Neurocomputing ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Neurocomputing

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

Adaptive multi-bit quantization for hashing

Cheng Deng [a], Huiru Deng [a], Xianglong Liu [b,*], Yuan Yuan [c]

[a] Department of Electronic and Engineering, Xidian University, Xi'an 710071, China
[b] State Key Lab of SDE, Beihang University, Beijing 100191, China
[c] Center for OPTical IMagery Analysis and Learning (OPTIMAL), Xian Institute of Optics and Precision Mechanics, CAS, Xian 710119, China

ARTICLE INFO

Article history:
Received 27 June 2014
Received in revised form
15 September 2014
Accepted 17 September 2014
Communicated by L. Shao

Keywords:
Image retrieval
Multi-bit quantization
Bit allocation
Incomplete coding
K-means clustering

ABSTRACT

Recently, hashing methods which try to solve similarity-preserving approximate nearest search problem have obtained widely applications in various fields such as content-based image retrieval, object recognition and pose estimation. However, how to learn effective hash codes to describe the similarities in the large-scale database still remains as a NP-hard problem. Besides, a fatal problem lying in the existing hashing methods is that they usually threshold the real values to binary codes using single-bit quantization (SBQ) at the highest point density, which may destroy the data structure seriously. Due to this problem, double-bit quantization (DBQ) is proposed to solve the problem of SBQ by adaptively learning thresholds to quantize the real values to two bits, and achieves impressive results. However, one problem in DBQ is that it neglects the amount of the information contained in different data dimensions. In this paper, we propose a multi-bit quantization method based on bit allocation to quantize each projected dimension with variable bit numbers. Besides, different from existing methods of choosing threshold, we propose an incomplete coding manner by clustering to generate binary codes. Experiments on two large datasets demonstrate the feasibility of our method.

1. Introduction

Approximate nearest neighbor (ANN) search in massive data has been widely used in many related areas, such as content-based retrieval [1–5], object recognition [6–10], action recognition [11,12], and human pose estimation [13]. A popular approach is to learn similarity-preserving hash functions, where similar data points in the original space are mapped into close binary hash codes in Hamming space. By using these binary hash codes, ANN search can be carried out in constant or sub-linear time. Moreover, the cost of data storage and transmission will be greatly reduced. This means that hashing methods can provide an effective and efficient way to perform ANN search on large-scale datasets. Therefore, considerable efforts have been spent on designing hashing methods from different perspectives. Loosely speaking, the existing hashing methods can be divided into two categories: data-independent and data-dependent.

Data-independent methods usually adopt random projections as hash functions without using any training data. One of the representative data-independent methods is Locality Sensitive Hashing (LSH) [14]. As a kernelized version of LSH, Kernelized LSH (KLSH) [15] captures the intrinsic relationships between data samples using kernels instead of linear inner products. Another representative method is Shift Invariant Kernel Hashing (SIKH) [16] which is a distribution-free method based on random features mapping for shift-invariant kernels, and the expected Hamming distance between the binary codes is approximated by the distance between data points in a kernel space. Data-independent methods make no prior assumption about the data distribution, and come with theoretical guarantees that the original distance or similarity is asymptotically preserved in Hamming space with increasing code lengths, hence they require relatively long codes to achieve good accuracy. As a result, data-independent methods are often less effective than data-dependent methods in practice.

Data-dependent methods learn hash functions from training data. Representative data-dependent methods include semantic hashing [17], spectral hashing (SH) [18], self-taught hashing (STH) [19], minimal loss hashing (MLH) [20], binary reconstruction embedding (BRE) [21], anchor graph hashing (AGH) [22], iterative quantization (ITQ) [23], collaborative hashing [24], and so on. Semantic hashing adopts a deep generative model based on the stacked restricted Boltzmann machine to learn hash functions. SH uses spectral graph partitioning for learning hash functions. Inspired by SH, STH learns hash functions via SVM for the unseen data points. MLH formulates similarity-preserving hash functions learning under a latent structure prediction framework. BRE learns hash functions by minimizing the reconstruction error between

* Corresponding author.
E-mail address: xlliu@nlsde.buaa.edu.cn (X. Liu).

http://dx.doi.org/10.1016/j.neucom.2014.09.033
0925-2312/

Euclidean distance in the original space and the Hamming distance of the corresponding binary codes. AGH utilizes anchor graphs to estimate graph Laplacian eigenvectors, and the hash functions are learned by thresholding the lower eigenfunctions of the anchor graph Laplacian. ITQ learns an orthogonal rotation matrix to refine the initial PCA-projection matrix to minimize the quantization error of mapping the data from original space to the vertices of binary hypercube. Collaborative hashing learns the compact hash codes and the hash functions for entities in two different views, in which both the inner similarities in intra-view and the interrelationships across views are preserved simultaneously.

Both data-independent and data-dependent hashing methods typically adopt a two-stage strategy to generate binary codes. In the first stage, several projected dimensions of real values are generated, while in the second stage the real values are quantized into binary codes by thresholding. However, most existing methods, such as those mentioned above, always overlook the important fact that the information of different projected dimensions is different due to disparities in the data distribution. Therefore, it is undesirable to use the same number of bits for different projected dimensions. To overcome this problem, some methods such as ITQ [25] and Isotropic hashing attempt to balance the variances of different projected dimensions. Subsequently, the same number of bits are utilized to quantize each balanced projected dimension.

Fig. 1 gives a brief demonstration of the common used three quantization strategies, single-bit quantization (SBQ), hierarchical hashing (HH) and double-bit quantization (DBQ). We can observed that the threshold in the SBQ lies in the region of highest point density, and consequently many neighboring points close to the threshold might be hashed to different bits, completely opposite to the principle of hashing. HH improves the effects of single-bit quantization while for any projected dimension, the Hamming distance between the two farthest points is the same as that between two relatively close points, which is far from ideal. Thus, DBQ [26] has been developed, in which each projected dimension is quantized into two bits by learning adaptive thresholds.

However, we have observed that although DBQ shows promising performance and is adaptable when combined with other projection functions. It still neglects two significant issues:

- Different projected dimensions contain different amounts of information. It is therefore intuitive that the projected dimensions with large variances should be allocated more bits (i.e., more than two), and the ones with small variances should be assigned either one bit or even none. It is more reasonable than to equalize the variances of projected dimensions.
- Theoretically, binary encoding should preserve the neighborhood structure in the original input space. This means that the points belonging to the same cluster should be encoded by the same binary codes and satisfy definite Hamming distance constraints, which is more reasonable and feasible than finding thresholds, as in SBQ and DBQ.

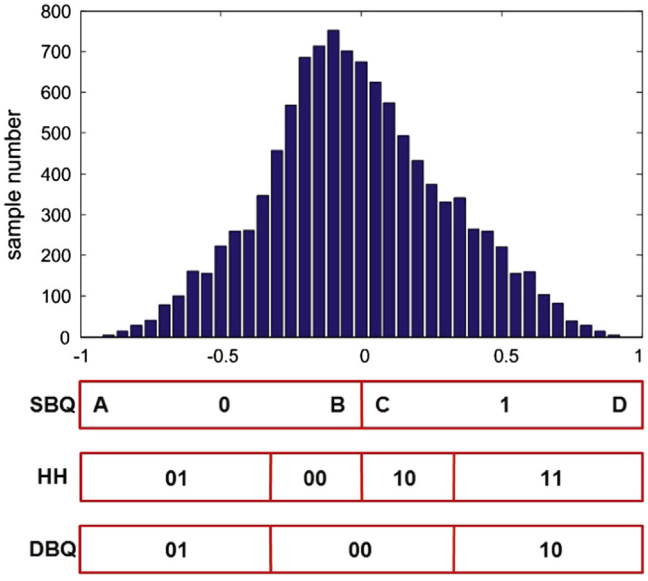


Fig. 1. Mean distribution of the real values computed by PCA on *CIFAR-10* dataset.

In this paper, we propose a novel quantization strategy, dubbed adaptive multi-bit quantization (AMBQ), to quantize each projected dimension. Experiments on real datasets demonstrate that our AMBQ method outperforms other state-of-the-art quantization methods including SBQ, HH, and DBQ.

The paper is organized as follows. Section 2 presents the notation and overview. In Section 3, we detail the description and the formulation of the algorithm and its optimization technique. Our experiments and analysis are depicted in Section 4. Finally, Section 5 concludes our paper.

2. Problem definition

Given a dataset consisting of n data points $\mathcal{X}=\{\mathbf{x}_i\}_{i=1}^{n}$ with $\mathbf{x}_i \in \mathbb{R}^d$, the hashing paradigm is to first use a set of linear or nonlinear hash functions $F=\{f_k : \mathbb{R}^d \to \mathbb{R}\}_{k=1}^{K}$ to map $\mathbf{x}$ to $F(\mathbf{x})$, and then binarize $F(\mathbf{x})=(f_1(\mathbf{x}),\ldots,f_K(\mathbf{x}))^{\top}$ by comparing each $f_k(\mathbf{x})$ with a predefined threshold θ to obtain a K-bit binary code $H(\mathbf{x}) \in \{0,1\}^K$. In most cases, each projected dimension is only quantized into a single bit with the threshold 0 when the data are normalized to have zero mean.

3. The proposed algorithm

In this section we detail our proposed AMBQ. The overall formulation is first described. Then, bit allocation and incomplete binary encoding are designed to determine the encoding rules. Finally, we propose a novel quantization method based on k-means clustering to generate the binary codes.

3.1. Formulation

Our basic model quantizes different projected dimensions according to their corresponding variances. More specifically, given a set of projected functions $F(\mathbf{x})$, we first allocate a certain number of bits to each projected dimension according to the information amount contained in this dimension, which is called the *bit allocation*.

After bit allocation we encode the real values in the current projected dimension with binary codes. Instead of complete encoding, here we use *incomplete binary encoding* in order to preserve the neighborhood structure of 1-dimensional (1D) data. Then, following classical vector quantization (VQ) theory [27], any real values generated by projected functions can be assigned to its nearest cluster center and minimize quantization error. In this way, *k-mean-based quantization* is adopted to binarize these real values, where the distance between any two real values $\mathbf{x}_1$ and $\mathbf{x}_2$ is approximated by the distance of their cluster centers [28]

$$d(\mathbf{x}_1,\mathbf{x}_2)=d(c_i(\mathbf{x}_1),\quad c_j(\mathbf{x}_2)) \approx d_h(i(\mathbf{x}_1),\quad j(\mathbf{x}_2)), \tag{1}$$

where $i(\mathbf{x})$ denotes the index of the cluster centers, $c_i(\mathbf{x})$ is the cluster center containing $\mathbf{x}$, $d(\cdot)$ is the Euclidean distance, and $d_h(\cdot)$ is the Hamming distance. The Hamming distance between any two

Please cite this article as: C. Deng, et al., Adaptive multi-bit quantization for hashing, Neurocomputing (2014), http://dx.doi.org/10.1016/j.neucom.2014.09.033

indices i and j can be further denoted as

$$d_h(i,j) \triangleq s \cdot h^{1/2}(i,j). \tag{2}$$

Here $h^{1/2}$ is the square root of the Hamming distance, and s is a constant scale to preserve the neighborhood structure between data points.

3.2. Bit allocation

As mentioned above, the variances of the different projected dimensions in AMBQ are unequal. Therefore, we highlight the effect caused by anisotropic variances and use a variable number of bits to encode these different projected dimensions, which are defined as

$$b_i = \frac{Var\_cur(i)}{Var\_all} \times b, \quad i \in \{1, \ldots K\}, \tag{3}$$

where $Var\_cur(i)$ is the variance in ith projected dimension, measuring the information contained in this dimension, Var_all is the total variances of all projected dimensions, b_i is the number of bits allocated to ith dimension, and b is the total code length for all projected dimensions.

Initially, we can determine the number of bits for each projected dimension according to its proportion of total variances, as shown in Eq. (3). However, this intuitive method for bit allocation requires a rounding operation during calculation, which may cause the real code length b to deviate from the expected value. We therefore transform this problem into a mixed-integer quadratic programming problem, and then solve it using an optimization algorithm to obtain a more reasonable bit allocation for each projected dimension. The objective function is defined as

$$\min_{c_i} \quad \sum_{i=1}^{K} (v_i - \sigma c_i)^2,$$

$$\text{s.t.} \quad b = \sum_{i=1}^{K} b_i, \quad c_i \in N, \quad b_i \in N \tag{4}$$

Here, v_i is the variance of ith projected dimension, $\sigma = Var\_all/c$ is the averaged variance for each cluster (where c is the total number of clusters), and c_i is the number of clusters in the ith projected dimension.

3.3. Incomplete binary encoding

Once the number of bits b_i for a given projected dimension is determined, the number of clusters for this dimension can then be calculated. As a complete encoding method, k-means hashing (KMH) [28] directly generates 2^{b_i} cluster centers. In practice, however, if the value of b_i is large, then the calculation of clusters will be very slow or even provide no solution. Moreover, complete encoding methods such as KMH are incapable of retaining the neighborhood structure of the 1D data distribution.

We therefore develop an incomplete binary encoding method that is more suitable for the 1D data distribution. As shown in Fig. 2, the encoding process dictates that the Hamming distance between two adjacent points is 1 and the Hamming distance between any two non-adjacent points is increased by 1, where the greatest distance must be less than the total number of bits b_i that the current projected dimension requires. Hence, the number of clusters of each projected dimension c_i and the number of encoding bits b_i that the dimension requires satisfy the following relationship:

$$c_i = b_i + 1, \quad i \in \{1, \ldots, K\} \tag{5}$$

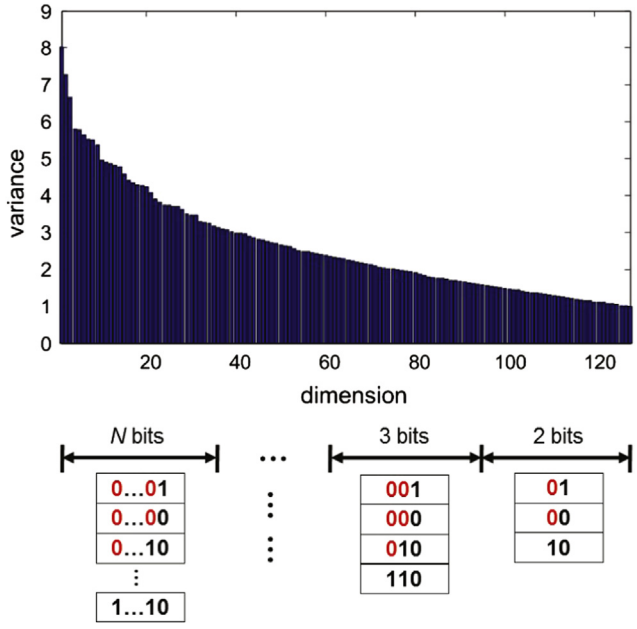


Fig. 2. Incomplete binary encoding based on variance distribution of the real values which is computed by PCA on *CIFAR-10* dataset.

3.4. K-means-based quantization

To quantize each data point, they first need to cluster the real-valued vectors for a given projected dimension. Each point is then assigned to their nearest clustering center coded by incomplete binary encoding. Inspired by KMH, we attempt to simultaneously minimize the average quantization error and the affinity error. This means that the distance between the points belonging to the same class and their cluster center should be as small as possible

$$E_{quan} = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{S}} \| \mathbf{x} - c_{i(\mathbf{x})} \|^2, \tag{6}$$

where $\mathcal{S}$ is the dataset with n points. In classical k-means clustering, this objective function can be solved by an Expectation–Maximization (EM)-like algorithm to alternatively optimize $i(\mathbf{x})$ and $\{c_i\}$.

The affinity error (the Hamming distance of the close cluster centers in the original space) after encoding should be small. To make this affinity error minimization problem tractable, we adopt the weighted difference between two c-by-c affinity matrices $d(\cdot,\cdot)$ and $d_h(\cdot,\cdot)$ to measure the affinity error

$$E_{aff} = \sum_{i=0}^{c-1} \sum_{j=0}^{c-1} w_{ij} (d(c_i, c_j) - d_h(i,j))^2. \tag{7}$$

Combining the quantization error and affinity error we can get the following objective function:

$$E = E_{quan} + \lambda E_{aff}, \tag{8}$$

where λ is a constant and is empirically set to 10 in this paper. To obtain the optimum, we minimize Eq. (8) in an simple alternating fashion:

- *Fix* $\{c_i\}$ *and update* $i(\mathbf{x})$. Each point $\mathbf{x}$ is assigned to its nearest codeword in the codebook $\{c_i\}$.
- *Fix* $i(\mathbf{x})$ *and update* $\{c_i\}$. The update of any codeword depends on all the others due to the pairwise affinity $d(c_i, c_j)$ in Eq. (7). Sequentially optimize each individual codeword c_j with other

fixed as $\{c_i\}_{i\neq j}$ as follows:

$$c_j = \arg\min_{c_j} \left(\frac{1}{n_{\mathbf{x};i(\mathbf{x})=j}} \sum \|\mathbf{x}-c_j\|^2 + 2\lambda \sum_{i;i\neq j} w_{ij}(d(c_i,c_j)-d_h(i,j)) \right)^2 \quad (9)$$

This problem can be solved using a quasi-Newton method [29]. The overall optimization procedure is summarized in Algorithm 1.

3.5. Out of sample extension

For a given test set $\mathcal{Y}=\{\mathbf{y}_1,\ldots,\mathbf{y}_n\}\in\mathbb{R}^{n\times d}$ and ordered centers set $\mathcal{C}=\{c_t|t=1,\ldots,b_t+1\},(t\in 1,\ldots,K)$, we can assign the data point's index by its nearest center's index.

Algorithm 1. The algorithm to learn adaptive multi-bit quantization for hashing.

1: **Input**: the training set $\{\mathbf{x}_i\}_{i=1}^n$, the total number of bits b for all projected dimensions;
2: **Initialize**: the cluster center set C and the scale parameter s in Eq. (2) of each projected dimension;
3: **for** $j=1$ to K **do**
4: For all $\mathbf{x}_i\in\mathcal{X}$, assign the index of $\mathbf{x}_i$ by its nearest center's index;
5: For each dimension, update the centers $c_j(j=1,\ldots,b_j+1)$, using Eq. (9);
6: **end for**
7: **Output**: a set of ordered cluster center set $C=\{c_j|j=1,\ldots,b_j+1\}, j\in\{1,\ldots,K\}$.

4. Experiments

In this section we evaluate our proposed AMBQ on two large-scale image retrieval benchmarks: CIFAR-10 [30] and LabelMe [31].

CIFAR-10 dataset consists of 60,000 color images. These images are manually labeled into 10 classes with 6000 image per class, which are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The size of each image is $32\times 32\times 3$ pixels. We represent them with 512-dimensional GIST descriptor [32].

22 K-LabelMe dataset contains 22,019 images sampled from the large LabelMe dataset. The images are scaled to 32×32 pixels, and then represented by 512-dimensional vector of GIST features.

4.1. Evaluation protocols

In the experiments, we randomly selected 1000 points as queries and used the remaining points as a training set to learn hash functions. All the experimental results were averaged over 10 random train/test partitions.

As the methodology most widely used in hashing methods, Euclidean neighbors in the original space were considered the ground truth. For each query, the ground truth neighbors were defined as all the points within the radius determined by the average distance to the 50-th nearest neighbor in the dataset. Based on the ground truth, we computed the precision-recall curve and the mean average precision (mAP) to evaluate performance.

The mAP for a set of queries is the mean of the average precision scores for each query, which is defined as follows:

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}, \quad (10)$$

where

$$AveP = \frac{1}{11} \sum_{r\in\{0,0.1,\ldots,1.0\}} p_{\text{interp}(r)}. \quad (11)$$

Table 1
mAP on 22 *K-LabelMe* dataset. The best mAP among SBQ, DBQ and AMBQ under the same setting for different hashing methods is shown in bold face.

No. of bits	32			64		
	SBQ	DBQ	AMBQ	SBQ	DBQ	AMBQ
LSH	**0.1569**	0.1130	0.1075	**0.2581**	0.2396	0.2429
ITQ	0.2792	**0.2837**	0.2616	0.3311	0.3852	**0.3928**
PCA	0.0551	**0.1420**	0.1309	0.0457	0.1635	**0.1682**
KLSH	**0.1384**	0.0821	0.0904	**0.1566**	0.1194	0.1339
No. of bits	128			256		
	SBQ	DBQ	AMBQ	SBQ	DBQ	AMBQ
LSH	0.3457	0.3934	**0.4060**	0.4104	0.5102	**0.5498**
ITQ	0.3590	0.4636	**0.5056**	0.3811	0.5086	**0.5887**
PCA	0.0368	0.1618	**0.1784**	0.0292	0.1394	**0.1664**
KLSH	**0.2391**	0.1908	0.1964	**0.3579**	0.2830	0.3050

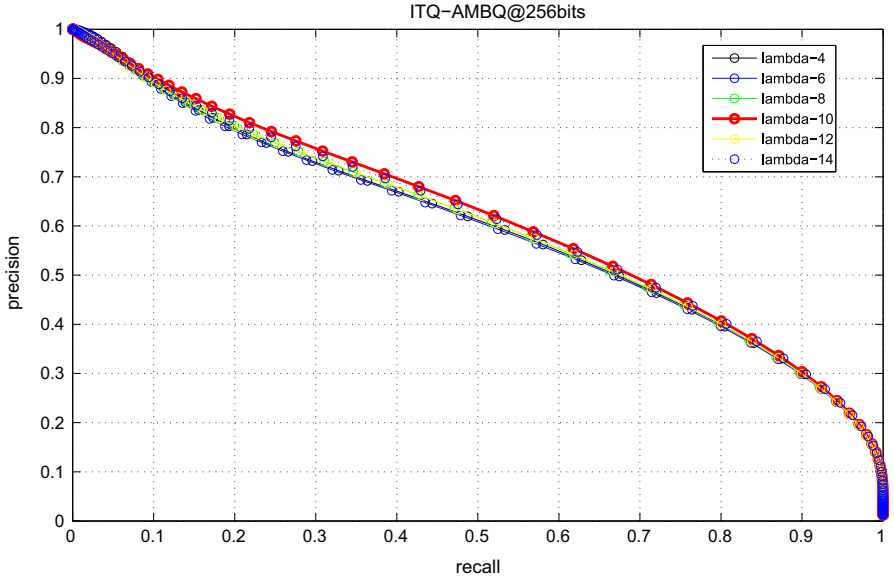


Fig. 3. The effects of the λ on the average performance. We change the value from 4 to 14, and experimentally find that the performance gains the best when λ is 10.

where $p_{\text{interp}(r)}$ is an interpolated precision that takes the maximum precision over all recalls greater that r.

Table 2
mAP on *CIFAR-10* dataset. The best mAP among SBQ, DBQ and AMBQ under the same setting for different hashing methods is shown in bold face.

No. of bits	32			64		
	SBQ	DBQ	AMBQ	SBQ	DBQ	AMBQ
LSH	**0.1116**	0.0688	0.0998	0.1819	0.1589	**0.1851**
ITQ	**0.2524**	0.2241	0.2068	0.3120	0.3346	**0.3361**
PCA	0.0420	**0.1208**	0.1146	0.0357	0.1651	**0.1878**
KLSH	**0.1327**	0.0717	0.0931	**0.1861**	0.0959	0.1308
No. of bits	128			256		
	SBQ	DBQ	AMBQ	SBQ	DBQ	AMBQ
LSH	0.2767	0.3134	**0.3284**	0.3451	0.4637	**0.5074**
ITQ	0.3478	0.4313	**0.4818**	0.3692	0.5237	**0.6235**
PCA	0.0228	0.2004	**0.2207**	0.0223	0.2032	**0.2397**
KLSH	**0.2884**	0.1407	0.1650	**0.3761**	0.2476	0.2665

AMBQ can replace the quantization stage in any existing hashing method to get a new version. In these comparative experiments, we selected only the most representative methods for evaluation, namely LSH [33], KLSH [15], PCA [23], and ITQ [23]. LSH and KLSH are data-independent hashing methods, while PCA and ITQ are data-dependent methods. By exploiting different quantization methods we can obtain different versions of a specific hashing method and utilize them as baselines. For example, "LSH-SBQ" is the original LSH method with single-bit quantization, "LSH-HH" denotes the original LSH method combined with hierarchical hashing, "LSH-DBQ" combines LSH projection with double-bit quantization, and "LSH-AMBQ" is a combination of LSH with our adaptive multi-bit quantization. For all the methods we use the source code provided by the original authors. Since the dynamic range of the variances in PCA hashing is very large, we limited the length of the binary code to no longer than 6 bits.

4.2. Parameter sensitivity

We simply give a brief demonstration about the selection of parameter λ in Eq. (9). According to the work of Kong et al. [28], the

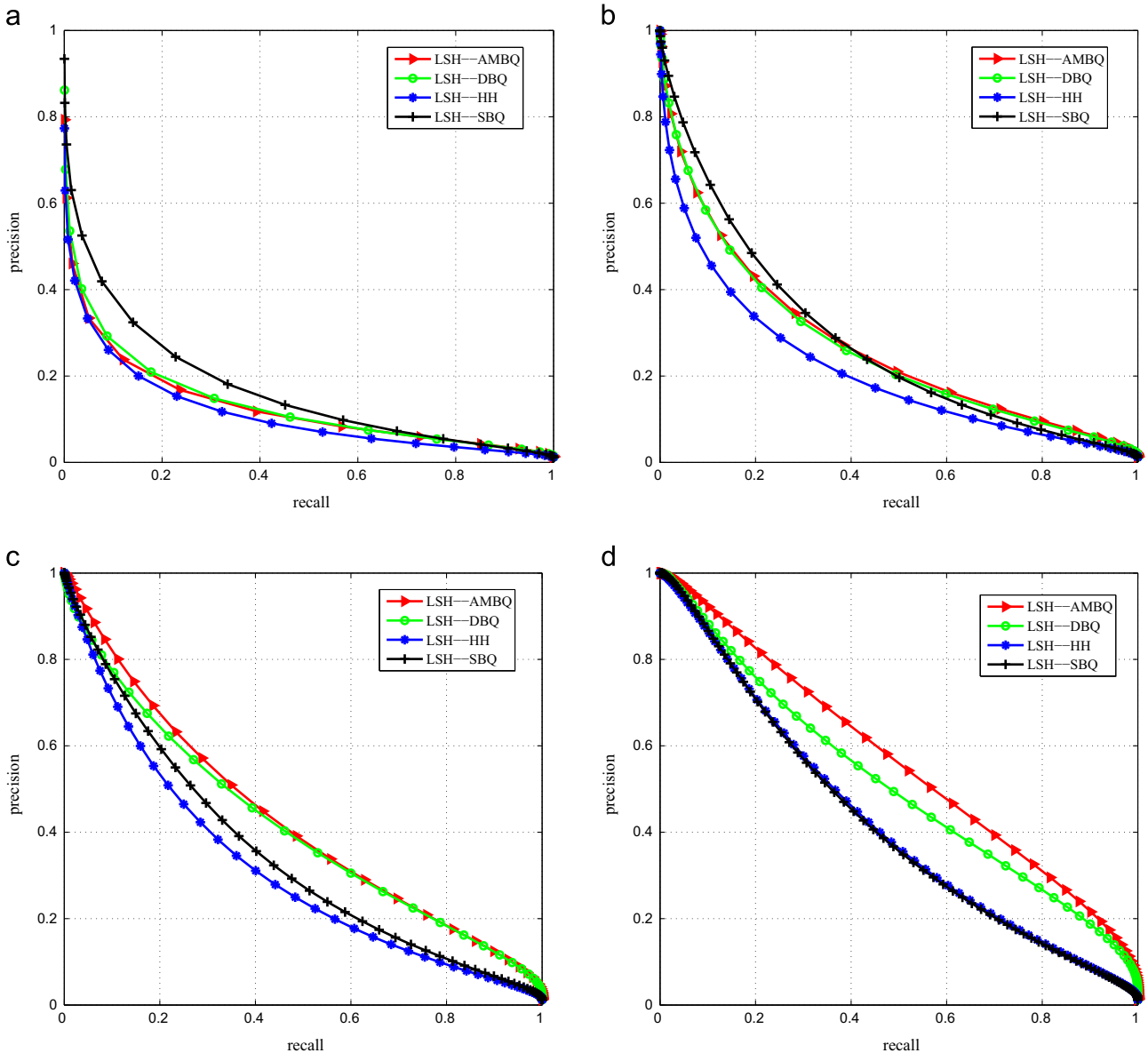


Fig. 4. Precision-recall curve for LSH on 22*K-LabelMe* dataset. (a) LSH 32 bits. (b) LSH 64 bits. (c) LSH 128 bits. (d) LSH 256 bits.

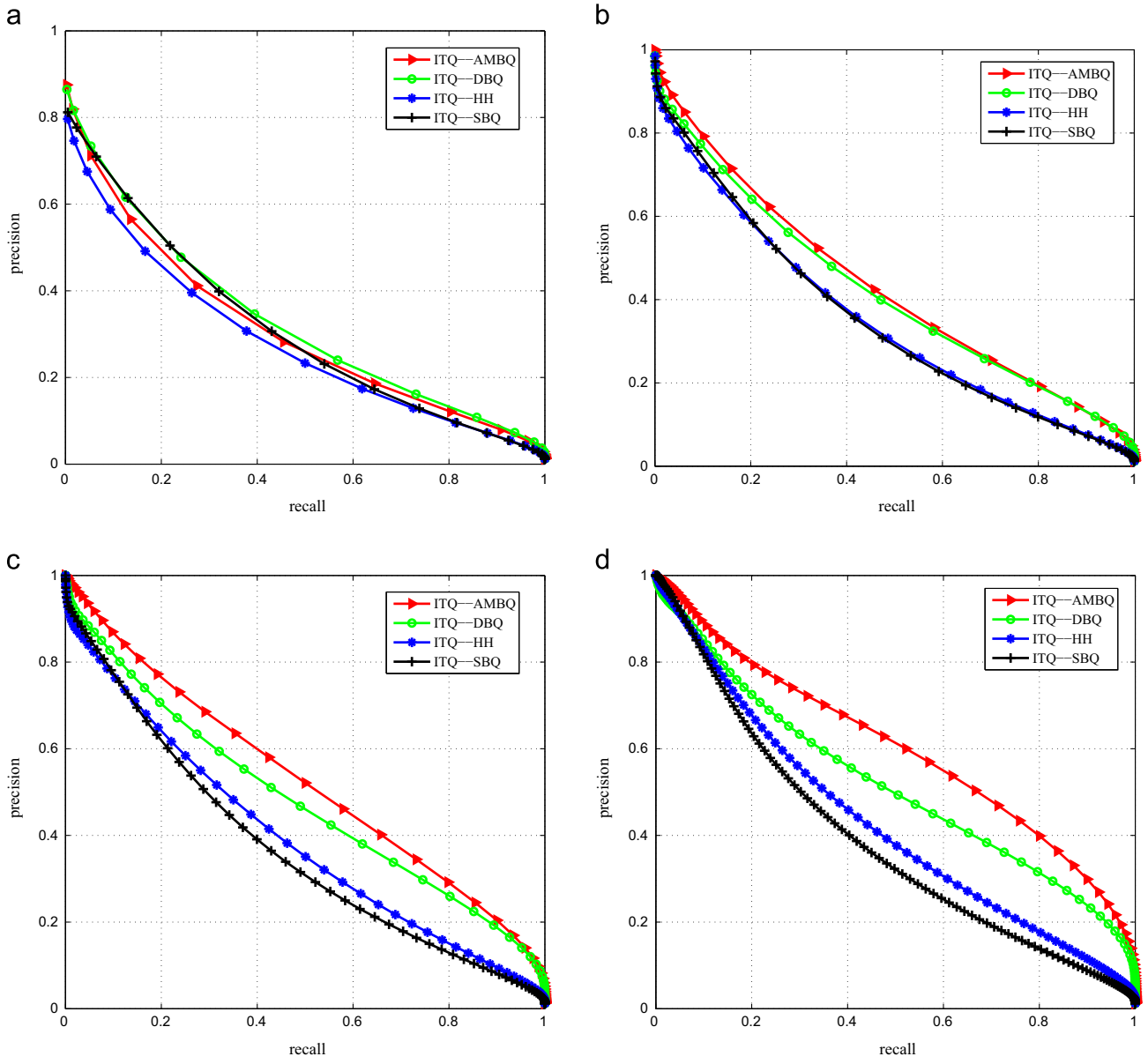


Fig. 5. Precision-recall curve for ITQ on 22*K-LabelMe* dataset. (a) ITQ 32 bits. (b) ITQ 64 bits. (c) ITQ 128 bits. (d) ITQ 256 bits.

parameter λ is just used to combining the quantization error and the affinity error together, which means it does not have a available method to guide this parameter selection. In the experiments, we verify different λ values within a large range, and find that the performance can reach to the best when λ is set to 10. Fig. 3 gives an explicitly demonstration of this parameter selection.

4.3. Accuracy

The mAP values for different methods using different code lengths on the CIFAR-10 and 22 K-LabelMe datasets, respectively, are shown in Tables 1 and 2. Each entry is the mAP of a combination of a hashing method with a quantization method for a specific code length, and the best mAP for SBQ, DBQ, and AMBQ using the same settings is marked in bold. For instance, in Table 1, for a code length of 32 bits using LSH, the mAP of SBQ (0.1569) is best when compared with DBQ (0.1130) and AMBQ (0.1075). From Tables 1 and 2 we find that when the code length is small (such as 32 bits), the performance of the data-dependent methods (i.e., PCA and ITQ) is superior to the performance of the data-independent methods (i.e., LSH and KLSH). The main reason may lies in when we project the original high-dimensional feature to the low-dimensional counterpart (e.g. 32), our multi-bit quantization method still allocate unequal bit number to each projected dimension, which results in the energy overly concentrate in a few projected dimensions, and instead is inferior to those methods with equal bit allocation strategy. Moreover, ITQ has relatively high mAP values for those settings with small code length, further validating our claims [16,23].

Figs. 4–6 show precision-recall curves for LSH, ITQ, and PCA with different code lengths on the 22 K-LabelMe dataset. It is clear that AMBQ outperforms SBQ, HH, and DBQ in most settings. Due to space limitations, the curves for CIFAR-10 are not shown but AMBQ still outperforms the other quantization methods on this dataset. More specifically, when the code length is increased from 32 to 256 bits, the mAP values of our method are significantly boosted.

It is worth noting that AMBQ outperforms HH and DBQ but not SBQ for KLSH, mainly because the nonlinear mapping adopted in KLSH has a significant impact on the neighborhood structure of the data.

Fig. 7 separately compares just HH and AMBQ due to space limitations, but illustrates that our AMBQ method has the highest

Please cite this article as: C. Deng, et al., Adaptive multi-bit quantization for hashing, Neurocomputing (2014), http://dx.doi.org/10.1016/j.neucom.2014.09.033

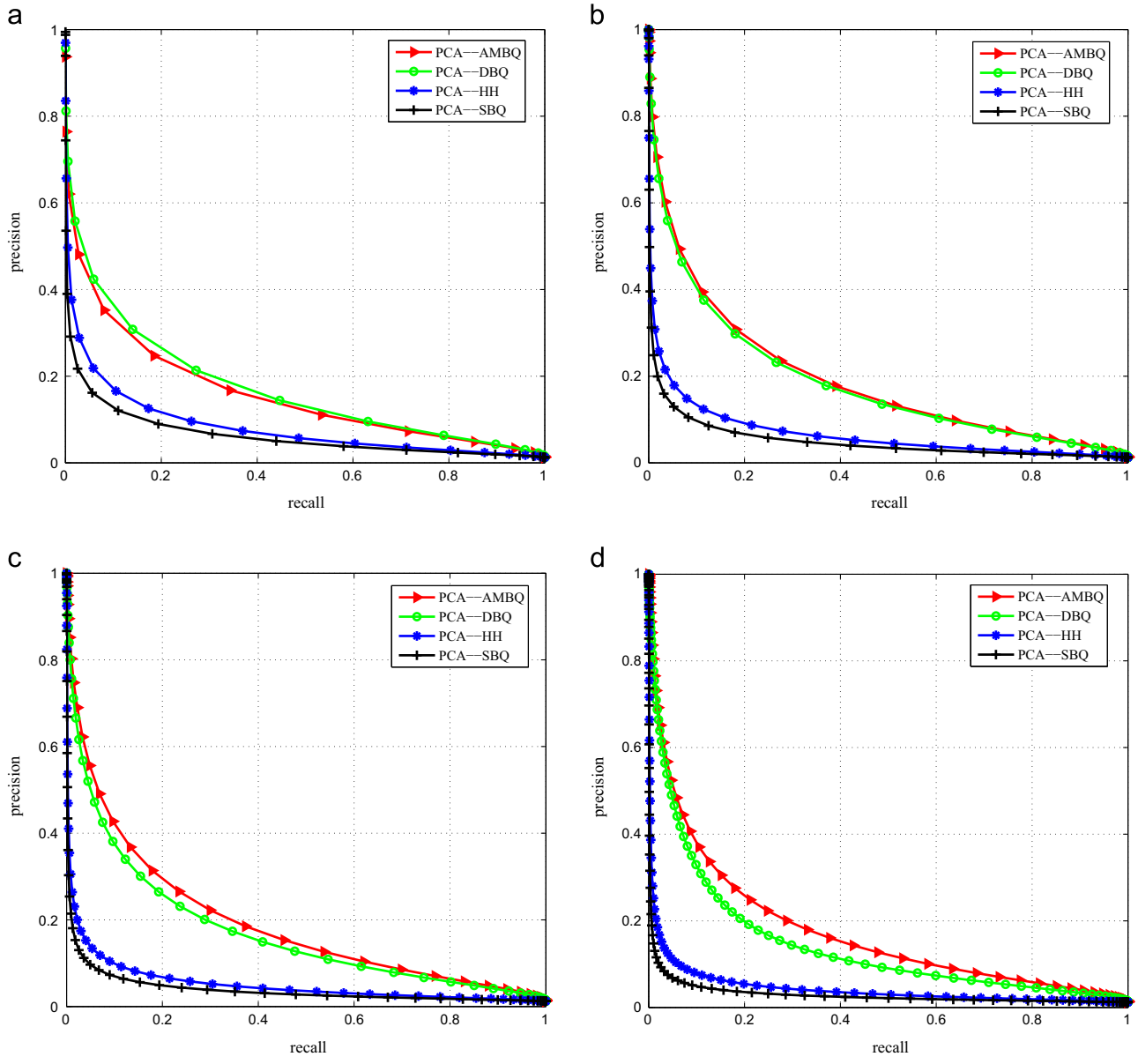


Fig. 6. Precision-recall curve for PCA on 22*K-LabelMe* dataset. (a) PCA 32 bits. (b) PCA 64 bits. (c) PCA 128 bits. (d) PCA 256 bits.

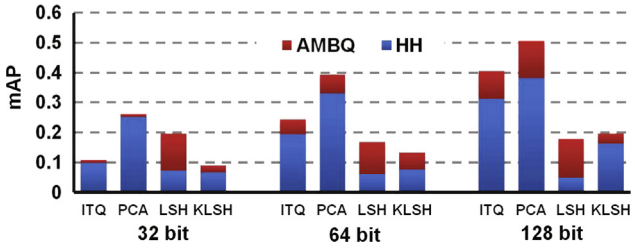


Fig. 7. mAP comparison between HH and AMBQ on 22*K-LabelMe* dataset.

Table 3
Training time on *CIFAR-10* dataset (in seconds).

No. of bits	32		64		256	
	DBQ	AMBQ	DBQ	AMBQ	DBQ	AMBQ
ITQ	8.46	5.23	14.1	9.52	80.1	96.8
PCA	3.93	5.35	4.01	8.3	4.58	19.1
LSH	0.21	2.25	0.27	4.53	0.71	17.3
KLSH	122.3	128.4	122.7	132.4	131.7	166.1

mAP values for ITQ, PCA, and LSH when the code lengths are 64 bits, 128 bits, and 256 bits, respectively, on both datasets.

4.4. Timing

Table 3 compares the training time for different quantization methods on CIFAR-10. All results were obtained on a workstation with 24 GB RAM and a 6-cores 3.4 GHz CPU. Since AMBQ needs to iteratively calculate the cluster centers for each projected dimension, the training of AMBQ is slower than DBQ. However, the computational cost remains acceptable when providing such good performance.

5. Conclusion

In this paper, we present an adaptive multi-bit quantization methods (AMBQ) for hashing. In particular, we adequately investigate the problems of the existing hashing methods and mainly

Please cite this article as: C. Deng, et al., Adaptive multi-bit quantization for hashing, Neurocomputing (2014), http://dx.doi.org/10.1016/j.neucom.2014.09.033

consider two significant ones lying in the hash learning process, which includes the amount of information of different projected dimensions and the data distribution structure. Different from other methods, our AMBQ can learn the hash bits adaptively based on the amount of information of different dimensions, meanwhile, instead of traditional hash function learning methods which may destroy the data structure heavily, we adopt the clustering strategy to assign the codewords of data points. Comprehensive experiments have demonstrated the advantages of the proposed approach in terms of accuracy and scalability.

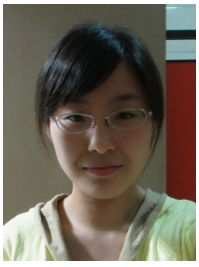
Acknowledgment

This work is supported by the National High Technology Research and Development Program of China (2013AA01A602), the National Natural Science Foundation of China (Nos. 61172143, 61101250 and 61402026), the Program for New Century Excellent Talents in University (NCET-12-0917), the Fundamental Research Funds for the Central Universities (No. K5051302019), the Key Science and Technology Program of Shaanxi Province, China (2014K05-16), Beihang YWF-14-JSJXY-010 and SKLSDE 2014ZX-07.

References

[1] H. Jegou, M. Douze, C. Schmid, Hamming embedding and weak geometric consistency for large scale image search, in: Proceedings of the European Conference on Computer Vision, 2009, pp. 304–317.
[2] C. Deng, R. Ji, W. Liu, D. Tao, X. Gao, Visual reranking through weakly supervised multi-graph learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2600–2607.
[3] X. Liu, J. He, D. Liu, B. Lang, Compact kernel hashing with multiple features, in: Proceedings of the 20th ACM International Conference on Multimedia, 2012, pp. 881–884.
[4] X. Liu, J. He, B. Lang, S.-F. Chang, Hash bit selection: a unified solution for selection problems in hashing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1570–1577.
[5] Q. Wang, G. Zhu, Y. Yuan, Statistical quantization for similarity search, Comput. Vis. Image Understand 124 (2014) 22–30.
[6] D.G. Lowe, Distinctive image features from scale invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[7] F. Zhu, L. Shao, Weakly-supervised cross-domain dictionary learning for visual recognition, Int. J. Comput. Vis. 109 (1–2) (2014) 42–59.
[8] L. Shao, L. Liu, X. Li, Feature learning for image classification via multiobjective genetic programming, IEEE Trans. Neural Netw. Learn. Syst. 25 (7) (2014) 1359–1371.
[9] Q. Wang, Y. Yan, Y. Yuan, X. Li, Multi-spectral saliency detection, Pattern Recogn. Lett. 34 (1) (2013) 34–41.
[10] Q. Wang, Y. Yuan, P. Yan, X. Li, Saliency detection by multiple-instance learning, IEEE Trans. Cybern. 43 (2) (2013) 660–672.
[11] L. Shao, X. Zhen, D. Tao, X. Li, Spatio-temporal Laplacian pyramid coding for action recognition, IEEE Trans. Cybern. 44 (6) (2014) 817–827.
[12] L. Shao, J. Simon, X. Li, Efficient search and localization of human actions in video databases, IEEE Trans. Circuits Syst. Video Technol. 24 (3) (2014) 504–512.
[13] G. Shakhnarovich, P. Viola, T. Darrell, Fast pose estimation with parameter-sensitive hashing, in: Proceedings of the IEEE Conference on Computer Vision, 2003, pp. 750–757.
[14] A. Gionis, P. Indyk, Similarity search in high dimensions via hashing, in: Proceedings of the International Conference on Very Large Data Bases, 2003, pp. 518–529.
[15] B. Kulis, K. Grauman, Kernelized locality sensitive hashing, IEEE Trans. Pattern Anal. Mach. Intell. 34 (6) (2012) 1092–1104.
[16] M. Raginsky, L. Svetlana, Locality-sensitive binary codes from shift-invariant kernels, in: Proceedings of the Advances in Neural Information Processing Systems, 2009, pp. 1509–1517.
[17] R. Salakhutdinov, H. Geoffrey, Semantic hashing, Int. J. Approx. Reason. 50 (7) (2009) 969–978.
[18] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: Proceedings of the Advances in Neural Information Processing Systems, 2008, pp. 1753–1760.
[19] 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.
[20] M. Norouzi, D. J. Fleet, Minimal loss hashing for compact binary codes, in: Proceedings of the International Conference on Machine Learning, 2011, pp. 353–360.
[21] B. Kulis, T. Darrell, Learning to hash with binary reconstructive embeddings, in: Proceedings of the Advances in Neural Information Processing Systems, 2009, pp. 1042–1050.
[22] W. Liu, J. Wang, S. Kumar, Hashing with graphs, in: Proceedings of the International Conference on Machine Learning, 2011, pp. 1–8.
[23] 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.
[24] X. Liu, J. He, C. Deng, B. Lang, Collaborative hashing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014.
[25] W. Kong, W.-J. Li, Isotropic hashing, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 1646–1654.
[26] W. Kong, W.-J. Li, Double-bit quantization for hashing, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012, pp. 634–640.
[27] R.M. Gray, Vector quantization, IEEE ASSP Mag. 1 (2) (1984) 4–29.
[28] K. He, F. Wen, J. Sun, 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, 2013, pp. 2938–2945.
[29] D.F. Shanno, Conditioning of quasi-Newton methods for function minimization. mathematics of computation, Math. Comput. 24 (111) (1970) 647–656.
[30] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, Technical Report, University of Toronto, 2009.
[31] A. Torralba, R. Fergus, Y. Weiss, Small codes and large image databases for recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[32] A. Oliva, A. Torralba, Modeling the shape of scene: a holistic representation of the spatial envelope, Int. J. Comput. Vis. 42 (3) (2001) 145–175.
[33] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximation nearest neighbor in high dimensions, Commun. ACM 51 (1) (2008) 117–122.

Cheng Deng received the B.Sc., M.Sc., and Ph.D. degrees in signal and information processing from Xidian University, Xi'an, China. He is currently a full professor with the School of Electronic Engineering at Xidian University. His research interests include computer vision, multimedia processing and analysis, and information hiding. He is the author and coauthor of more than 40 scientific articles at top venues, including IEEE TMM, TSMC, TIP, ICCV and CVPR.



Huiru Deng obtained the B.Sc. degree from Xidian University in 2012. Currently she is pursuing her M.S. degree at School of Electronic Engineering, Xidian University. Her research interest is majorly in large scale image/video retrieval.

Xianglong Liu received the B.S. and Ph.D. degrees in computer science from Beihang University, Beijing, in 2008 and 2014. From 2011 to 2012, he visited the Digital Video and Multimedia (DVMM) Lab, Columbia University as a joint Ph.D. student. He is now an assistant professor with Beihang University. His research interests include machine learning, computer vision and multimedia information retrieval.

Yuan Yuan is a full professor with the Chinese Academy of Sciences (CAS), China. She has published over 150 papers, including about 100 in reputable journals such as IEEE Transactions and Pattern Recognition, as well as conferences papers in CVPR, BMVC, ICIP, and ICASSP. Her current research interests include visual information processing and image/video content analysis.