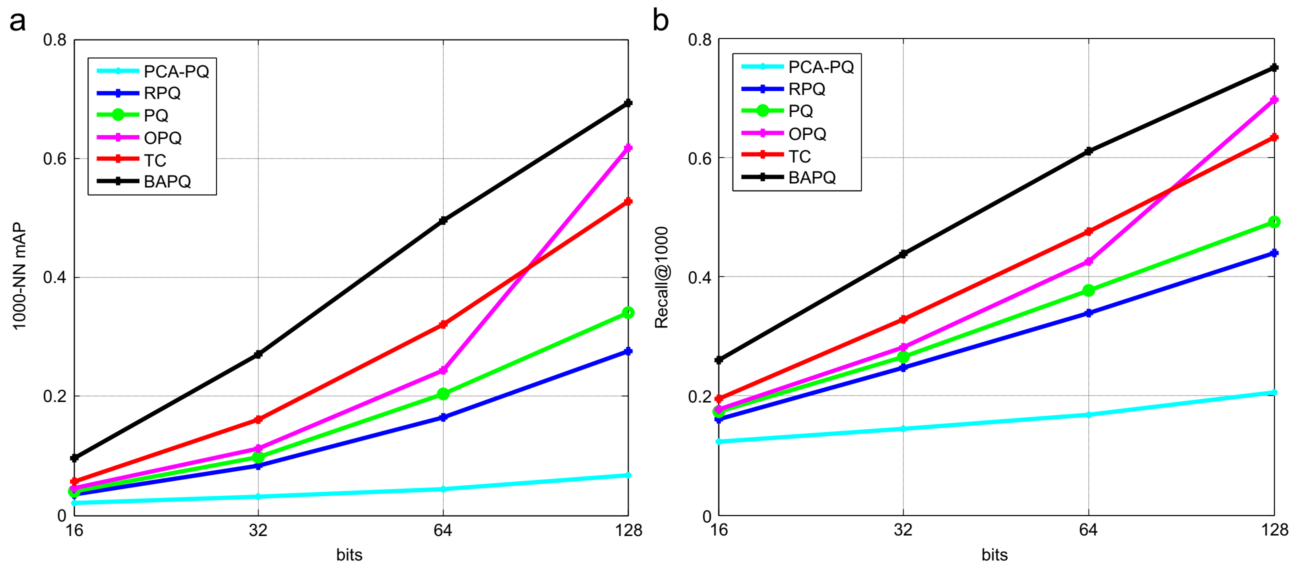


Fig. 4. Comparison of BAPQ to state-of-the-art methods with ADC on GIST1M512 in terms of 1000-NN mAP and recall@1000: (a) 1000-NN mAP on GIST1M512 and (b) recall@1000 on GIST1M512.

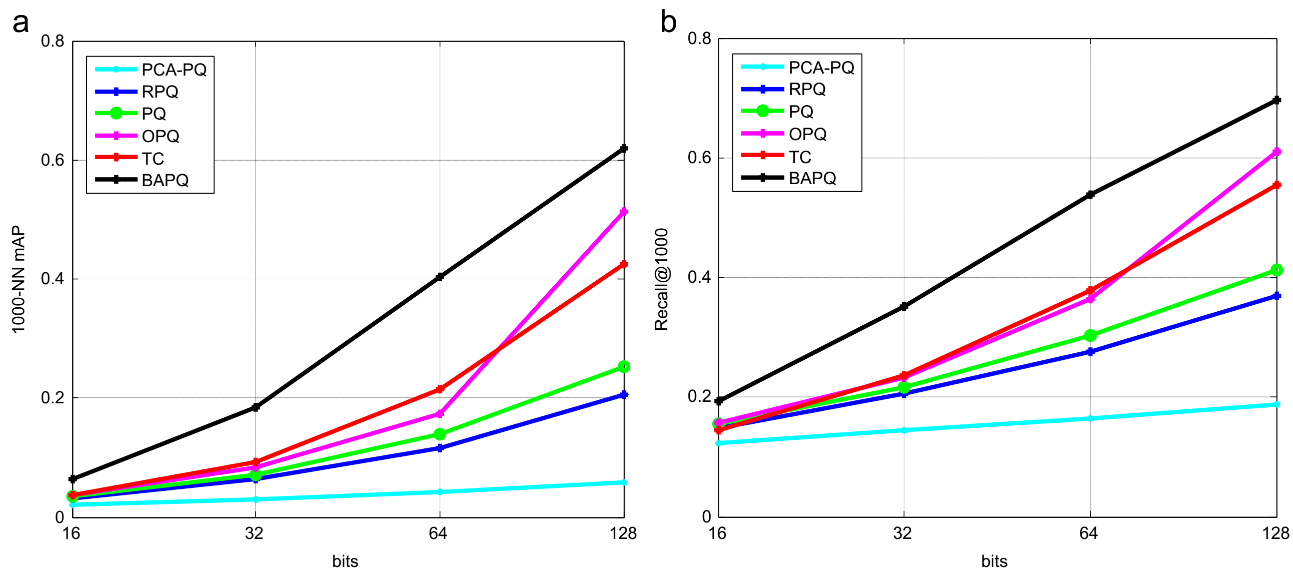


Fig. 5. Comparison of BAPQ to state-of-the-art methods with SDC on GIST1M512 in terms of 1000-NN mAP and recall@1000: (a) 1000-NN mAP on GIST1M512 and (b) recall@1000 on GIST1M512.

the least quantization distortion since our method adaptively allocates different numbers of bits to subspaces to minimize the quantization distortion. TC which allocates bits to each dimension needs less memory than BAPQ which allocates bits to each subspace. However, the search precision of TC is inferior to that of BAPQ due to the larger quantization distortion of TC.

7In BAPQ, Algorithm 1 is designed to find the optimal bit allocation scheme by minimizing the quantization distortion. To evaluate the validity of Algorithm 1, we perform Algorithm 1 on the OPQ-processed data, which is rotated by the projection matrix of OPQ. OPQ_BA is the method where we carry out Algorithm 1 on data processed by OPQ to compute the bit allocation. Note that, in OPQ_BA we use the same subspaces partition as OPQ, namely two subspaces for 16 bits, four subspaces for 32 bits, eight subspaces for 64 bits and sixteen subspaces for 128 bits. The only difference between OPQ and OPQ_BA is the bit allocation scheme. As we can see in Table 5, the quantization distortion of OPQ_BA is further reduced compared to that of OPQ, which indicates the validity of Algorithm 1. But since BAPQ fully uses the unbalance of the data, it achieves lowest quantization distortion among the compared methods. Tables 6 and 7 present the performance comparison on GIST1M960 in terms of mAP and recall respectively. OPQ_BA obtains better results than OPQ due to lower quantization distortion. BAPQ still achieves the best results.

The memory cost comparisons among PQ, OPQ, TC and BAPQ are presented in Table 8. Our BAPQ method costs less memory but achieves better results (see below) than PQ/OPQ. Note that, PCA-PQ and RPQ have the same memory cost as PQ/OPQ.

4.4. Results and analysis

Fig. 2 shows the mAP and Recall on GIST1M960 when using ADC. We return the vectors with the smallest 1000 distances as the results and calculate 1000-NN mAP and recall@1000. Our BAPQ method performs best across the tested bit lengths ranging from 16 bits to 128 bits. Not surprisingly, PCA-PQ always achieves

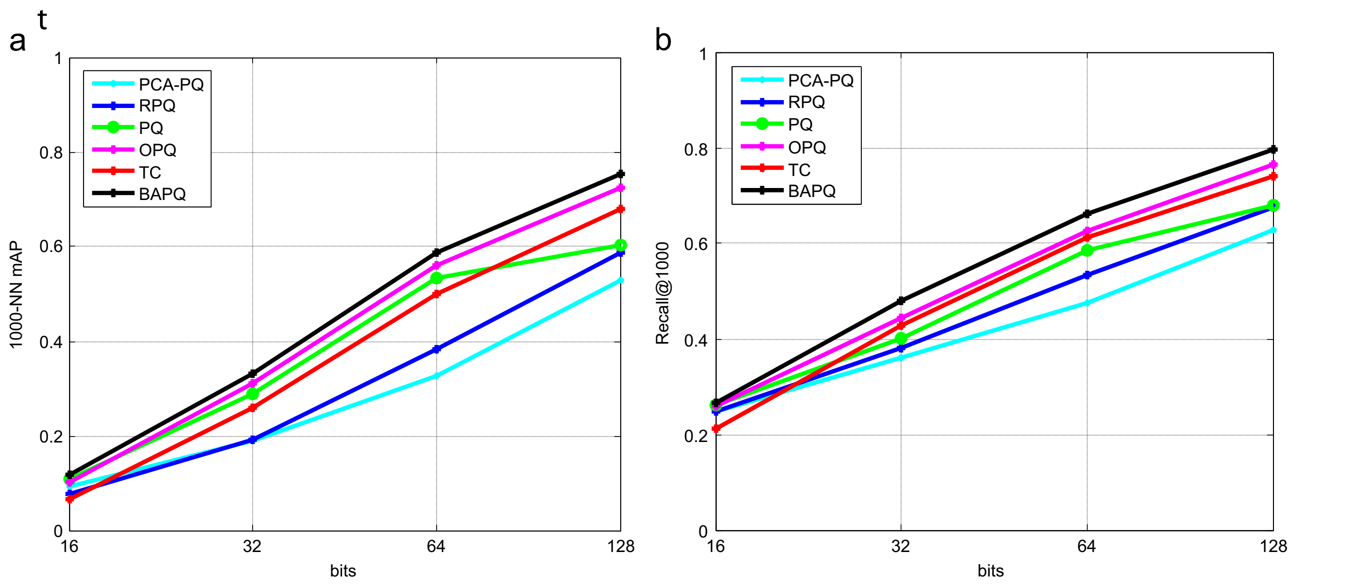


Fig. 6. Comparison of BAPQ to state-of-the-art methods with ADC on SIFT1M in terms of 1000-NN mAP and recall@1000: (a) 1000-NN mAP on SIFT1M and (b) recall@1000 on SIFT1M.

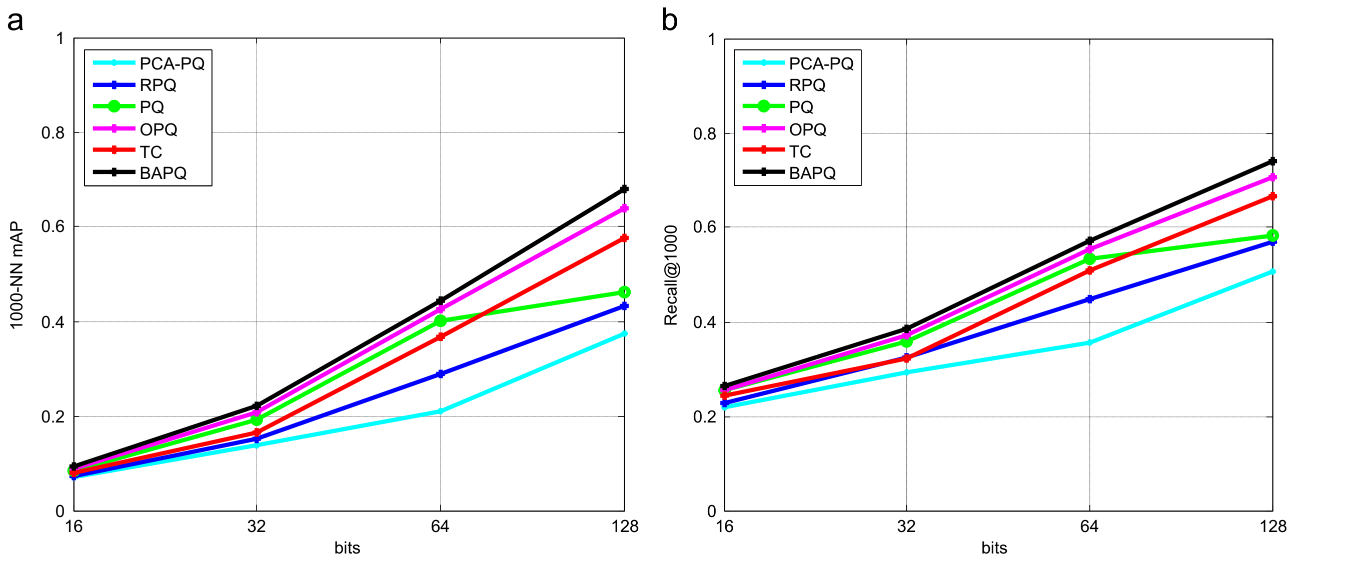


Fig. 7. Comparison of BAPQ to state-of-the-art methods with SDC on SIFT1M in terms of 1000-NN mAP and recall@1000: (a) 1000-NN mAP on SIFT1M and (b) recall@1000 on SIFT1M.

the worst results since the energies of subspaces are very unbalanced. The PQ and RPQ methods get better results than PCA-PQ, which is because that the random projection or randomly ordering dimensions balances the energies of subspaces to some extent. OPQ which aims to minimize the quantization distortion with respect to the subspace decomposition and the quantization codebooks obtains better results than the random methods such as PQ and RPQ. TC which allocates different numbers of bits to dimensions can settle the problem of unbalanced energies of dimensions to some extent and achieves better results than PQ and RPQ. But OPQ and TC are not better than our BAPQ method since BAPQ can adaptively allocate different numbers of bits to subspaces and get lower quantization distortion than OPQ and TC.

Fig. 3 gives the performance of all the compared methods on GIST1M960 using SDC in terms of 1000-NN mAP and recall@1000. BAPQ still achieves the best results for all the bit length cases. Since using ADC can give more precise distance estimation than using SDC, the results by using ADC are better than results by using SDC. From Figs. 2 and 3, we can see that the superiority of our method does not depend on the distance computation choice of SDC or ADC.

Results on GIST1M512 are presented in Fig. 4 for ADC and Fig. 5 for SDC in terms of 1000-NN mAP and recall@1000. Still, PCA-PQ gets the worst results among the compared methods due to its extremely unbalanced energies of subspaces. RPQ and PQ obtain the similar results since they are all random projection methods. (For PQ, the dimensions re-ordering can be represented by an orthonormal matrix.) TC achieves worse results than BAPQ due to its larger quantization distortion. Our BAPQ also achieves the best results. The performance difference among PQ, RPQ, and OPQ indicates that the performance of product quantization largely depends on the energy distribution of subspaces. PQ which uses randomly ordering dimensions performs better than the random projection method RPQ since the randomly ordering dimensions makes the subspaces independent to each other while the random projection process in RPQ destroys the independence [10].

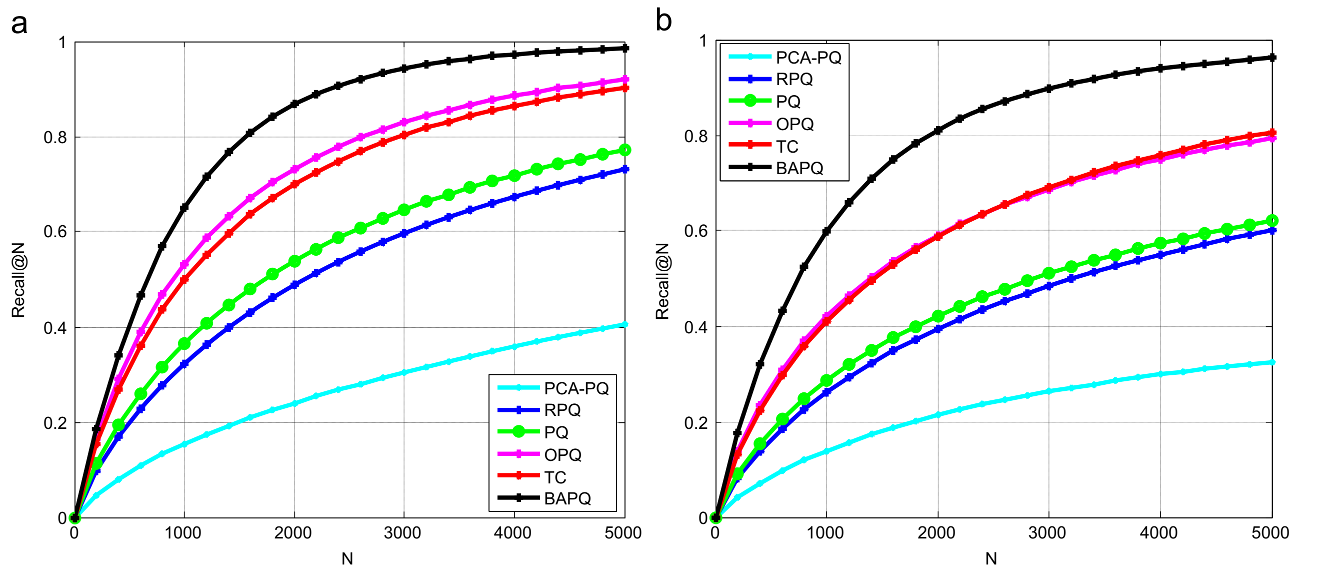


Fig. 8. Comparison of BAPQ to state-of-the-art methods on GIST1M960 in terms of recall at the N top ranked samples with 128 bits: (a) using ADC and (b) using SDC.

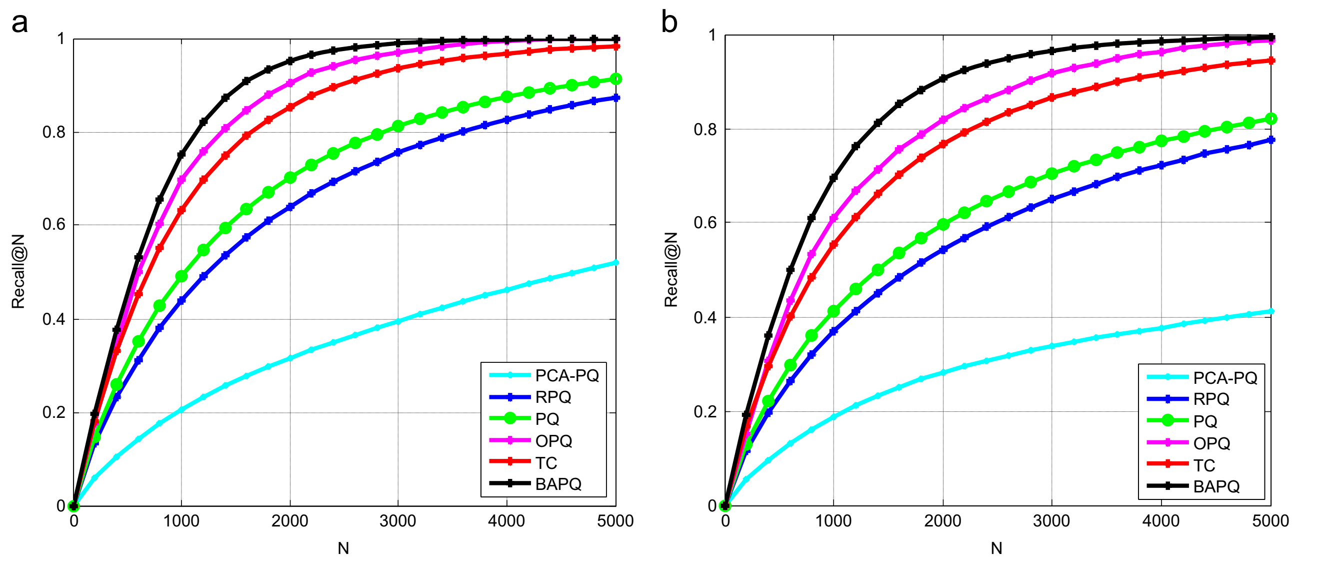


Fig. 9. Comparison of BAPQ to state-of-the-art methods on GIST1M512 in terms of recall at the N top ranked samples with 128 bits: (a) using ADC and (b) using SDC.

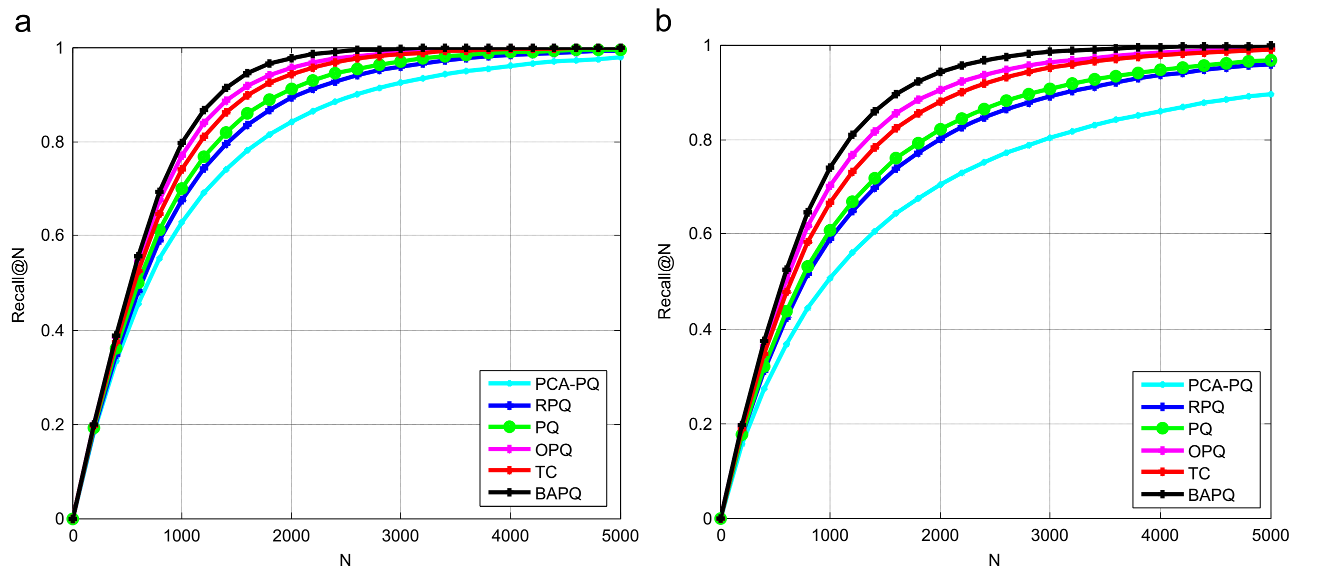


Fig. 10. Comparison of BAPQ to state-of-the-art methods on SIFT1M in terms of recall at the N top ranked samples with 128 bits: (a) using ADC and (b) using SDC.

From Figs. 2 and 4, we can see that the relative improvement of our BAPQ method over PQ and OPQ is greater for higher dimension cases and for more bits cases.

Experimental results on SIFT1M are given in Fig. 6 for ADC and Fig. 7 for SDC. Our BAPQ method still achieves the best results. Compared with the results on GIST1M960 and GIST1M512, the advantage of BAPQ on SIFT1M is not obvious. As we stated above, BAPQ can obtain greater improvement for higher dimension cases. The dimensionality of SIFT1M is 128, which is far less than that of GIST1M960 and GIST1M512. In addition, with the same setup, all the PQ-based methods and TC can obtain better results on lower dimensional datasets than that on higher dimensional datasets.

Figs. 8–10 show the performance on GIST1M960, GIST1M512, and SIFT1M respectively for both ADC and SDC with 128 bits encoding the data. The performance is evaluated through recall vs. *N*, namely, the proportion of the true nearest neighbors ranked in the first *N* positions. We can see that BAPQ always achieves the best results.

For image retrieval, we test our method on image set UKB. Results on UKB can be seen in Fig. 11. Our BAPQ approach outperforms other compared methods from 16 bits to 128 bits. BAPQ and TC achieve better results than other methods, which indicates that adaptively allocating different numbers of bits to subspaces/dimensions may be a better choice for product quantization based methods. Baseline is computed by the exhaustive, linear scan based on the Euclidean distance. One important thing we should know is that the retrieval results not only depend on the ANN search methods, but also are affected by the image representation to a great extent.

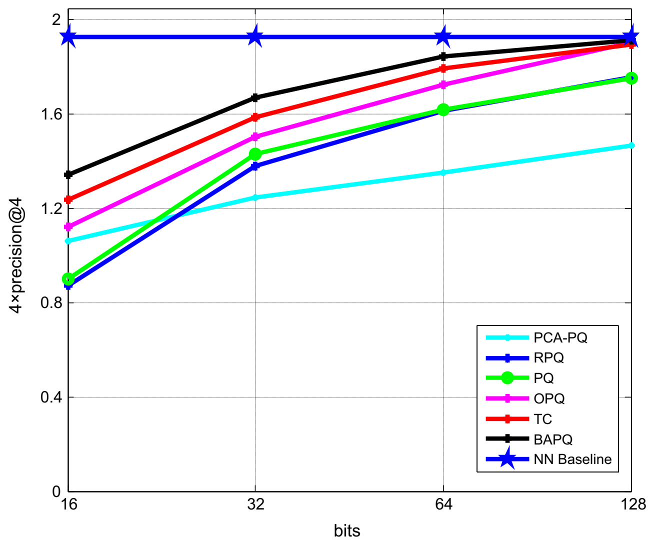


Fig. 11. Comparison of BAPQ to other methods on UKB dataset.

Table 9
Results on PCA projected, randomly projected and raw GIST1M960 in term of 1000-NN mAP for BAPQ using ADC.

	16	32	64	128
PCA	0.0619	0.1909	0.3602	0.5543
Random	0.0019	0.0085	0.0291	0.0800
Raw	0.0011	0.0041	0.0131	0.0294

4.5. Additional analysis

We also evaluate BAPQ on the raw data and randomly projected data. For the raw and randomly projected data, we form subspaces in the way of BAPQ and allocate different numbers of bits to subspaces by using Algorithm 1. The results are presented in Table 9. BAPQ achieves the best results on PCA projected data. The reason is that, when using PCA to rotate the data, the energy will be gathered in the principal components. Thus, subspaces formed by principal components will have more energy and will be allocated with more bits by BAPQ. According to the way of partitioning subspaces and our bit allocation scheme, many subspaces' code length is zero, which can be seen as subspaces selection. As to PCA projected data, we select the subspaces which are formed by principal components and these subspaces have dominant energy. However, for both the raw data and randomly projected data, the subspaces we select are not formed by principal components and many useful data in the unselected subspaces will be discarded.

To check whether the success of BAPQ derives from the redundancy of data, we test BAPQ on GIST1M128. We use PCA to compress GIST1M960 to 128 dimensions and apply a random

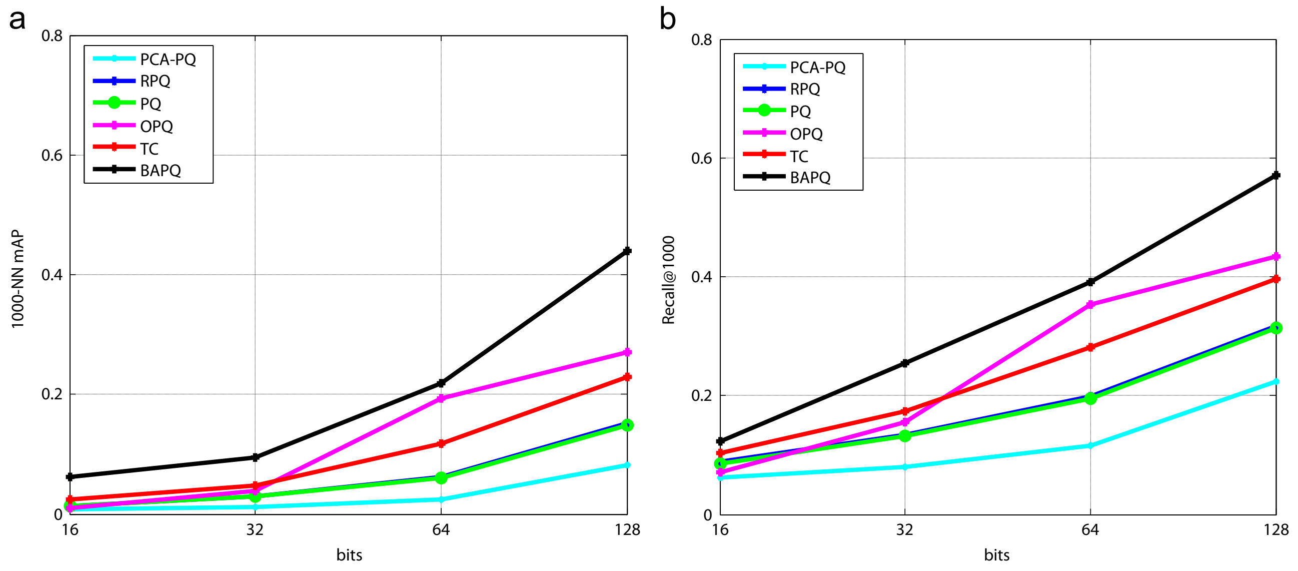


Fig. 12. Comparison of BAPQ to state-of-the-art methods with ADC on GIST1M128 in terms of 1000-NN mAP and recall@1000: (a) 1000-NN mAP on GIST1M128 and (b) recall@1000 on GIST1M128.

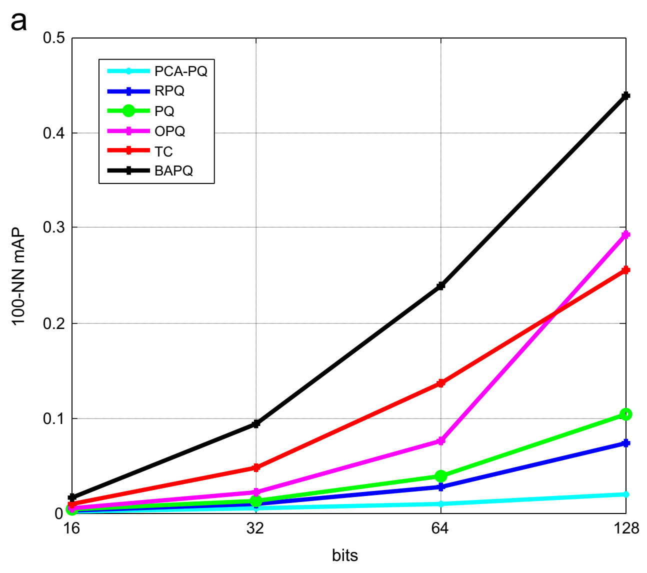


Fig. 13. Comparison of BAPQ to state-of-the-art methods with ADC on GIST1M960 in terms of 100-NN mAP and recall@100: (a) 100-NN mAP on GIST1M960 and (b) recall@100 on GIST1M960.

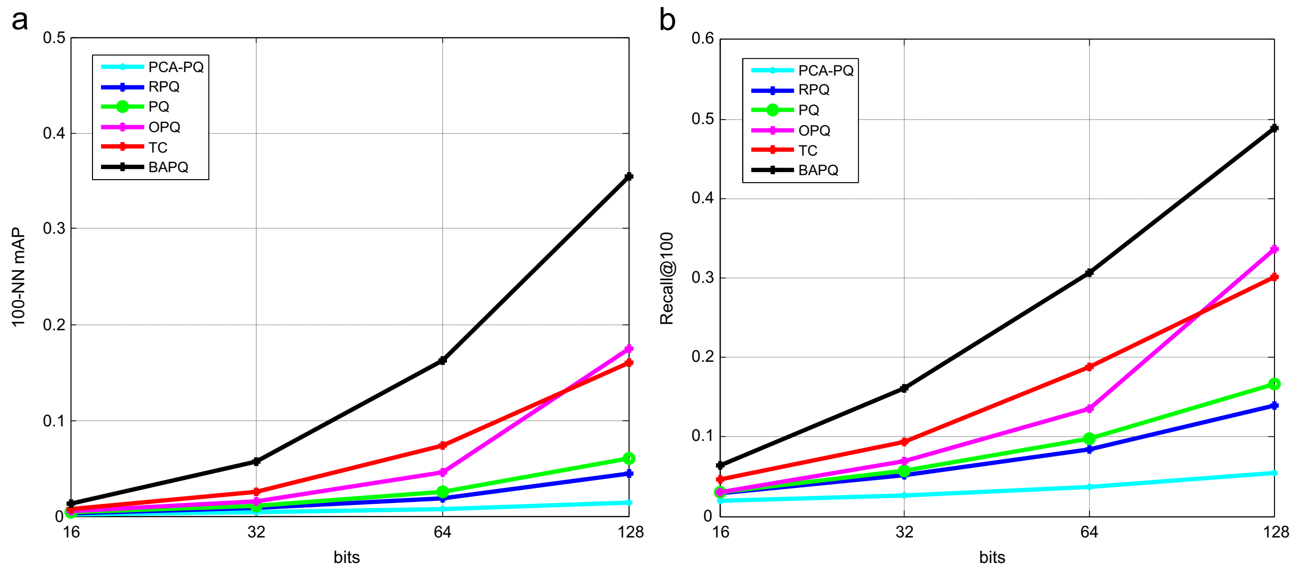


Fig. 14. Comparison of BAPQ to state-of-the-art methods with SDC on GIST1M960 in terms of 100-NN mAP and recall@100: (a) 100-NN mAP on GIST1M960 and (b) recall@100 on GIST1M960.

Table 10
Time consuming of BAPQ at 64 bits encoding.

	Training	Encoding	Query
GIST1M960	54 min 5 s	50.077 s	234 ms
GIST1M512	48 min 14 s	49.234 s	156 ms
SIFT1M	18 min 38 s	10.64 s	47 ms

orthogonal matrix to rotate the PCA-projected data to get GIST1M128. The results on GIST1M128 are presented in Fig. 12. We can see that BAPQ always achieves the best results, which indicates that the superiority of BAPQ derives from the method itself rather than the redundancy of data.

We also evaluate all the compared methods by defining the 100 nearest neighbors as ground truth. The results on GIST1M960 are presented in Figs. 13 and 14. We use both ADC and SDC to compute the distances and 100-NN mAP and recall@100 to measure the performance. Compared with Figs. 2 and 3, as we can see, the comparisons among the methods are almost the same as the comparisons where 1000 nearest neighbors are defined as ground truth, which is also pointed out by OPQ [10].

Table 10 shows the time consuming of BAPQ at 64 bits encoding on single core with our unoptimized $C++$ implementation at the training, encoding and query stage. The encoding time consuming for one million data is less than that of PQ and OPQ because of the lower assignment complexity of BAPQ. The query time of ADC and that of SDC are the same since they have the same complexity. The query time consuming of BAPQ is similar to that of PQ. Besides, as we can see, the training of BAPQ with Algorithm 1 is efficient.

5. Conclusion

In this paper, we propose an adaptive bit allocation product quantization method (BAPQ) for approximate nearest neighbor search. In BAPQ, we firstly project the original data by PCA to find the principal components and group the principal components into several subspaces. In order to minimize the quantization distortion, we adaptively allocate different numbers of codewords (or bits) to subspaces. The essence of BAPQ is the optimal bit allocation scheme. Exhaustively finding the optimal bit allocation scheme is intractable and a greedy algorithm is proposed to get the near-optimal bit allocation scheme. Our method can achieve lower quantization distortion and better performance than state-of-the-art methods. To find the optimal bit allocation scheme of BAPQ, we use a greedy algorithm which may lead to the local optimal or suboptimal bit allocation solution. In the future, we will seek for the algorithm that can give the optimal bit allocation solution with acceptable time consuming.

Acknowledgment

This work has been supported by the National Key Technology R&D Program of China under Grant no. 2012BAH88F02.

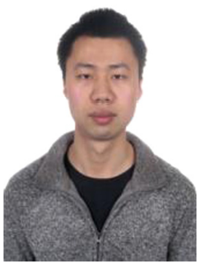
References

[1] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: Proceedings of Conference on Advances in Neural Information Processing Systems, 2008, pp. 1753–1760.
[2] O. Boiman, E. Shechtman, M. Irani, In defense of nearest-neighbor based image classification, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[3] C. Strecha, A. Bronstein, M. Bronstein, P. Fua, Ldahash: improved matching with smaller descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 34 (1) (2012) 66–78.
[4] S. Arya, D. Mount, N. Netanyahu, R. Silverman, A. Wu, An optimal algorithm for approximate nearest neighbor searching fixed dimensions, J. ACM 45 (6) (1998) 891–923.
[5] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: Proceedings of International Conference on very Large Data Base, 1999, pp. 518–529.
[6] J. He, W. Liu, S.-F. Chang, Scalable similarity search with optimized kernel hashing, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 1129–1138.
[7] T. Trzcinski, V. Lepetit, P. Fua, Thick boundaries in binary space and their influence on nearest-neighbor search, Pattern Recognit. Lett. 33 (16) (2012) 2173–2180.
[8] R. Gray, Vector quantization, ASSP Mag. IEEE (1984).
[9] H. Jégou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE Trans. Pattern Anal. Mach. Intell. 33 (1) (2011) 117–127.
[10] T. Ge, K. He, Q. Ke, J. Sun, Optimized product quantization for approximate nearest neighbor search, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2013.
[11] M. J. Huiskes, M. S. Lew, The MIR Flickr retrieval evaluation, in: Proceedings of ACM International Conference on Multimedia Information Retrieval, 2008, pp. 39–43.
[12] J.-P. Heo, Z. Lin, S.-E. Yoon, Distance encoded product quantization, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2014.
[13] H. Jegou, M. Douze, C. Schmid, P. Perez, Aggregating local descriptors into a compact image representation, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2010, pp. 3304–3311.
[14] T. Liu, A. Moore, A. Gay, K. Yang, An investigation of practical approximate nearest neighbor algorithm, in: Proceedings of Conference on Advances in Neural Information Processing Systems, 2004, pp. 32–33.
[15] Y. Gong, S. Lazebnik, Iterative quantization: a procrustean approach to learning binary codes, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2011, pp. 817–824.
[16] A. B. Torralba, R. Fergus, Y. Weiss, Small codes and large image databases for recognition, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[17] Q.-Z. Guo, Z. Zeng, S. Zhang, Adaptive bit allocation hashing for approximate nearest neighbor search, Neurocomputing 151 (2) (2013) 719–728.
[18] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2010, pp. 18–25.
[19] Y. Zhen, D.-Y. Yeung, A probabilistic model for multimodal hash function learning, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 940–948.
[20] P. Li, J. Cheng, H. Lu, Hashing with dual complementary projection learning for fast image retrieval, Neurocomputing 120 (2013) 83–89.
[21] J. Brandt, Transform coding for fast approximate nearest neighbor search in high dimensions, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2010, pp. 1815–1822.
[22] A. Gersho, R. Gray, Vector Quantization and Signal Compression, Kluwer, 1991.
[23] C. Yao, J. Bu, C. Wu, G. Chen, Semi-supervised spectral hashing for fast similarity search, Neurocomputing 101 (2013) 52–58.
[24] W. Liu, J. Wang, Y. Mu, S. Kumar, S.-F. Chang, Compact hyperplane hashing with bilinear functions, in: Proceedings of International Conference on Machine Learning, 2012.
[25] D. Nistér H. Stewénius, Scalable recognition with a vocabulary tree, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2006, pp. 2161–2168.

Qin-Zhen Guo received the B.S. degree in Automation from Hunan University in 2011. He is currently pursuing the Ph.D. degree at the High-tech Innovation Center, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include image retrieval, machine learning, and pattern recognition.

Zhi Zeng received the B.S. and M.S. degree in Computer Science in 2003 and 2006 respectively, both from Chongqing University, China, and the Ph.D. degree in Pattern Recognition from the Institute of Automation, Chinese Academy of Sciences, in 2009. He is currently a Senior Engineer in the Institute of Automation, Chinese Academy of Sciences. His research interests include information retrieval, machine learning, multimedia, and digital rights management.

Shuwu Zhang got his Ph.D. from Chinese Academy of Sciences in 1997. Currently, he is a professor of Institute of Automation, Chinese Academy of Sciences. His research interests are focused on digital content analysis, digital right management, and web-based cultural content service technologies.



Guixuan Zhang received the B.S. degree in measurement and control technology from University of Science and Technology, Beijing, China, in 2012. He is currently persuing the Ph.D. degree at the High-tech Innovation Center, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include image retrieval and computer vision.

Yuan Zhang received the B.S. degree in Automation from Beijing University of Posts and Telecommunications in 2009 and the Ph.D. degree in Pattern Recognition from the Institute of Automation, Chinese Academy of Sciences in 2014. He is currently an Algorithm Engineer in Samsung Electronics. His research interests include image retrieval, object detection, machine learning, and pattern recognition.