Bilinear Optimized Product Quantization for Scalable Visual Content Analysis

Litao Yu, Zi Huang, Fumin Shen, Jingkuan Song, Heng Tao Shen* and Xiaofang Zhou

***Abstract*—Product quantization (PQ) has been recognized as a useful technique to encode visual feature vectors into compact codes to reduce both the storage and computation cost. Recent advances in retrieval and vision tasks indicate that high-dimensional descriptors are critical to ensuring high accuracy on large-scale datasets. However, optimizing PQ codes with high-dimensional data is extremely time and memory consuming. To solve this problem, in this paper, we present a novel PQ method based on bilinear projection, which can well exploit the natural data structure and reduce the computational complexity. Specifically, we learn a global bilinear projection for PQ, where we provide both non-parametric and parametric solutions. The non-parametric solution does not need any data distribution assumption. The parametric solution can avoid the problem of local optima caused by random initialization, and enjoys a theoretical error bound. Besides, we further extend this approach by learning locally bilinear projections to fit underlying data distributions. We show by extensive experiments that our proposed method, dubbed Bilinear Optimization Product Quantization (BOPQ), achieves competitive retrieval and classification accuracies while having significant lower time and space complexities.**

***Index Terms*—product quantization, bilinear projection, visual content analysis.**

I. Introduction

Large-scale data analysis has attracted ever increasing interests in many research areas, e.g., computer vision, biomedical analysis, and finance. For example, the ImageNet database, which contains 14 million images, plays a crucial role in improving many computer vision studies[1]. To aggregate the visual descriptors of images or videos, several high-dimensional image representation approaches have been proposed, such as Locality-constrained Linear Codes (LLC) [29], Fisher Vectors (FV) [21] or Vectors of Locally Aggregated Descriptors (VLAD) [11]. There is evidence that it is necessary to represent images or videos as very high-dimensional feature vectors to achieve high classification or retrieval accuracies [18], [20]. A key challenge is that processing and storing the large-scale and high-dimensional data are extremely time and memory consuming. To deal with the problem of *curse of dimensionality*, encoding the high-dimensional data into compact codes is a promising way for scalable visual content retrieval and classification in massive of data.

L. Yu, Z. Huang and X. Zhou are with the School of Information Technology and Electrical Engineering at The University of Queensland, Australia. Email: l.yu4@uq.edu.au; huang@itee.uq.edu.au; zxf@itee.uq.edu.au.

F. Shen, J. Song and H. T. Shen are with Center for Future Media and School of Computer Science and Engineering, University of Electronic Science and Technology of China. Email: fumin.shen@gmail.com; jingkuan.song@gmail.com; shenhengtao@hotmail.com.

*Corresponding author: Heng Tao Shen.

[1]http://www.image-net.org

Learning to encode the visual feature vectors into compact representations has a wide range of applications such as similarity search [31], [35], [26], reranking [41], classification [22] and clustering [37]. Among various encoding methods, binary embedding (or hashing) [30], [14], [24], [5], [6], [17], [42], [38], [23], [25] and vector quantization [3], [7], [16], [31], [10] are the most representative strategies. The idea of binary embedding is to represent feature vectors as compact binary codes, so the Euclidean distance between two vectors could be approximated by Hamming distance in the binary space. The advantage of binary embedding methods is due to the extremely efficient Hamming distance computation, which can be implemented by the XOR and POPCOUNT operations by modern CPUs. However, most of the binary embedding models suffer from significant information loss, because the number of possible distances is very limited and the distance approximation is thus less accurate.

Different from binary embedding based approaches, vector quantization adopts k-means and its variants to generate a codebook, and feature vectors can be quantized into a set of codewords. In this way, the distance between two vectors can be approximated by the distance between their corresponding codewords. When the original feature space is decomposed into the Cartesian product of several low-dimensional subspaces, vector quantization becomes product quantization (PQ) [10]. In each subspace, K-means algorithm is conducted to generate a centroid table as a codebook, which is used to quantize the high-dimensional vectors into compact codes. PQ has been successfully applied to many visual content analysis tasks, such as Approximate Nearest Neighbour (ANN) search and classification. For example, a comprehensive experimental study in [27] shows the combination of VLAD+PQ can lead to significant improvements on image classification and retrieval over most other encoding frameworks.

PQ has been shown to achieve better retrieval accuracy than binary embedding. However, its search speed is to some extent lower. The reason for this is that, as one of the most basic operators, XOR is faster than ADD operator. Vector quantization can be combined with inverted indexing to achieve non-exhaustive search and improve the efficiency. Also, the authors in [15] presented a PQ based hash table requiring neither parameter tuning nor training procedure, which can achieve considerably faster search on very large-scale datasets. It has shown that before generating a codebook to encode feature vectors into compact PQ codes, a random rotation of data could slightly improve the retrieval accuracy sometimes [10], which is analogous to the binary embedding model Locality Sensitive Hashing (LSH) [4]. In [2], Ge et al. formulated two

optimization strategies for PQ (OPQ), which can reduce the data distortion, resulting in the boosted performance. Yannis et al. extended OPQ by further considering data distortions locally, and their Locally Optimized PQ (LOPQ) achieved the state-of-the-art accuracies in retrieval tasks [12].

Despite the promising performance achieved by the optimization strategies, they suffer from prohibitive computations, particularly for high-dimensional features. For example, in the optimization of both OPQ and LOPQ, the storage and computation requirements for the full projection matrix increase quadratically and cubically, respectively, with the feature dimension. For example, a 32768×32768 full-rotation matrix with float data type takes about 4GB memory, so the rotation and factorization of large matrices are nearly infeasible when training with large-scale datasets on a machine with limited memory.

Inspired by the recent advances of the bilinear model in binary embedding [5], classification [19] and Linear Discriminant Analysis (LDA) [40], in this paper, we propose a bilinear projection based PQ optimization model called Bilinear Optimized Product Quantization (BOPQ). BOPQ can not only take advantage of the natural matrix structure of features such as FV and VLAD but also significantly reduce the time and memory overhead. The motivation of our proposed BOPQ is to achieve a sub-optimal solution to optimize the PQ codes, with a much less computational complexity but a satisfactory accuracy. The idea behind is simply applying two smaller mapping matrices instead of a large full-rotation matrix to minimize the data distortion. Suppose a feature vector is D-dimensional, by applying bilinear projection, the memory cost of the rotation matrix can be reduced from $O(D^2)$ to $O(D)$, and the time complexity can be reduced from $O(D^3)$ to $O(D^{\frac{3}{2}})$, respectively. Thus, our proposed BOPQ is more practical to optimize PQ codes for high-dimensional data.

The main contributions of our work are summarized as follows:

- We propose a novel bilinear projection based product quantization method named Bilinear Optimization Product quantization (BOPQ). In detail, we present both parametric and non-parametric solutions. The non-parametric solution is obtained by an efficient iterative optimization, while the parametric one assumes a Gaussian distribution and can avoid local optima caused by random initialization. In particular, we derive a theoretical error bound for the latter method.
- We further extend BOPQ by learning local bilinear rotations to minimizing data distortions in the local data areas. This approach can better fit the underlying data distributions and boost the search accuracy.
- We conducted extensive experiments on several large-scale datasets for ANN search, image retrieval, and video classification. The results show that our proposed BOPQ produces very little degradation in accuracy while has significantly lower running time and memory usage in optimization.

The rest of the paper is organized as follows. Section II introduces the preliminaries of PQ and some optimized PQ approaches. In section III we elaborate the bilinear optimized PQ in detail, including both non-parametric and parametric solutions to globally optimize the PQ codes using bilinear projections, and its extensions of local optimization methods. Experimental results and analysis are presented in section IV, and section V concludes the paper.

TABLE I: Notations used in this paper

Symbols	Explanation
$\boldsymbol{x}$	The original feature vector to represent a data sample
$\hat{\boldsymbol{x}}$	The feature vector in the rotated feature space
$\mathbf{x}$	The matrix representation of $\boldsymbol{x}$
$q(\cdot)$	The encoding function
$t(\cdot)$,$t^{-1}(\cdot)$	The function converting vector to matrix, and its reverse
$\mathcal{C}$	The codebook for PQ
$\hat{\mathcal{C}}$	The codebook for OPQ and BOPQ
K	The size of sub-codebook in each subspace
M	The fixed number of subspaces for PQ methods
$\boldsymbol{I}$	The identity matrix
$\mathbf{c}$	The codeword generated by Kmeans
$\hat{\mathbf{c}}$	The optimized codeword by data rotation
$\boldsymbol{R}$	The full-rotation matrix used in OPQ
D	The dimension of $\boldsymbol{x}$
$\boldsymbol{R}^{(\cdot)}$	The full-rotation matrices used in LOPQ
$\boldsymbol{R}_1$,$\boldsymbol{R}_2$	The bilinear matrices used in our proposed BOPQ-G
d_1, d_2	The dimensions of $\mathbf{x}$
$\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$	The covariance matrices in the original feature space
$\hat{\boldsymbol{\Sigma}}_1, \hat{\boldsymbol{\Sigma}}_2$	The covariance matrices in the rotated feature space
$\boldsymbol{e}$	The codeword of a coarse cluster
$\boldsymbol{z}$	The residual between $\boldsymbol{x}$ and $\boldsymbol{e}$
$\mathbf{z}$	The matrix format of $\boldsymbol{z}$
V	The number of coarse clusters
$\mathcal{E}$	The codebook of the coarse clusters
$\mathcal{Z}$	The residual cell of a coarse cluster
$\boldsymbol{R}_1^{(\cdot)}$,$\boldsymbol{R}_2^{(\cdot)}$	The bilinear matrices used in our proposed BOPQ-L

II. Preliminary

For the engineering aspect of a scalable visual content analysis, Product Quantization (PQ) is one of the most popular techniques to compress the high-dimensional vector representations. Compared with binary embedding, PQ based encoding methods generally have a lower information loss. The idea of PQ is to decompose feature vectors into several sub-vectors with equal lengths, then K-means algorithm is applied on each sub-vector to generate a centroid table as a codebook. Subsequently, all sub-vectors are approximated by the nearest centroids and encoded into PQ codes. Despite the widely-studied retrieval task, the compact data representation can potentially significantly reduce the memory and disk space for facilitating other tasks, such as visual recognition [13], [28], [39], and multimodal learning tasks [34], [32], [33].

Let's first have a brief review of Vector Quantization (VQ). The notations used in this paper are listed in Table I. A vector quantizer is to map a vector $\boldsymbol{x} \in \mathbb{R}^D$ to a codeword $\mathbf{c}$ in a codebook $\mathcal{C}$. The mapping is represented as $\boldsymbol{x} \mapsto q(\boldsymbol{x})$, where $q(\cdot)$ is an encoder. Given a codebook $\mathcal{C}$, an encoder satisfies the first Lloyd's condition, i.e., the encoder $q(\boldsymbol{x})$ always maps the vector $\boldsymbol{x}$ to its nearest codeword $\mathbf{c}$ in $\mathcal{C}$. Usually, the codebook is generated by K-means. When the encoder $q(\boldsymbol{x})$ is fixed, a codeword $\mathbf{c}$ is the mean of the vectors, which is the second Lloyd's condition.

When the codeword $\mathbf{c}$ is taken from the Cartesian product of sub-codebooks, such VQ becomes Product Quantization

(PQ). In this case, let $\boldsymbol{x}$ be a D-dimensional vector, which can be concatenated by M sub-vectors with equal lengths, i.e., $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \ldots, \boldsymbol{x}_M]$, $1 < m < M$. The dimension of each sub-vector is D/M. x's nearest codeword $\mathbf{c} \in \mathcal{C}$ is the concatenation of the M nearest sub-codewords $\mathcal{C} = [c_1, \ldots, c_m, \ldots, c_M]$, where the Cartesian product $\mathcal{C} = \mathcal{C}_1 \times \ldots, \times \mathcal{C}_M$ is a codebook generated by K-means. Assume all sub-codebooks have K sub-codewords, i.e., the number of each cluster centroids in each sub-space is K, the Cartesian product $\mathcal{C}$ can represent at most K^M data samples in Euclidean space. In this case, the vector $\boldsymbol{x}$ can be encoded by M quantizers as $q(\boldsymbol{x}) = [q_1(\boldsymbol{x}_1), \ldots, q_m(\boldsymbol{x}_m), \ldots, q_M(\boldsymbol{x}_M)]$. As a result, for large-scale datasets, we only need to store the compact PQ codes and a lookup table.

In visual search tasks, there are two ways to compute the approximate distance between two vectors: symmetric and asymmetric distance calculations. For symmetric distance calculation (SDC), the distance in each subspace is approximated by the distance of their sub-codewords. In this case, the distances between any two sub-codewords are pre-computed and stored in a K-by-K lookup table. For asymmetric distance calculation (ADC), the distance in each subspace is calculated online and stored in a 1-by-K lookup table. In either case, the distance in the original space can be efficiently computed by plus operations in M subspaces.

In [11], Jegou et al. proposed to optimize a Householder Transform so the vectors have balanced variances in all components, and suggested that a random rotation has a similar performance to the Householder transform when prior knowledge is unavailable. However, the orthogonal random projection cannot guarantee if the rotation is even sub-optimal to reflect the data distribution. To deal with this problem, Ge et al. proposed the optimized PQ model (OPQ) [2]. In this model, they defined the *quantization distortion* as:

$$E = \frac{1}{n} \sum_{\boldsymbol{x}} \|\boldsymbol{x} - \mathbf{c}\|^2, \tag{1}$$

where $\mathbf{c}$ is the code word generated in the original feature space.

The statistical relationship between data distortion and distance estimation has been discussed in [10]. Therefore, it is natural to optimize the original PQ by minimizing the data distortion. The objective function of OPQ is:

$$\begin{aligned} \min_{\hat{\mathcal{C}}, \boldsymbol{R}} \quad & \sum_{\boldsymbol{x}} \|\boldsymbol{R}^\top \boldsymbol{x} - \hat{\mathbf{c}})\|^2 \\ \text{s.t.} \quad & \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}, \hat{\mathbf{c}} = q(\boldsymbol{R}^\top \boldsymbol{x}) \\ & \hat{\mathbf{c}} \in \hat{\mathcal{C}} = \hat{\mathcal{C}}_1 \times \ldots \times \hat{\mathcal{C}}_M, \end{aligned} \tag{2}$$

where $\hat{\mathbf{c}}$ is the codeword generated in the rotated feature space.

The mapping matrix $\boldsymbol{R}$ arbitrary rotates data and permutes vector components, to better align to data distribution and balance their variance. With the consideration of these factors, the performance of PQ can be further improved. To solve the problem in Eq. (2), they provided two solutions: non-parametric and parametric approaches. In the non-parametric approach, the objective is to minimize the data distortion iteratively with an orthogonal constraint. In the parametric approach, the rotation matrix $\boldsymbol{R}$ is directly optimized by balanced partition.

In their experiments, the non-parametric solution has a slightly better performance than the parametric one, because it does not have any assumption of data distribution. The basic idea of the non-parametric solution is to optimize an orthogonal matrix $R \in \mathbb{R}^{D \times D}$ to rotate the data, to minimize the data distortion. However, the final result is affected by the initialization of the rotation matrix $\boldsymbol{R}$. The parametric solution assumes the Gaussian distribution of data, which has a simpler algorithm to calculate $\boldsymbol{R}$ by eigenvector alignment. With such an assumption, it has a solid theoretical explanation to the distortion bound.

Applying a global rotation matrix $\boldsymbol{R}$ to minimize the data distortion may not well reflect the local data structure. In [12], Yannis et al. proposed the Locally Optimized Product Quantization (LOPQ) to optimize each centroid individually to align the local data distribution. This idea is based on the fact that no single centroids should be wasted by not representing actual data. The LOPQ algorithm first generates several coarse clusters, then for each cluster, a full-rotation matrix is optimized in the local data area. In this method, the quantization is conducted on data residuals because it can effectively reflect the data distances to the coarse centroids. Similarly, Heo et al. proposed the Distance Encoded PQ (DPQ) that adopts additional bits to quantize the distance from the sub-vectors to their closest centroids [8].

Both OPQ and LOPQ can achieve better retrieval accuracy than the original PQ method, since minimizing the data distortion is well considered as the objective to help adjust locations of the PQ codes. However, for very high-dimensional vectors, the calculation of full-rotation matrices is extremely time and memory consuming, which is unaffordable when memory is limited. In the next section, we propose several bilinear projection strategies to optimize PQ codes, which can effectively reduce the computation complexity, while keeping a satisfactory performance for visual analysis tasks.

III. Bilinear optimized PQ

In visual feature representations, many kinds of high-dimensional vectors have a natural matrix structure, i.e., a vector could be represented as a matrix. Let $\boldsymbol{x} \in \mathbb{R}^D$ be a vector, $D = d_1 \times d_2$, so $\boldsymbol{x}$ can be reshaped as $\mathbf{x} \in \mathbb{R}^{d_1 \times d_2}$ with the transformation t:

$$\mathbf{x} = t(\boldsymbol{x}). \tag{3}$$

The reshaping of $\boldsymbol{x}$ can be either row-wise or column-wise. Correspondingly, $\boldsymbol{x}$ can be recovered by flattening $\mathbf{x}$:

$$\boldsymbol{x} = t^{-1}(\mathbf{x}). \tag{4}$$

The optimized product quantization (OPQ) is basically applying a full-rotation matrix $\boldsymbol{R}$ to the original feature vector $\boldsymbol{x}$:

$$q(\hat{\boldsymbol{x}}) = q(\boldsymbol{R}^\top \boldsymbol{x}). \tag{5}$$

Instead of transforming a vector $\boldsymbol{x}$ with a full-rotation matrix $R \in \mathbb{R}^{D\times D}$, we use two smaller matrices $\boldsymbol{R}_1 \in \mathbb{R}^{d_1\times d_1}$ and $\boldsymbol{R}_2 \in \mathbb{R}^{d_2\times d_2}$ to map $\mathbf{x}$ by bilinear projection. Consequently, the encoding function becomes:

$$q(\hat{\boldsymbol{x}}) = q(t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2)). \tag{6}$$

When $\boldsymbol{R}$ satisfies the condition $\boldsymbol{R} = \boldsymbol{R}_2 \otimes \boldsymbol{R}_1$, where $\otimes$ is the Kronecker product, the bilinear projection is equivalent to the full rotation:

$$t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2) = (\boldsymbol{R}_2^\top \otimes \boldsymbol{R}_1^\top) t^{-1}(\mathbf{x}) = \boldsymbol{R}^\top t^{-1}(\mathbf{x}). \tag{7}$$

It can be easily seen that when both $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are orthogonal, $\boldsymbol{R}$ is also orthogonal. As a result, the bilinear projection is a special case of full rotation. Although the freedom degree of bilinear projection has more restrictions than the full rotation, using two smaller matrices can substantially reduce the memory cost and accelerate the computation. For example, the eigenvalue decomposition on $\boldsymbol{R}$ has the time complexity $O(d_1^3 d_2^3)$ and memory $O(d_1^2 d_2^2)$. However, the same computation on $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ only needs $O(d_1^3 + d_2^3)$ in time and $O(d_1^2 + d_2^2)$ in memory, respectively. In the ideal case, it only takes $O(2D^{\frac{3}{2}})$ in time and $O(2D)$ in memory. However, the condition $d_1 = d_2 = \sqrt{D}$ is hardly satisfied, so d_1 and d_2 should be selected as the closest positive integers.

In the next two subsections, we present two strategies, global and local bilinear optimization methods, to facilitate encoding high-dimensional feature vectors into compact PQ codes.

A. Learning a global bilinear projection for PQ

When two small matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are randomly initialized and orthogonalized, transforming $\mathbf{x}$ into a new space is equivalent to random full-rotation [11]. Here we will not discuss the random rotation since it can be further optimized to achieve better performance. Similar to the ideas given in [2], [6], [5], our target is to find a global bilinear projection such that the angle between a rotated feature vector $\boldsymbol{x}$ and its codeword $\hat{\mathbf{c}}$ is minimized, i.e., their cosine similarity is maximized:

$$\begin{aligned}
&\sum_{\boldsymbol{x}} \cos(\boldsymbol{x}, \hat{\mathbf{c}}) \\
=&\sum_{\boldsymbol{x}} \frac{q(\boldsymbol{R}^\top \boldsymbol{x})^\top (\boldsymbol{R}^\top \hat{\mathbf{c}})}{\sqrt{D}} \\
=&\sum_{\mathbf{x}} \frac{q(t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2))^\top t^{-1}(\boldsymbol{R}_1^\top \hat{\mathbf{c}} \boldsymbol{R}_2)}{\sqrt{D}} \\
=&\frac{1}{\sqrt{D}} \sum_{\mathbf{x}} (\hat{\mathbf{c}}^\top t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2)).
\end{aligned} \tag{8}$$

Using the full-rotation matrix $\boldsymbol{R}$ to optimize codewords is quite challenging when the dimensionality of $\boldsymbol{x}$ is extremely high. However, when we apply bilinear projection as mentioned above, the optimization is much more practical because both $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are much smaller. Thus, the objective function of globally bilinear optimized PQ (BOPQ-G) becomes:

$$\max_{\hat{\mathcal{C}}, \boldsymbol{R}_1, \boldsymbol{R}_2} \quad \sum_{\mathbf{x}} (\hat{\mathbf{c}}^\top t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2)). \tag{9}$$

The above objective function is equivalent to the minimization of the quantization distortion as Eq. (1). Considering the orthoganal constraints and Cartesian product, the problem of BOPQ-G is formulated as:

$$\begin{aligned}
\min_{\hat{\mathcal{C}}, \boldsymbol{R}_1, \boldsymbol{R}_2} \quad & \sum_{\mathbf{x}} \|t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2) - \hat{\mathbf{c}}\|^2 \\
\text{s.t.} \quad & \boldsymbol{R}_1^\top \boldsymbol{R}_1 = \boldsymbol{I}, \boldsymbol{R}_2^\top \boldsymbol{R}_2 = \boldsymbol{I}, \\
& \hat{\mathbf{c}} = q(t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2)), \\
& \hat{\mathbf{c}} \in \hat{\mathcal{C}} = \hat{\mathcal{C}}_1 \times \ldots \times \hat{\mathcal{C}}_M.
\end{aligned} \tag{10}$$

To solve this problem, we introduce two algorithms: non-parametric and parametric solutions in the next two subsections in detail.

1) A non-parametric solution: We first provide a non-parametric solution of (10) called BOPQ-G$_{\text{NP}}$, i.e., we do not assume any data distribution as prior knowledge.

Since both $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are orthogonal, the following condition is satisfied:

$$\|\boldsymbol{x} - \mathbf{c}\|^2 = \|t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2) - \hat{\mathbf{c}}\|^2. \tag{11}$$

We split the problem (10) into three sub-problems to optimize $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\hat{\mathbf{c}}$ respectively. Suppose the number of training samples is n, the iterative update scheme is described as follows:

First, we fix $\hat{\mathcal{C}}$ and $\boldsymbol{R}_1$, and update $\boldsymbol{R}_2$. Let $\boldsymbol{X}_2 = [\boldsymbol{R}_1^\top \mathbf{x}_1; \ldots; \boldsymbol{R}_1^\top \mathbf{x}_n] \in \mathbb{R}^{(d_1\times n)\times d_2}$, and $\boldsymbol{Y}_2 = [t(q(\boldsymbol{x}_1)); \ldots; t(q(\boldsymbol{x}_n))] \in \mathbb{R}^{(d_1\times n)\times d_2}$. The sub-problem becomes:

$$\begin{aligned}
\min_{\boldsymbol{R}_2} \quad & \|\boldsymbol{X}_2 \boldsymbol{R}_2 - \boldsymbol{Y}_2\|^2 \\
\text{s.t.} \quad & \boldsymbol{R}_2^\top \boldsymbol{R}_2 = \boldsymbol{I}.
\end{aligned} \tag{12}$$

The above problem with an orthogonal constraint has a closed-form solution: we can conduct singular vector decomposition (SVD) to $\boldsymbol{Y}_2^\top \boldsymbol{X}_2 = \boldsymbol{U}_2 \boldsymbol{S} \boldsymbol{V}_2$, and then let $\boldsymbol{R}_2 = \boldsymbol{U}_2 \boldsymbol{V}_2^\top$.

Second, we fix $\hat{\mathcal{C}}$ and $\boldsymbol{R}_2$, and update $\boldsymbol{R}_1$. Let $\boldsymbol{X}_1 = [\mathbf{x}_1 \boldsymbol{R}_2, \ldots, \mathbf{x}_n \boldsymbol{R}_2] \in \mathbb{R}^{d_1\times(d_2\times n)}$ and $\boldsymbol{Y}_1 = [t(q(\boldsymbol{x}_1)), \ldots, t(q(\boldsymbol{x}_n))] \in \mathbb{R}^{d_1\times(n\times d_2)}$. The sub-problem becomes:

$$\begin{aligned}
\min_{\boldsymbol{R}_1} \quad & \|\boldsymbol{R}_1^\top \boldsymbol{X}_1 - \boldsymbol{Y}_1\|^2 \\
\text{s.t.} \quad & \boldsymbol{R}_1^\top \boldsymbol{R}_1 = \boldsymbol{I}.
\end{aligned} \tag{13}$$

Similar to $\boldsymbol{R}_1$, the optimization of $\boldsymbol{R}_2$ can be achieved by SVD: $\boldsymbol{Y}_1 \boldsymbol{X}_1^\top = \boldsymbol{U}_1 \boldsymbol{S} \boldsymbol{V}_1$, and $\boldsymbol{R}_1 = \boldsymbol{U}_1 \boldsymbol{V}_1^\top$.

Third, we fix $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$, and update the codebook $\hat{\mathcal{C}}$. Denote $\hat{\boldsymbol{x}} = t^{-1}(\boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2)$, the sub-problem is:

Algorithm 1 A Non-parametric solution of Globally Bilinear Optimized Product Quantization (BOPQ-G_{NP})

Input: Training samples $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, number of subspaces M, number of sub-codewords k in each sub-codebook.
Output: A codebook $\hat{\mathcal{C}}$, and two rotation matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$.

1: Initialize $\boldsymbol{R}_1 = \boldsymbol{I}$ and $\boldsymbol{R}_2 = \boldsymbol{I}$, or initialize them with random orthogonal matrices;
2: Initialize M sub-codebooks $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_M$ with $k \times M$ codewords by conducting K-means;
3: **repeat**
4: Update $\boldsymbol{R}_2$ by solving the problem of Eq. (12);
5: Update $\boldsymbol{R}_1$ by solving the problem of Eq. (13);
6: Use bilinear projection Eq. (3) to rotate each training sample $\boldsymbol{x}$ to $\mathbf{x}$, then recover it to $\hat{\boldsymbol{x}}$ according to Eq. (4).
7: Update $\hat{\mathcal{C}}$ using K-means in each subspace of the training samples as in Eq. (14);
8: **until** Convergence;
9: Return $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\hat{\mathcal{C}}$.

$$\begin{aligned} \min_{\hat{\mathcal{C}}} \quad & \sum_{\hat{x}} \|\hat{\boldsymbol{x}} - \hat{\mathbf{c}}\|^2 \\ \text{s.t.} \quad & \hat{\mathbf{c}} = q(\hat{\boldsymbol{x}}), \\ & \hat{\mathbf{c}} \in \hat{\mathcal{C}} = \hat{\mathcal{C}}_1 \times \ldots \times \hat{\mathcal{C}}_M. \end{aligned} \tag{14}$$

This is the objective function of original PQ, and we can just run K-means in each subspace to compute the sub-codebooks individually.

The three updates can be cycled for several iterations to shrink to local optima, and the convergence is guaranteed in the finite iterations. Also, in the iterative procedure, we do not need to restart the K-means when generating the sub-codebooks. Instead, the centroids can be refined using the previous sub-codebooks with only one K-means iteration.

The iterative solution of globally bilinear optimized PQ (BOPQ-G_{NP}) is summarized in Algorithm 1.

The problem is neither convex nor concave, so the non-parametric solution shrinks to local optima, and the final result is dependent on the initialization. However, an advantage of such non-parametric solution is that it does not need any assumption of data distribution.

2) A parametric solution: Next, we propose a parametric solution called BOPQ-G_P, which could be obtained by two eigenvalue allocations. Different from the non-parametric solution discussed above that relies on random initialization, the parametric solution can directly optimize two projection matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$.

Assume that when a vector $\boldsymbol{x}$ is represent as a matrix $\mathbf{x}$ according to Eq. (3), it follows a 2D independent Gaussian distribution with zero mean, i.e., $\mathbf{x} \sim \mathcal{N}(0, 0, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, 0)$, where $\boldsymbol{\Sigma}_1 \in \mathbb{R}^{d_1 \times d_1}$ and $\boldsymbol{\Sigma}_2 \in \mathbb{R}^{d_2 \times d_2}$ are covariance matrices. If we apply bilinear projection on $\mathbf{x}$, the variable $\hat{\mathbf{x}} = \boldsymbol{R}_1^\top \mathbf{x} \boldsymbol{R}_2$ is subject to another 2D Gaussian distribution $\hat{\mathbf{x}} \sim \mathcal{N}(0, 0, \hat{\boldsymbol{\Sigma}}_1, \hat{\boldsymbol{\Sigma}}_2, 0)$, with $\hat{\boldsymbol{\Sigma}}_1 = \boldsymbol{R}_1^\top \boldsymbol{\Sigma}_1 \boldsymbol{R}_1$ and $\hat{\boldsymbol{\Sigma}}_2 = \boldsymbol{R}_2 \boldsymbol{\Sigma}_2 \boldsymbol{R}_2^\top$. The two covariance matrices $\hat{\boldsymbol{\Sigma}}_1$ and $\hat{\boldsymbol{\Sigma}}_2$ can be decomposed into two $M \times M$ sub-matrices:

$$\hat{\boldsymbol{\Sigma}}_1 = \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_1^{11} & \ldots & \hat{\boldsymbol{\Sigma}}_1^{1M} \\ \vdots & \ddots & \vdots \\ \hat{\boldsymbol{\Sigma}}_1^{M1} & \ldots & \hat{\boldsymbol{\Sigma}}_1^{MM} \end{pmatrix}, \tag{15}$$

and

$$\hat{\boldsymbol{\Sigma}}_2 = \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_2^{11} & \ldots & \hat{\boldsymbol{\Sigma}}_2^{1M} \\ \vdots & \ddots & \vdots \\ \hat{\boldsymbol{\Sigma}}_2^{M1} & \ldots & \hat{\boldsymbol{\Sigma}}_2^{MM} \end{pmatrix}. \tag{16}$$

The diagonal matrices $\hat{\boldsymbol{\Sigma}}_1^{mm}$ and $\hat{\boldsymbol{\Sigma}}_2^{mm}$ are the covariance of m-th subspace, $1 \leq m \leq M$. According to the *rate distortion theory* [1] and [2], [9], the lower bounds of two-dimensional distortions satisfy:

$$\sum_{m=1}^{M} |\hat{\boldsymbol{\Sigma}}_1^{mm}|^{\frac{M}{d_1}} \geq M |\boldsymbol{\Sigma}_1|^{\frac{1}{d_1}}, \tag{17}$$

and

$$\sum_{m=1}^{M} |\hat{\boldsymbol{\Sigma}}_2^{mm}|^{\frac{M}{d_2}} \geq M |\boldsymbol{\Sigma}_2|^{\frac{1}{d_2}}. \tag{18}$$

Based on the above analysis, we propose a direct method to optimize PQ by applying eigenvalue allocation twice. In detail, we first transform each data point $\boldsymbol{x}$ to $\mathbf{x}$ via Eq.(3), then optimize $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ as follows:

1. Let $\boldsymbol{X}_1 = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d_1 \times (d_2 \times n)}$. First, we apply PCA on $\boldsymbol{X}_1^\top$, keep all eigenvectors, and sort all eigenvalues in descending order. Then we prepare M buckets, and sequentially pick out the largest eigenvalue and allocate it to the bucket with the minimum product of the eigenvalues in it. After that, we reform each column of $\boldsymbol{R}_1$ using the eigenvectors corresponding to the eigenvalues in the bucket.

2. Let $\boldsymbol{X}_2 = [\mathbf{x}_1; \ldots; \mathbf{x}_n] \in \mathbb{R}^{(d_1 \times n) \times d_2}$. The calculation of $\boldsymbol{R}_2$ is the same as the re-order of eigenvectors on $\boldsymbol{X}_2$ in optimizing $\boldsymbol{R}_1$.

In summary, the parametric solution first computes the two smaller covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, then applies a column-wise eigenvalue allocation to obtain $\boldsymbol{R}_1$ and a row-wise eigenvalue allocation to get $\boldsymbol{R}_2$. Note that in order to make sure all the buckets are allocated to at least one eigenvalue, the following constraint should be satisfied:

$$M \leq \min(d_1, d_2). \tag{19}$$

In the encoding phase, for a new data point $\boldsymbol{x}$, we just apply Eq. (3) to represent it as a matrix $\mathbf{x}$, then conduct bilinear projection using $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$. Finally, we use Eq. (6) to encode it into several PQ codes.

Interestingly, if we apply the assumption of 2D Gaussian distribution to the objective of BPBC [5], it can derive that the distortion of BPBC is minimized when the 2D variances of the components are balanced. This indicates the parametric solution of bilinear optimization can also be used to generate binary embedding functions.

B. Learning the local bilinear projections for PQ

From the BOPQ-G as described above, we can observe that the data distortion could be minimized when a proper bilinear projection is obtained. However, the underlying data distributions are not unique in real cases, so in the high-dimensional feature space, the global projection may not well reflect the local properties of data. In [12], Yannis et al. extended the optimized PQ for ANN search. In their proposed Locally Optimized PQ (LOPQ), the K-means algorithm is first conducted on the training dataset to generate several coarse clusters (cells), then for each of them, a rotation matrix is optimized individually. Since all data samples are partitioned into different cells, the quantization can be conducted on residuals, which surround the data areas near the corresponding centroids as "spines", rather than in the original data space. In retrieval tasks, the LOPQ generally has better performance when a query and its nearest neighbor are quantized in the same cell, and its success lies on the local rotation and augmented data granularity led by the quantization of residuals. However, in very few cases when some data points are located in the boundary areas of a cell, they are likely to be "pushed" far away into other cells from their original locations. Also, the quantization of residuals makes it impractical for other machine learning tasks such as classification, because the generated PQ codes can be only recovered in the local area rather than the whole feature space.

Let V be the number of coarse clusters, and $\mathcal{E} = \boldsymbol{e}_1, \ldots, \boldsymbol{e}_V\}$ be the codebook consists of V centroids. For a data point $\boldsymbol{x}$ in a cell, it first calculates its residual $\boldsymbol{z}$ between itself and the corresponding centroid $\boldsymbol{e} \in \mathcal{E}$:

$$\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{e}. \tag{20}$$

Assume $\mathcal{Z}^{(1)}, \ldots, \mathcal{Z}^{(V)}$ are V residual cells. The objective function of LOPQ is:

$$\begin{aligned} \min_{\hat{\mathbf{c}}, \boldsymbol{R}^{(v)}} \quad & \sum_{\boldsymbol{z} \in \mathcal{Z}^{(v)}} \|(\boldsymbol{R}^{(v)\top}\boldsymbol{z}) - \hat{\mathbf{c}})\|^2 \\ \text{s.t.} \quad & \boldsymbol{R}^{(v)\top}\boldsymbol{R}^{(v)} = \boldsymbol{I}, \\ & \hat{\mathbf{c}} = q(\boldsymbol{R}^{(v)\top}\boldsymbol{z}), \\ & \hat{\mathbf{c}} \in \hat{\mathcal{C}} = \hat{\mathcal{C}}_1 \times \ldots \times \hat{\mathcal{C}}_M, \\ & v = 1, \ldots, V. \end{aligned} \tag{21}$$

In the local optimization procedure, each rotation matrix is computed based on the residuals within the cell. Finally, the sub-codebooks are generated by K-means on all full-projected residuals.

The storage requirement for LOPQ is $O(VD^2)$ since there are V projection matrices to locally rotate feature vectors, to minimize the residual distortion. When the dimensionality of the feature vectors is very high, solving the problem is even more difficult if memory is limited. To deal with this problem, in [12] the authors also proposed a multi-index strategy, which separately optimizes per-cell of two subspace quantizers and encodes two sub-residuals. The drawback of multi-index is that the covariance of the full dimensions is lost, which negatively affects the quality of data. In this paper, we only consider the single-index case, because the bilinear projection is able to reduce the memory cost effectively.

Algorithm 2 Locally Bilinear Optimized Product Quantization (BOPQ-L)

Input: Training samples $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, number of coarse clusters V, number of subspaces M, number of sub-codewords k in each sub-codebook.
Output: A codebook $\hat{\mathcal{C}}$, and V local rotation matrix pairs $(\boldsymbol{R}_1^{(1)}, \boldsymbol{R}_2^{(1)}), \ldots, (\boldsymbol{R}_1^{(V)}, \boldsymbol{R}_2^{(V)})$.

1: Generate V coarse clusters (cells) by conducting K-means on $\boldsymbol{X}$;
2: For each data sample $\boldsymbol{x}$ in a cell, calculate the residual $\boldsymbol{z}$ as Eq. (20);
3: **for** $v = 1, \ldots, V$ **do**
4: Collect residuals in cell $\mathcal{Z}^{(v)}$;
5: Optimize $\boldsymbol{R}_1^{(v)}$ and $\boldsymbol{R}_2^{(v)}$ by eigenvalue allocation;
6: Use $\boldsymbol{R}_1^{(v)}$ and $\boldsymbol{R}_2^{(v)}$ to project each residual in the cell;
7: **end for**
8: Collect all local residual projections, then apply K-means on M subspaces to generate the codebook $\hat{\mathcal{C}}$ with $K \times M$ codewords;
9: Return $(\boldsymbol{R}_1^{(1)}, \boldsymbol{R}_2^{(1)}), \ldots, (\boldsymbol{R}_1^{(V)}, \boldsymbol{R}_2^{(V)})$ and $\hat{\mathcal{C}}$.

Different from BOPQ-G as discussed in the previous subsection that optimizes two global rotation matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$, we use a set of small matrix pairs $\{(\boldsymbol{R}_1^{(1)}, \boldsymbol{R}_2^{(1)}), \ldots, (\boldsymbol{R}_1^{(V)}, \boldsymbol{R}_2^{(V)})\}$ to implement V local bilinear projections. We name the locally bilinear optimized PQ as BOPQ-L, and its objective function is:

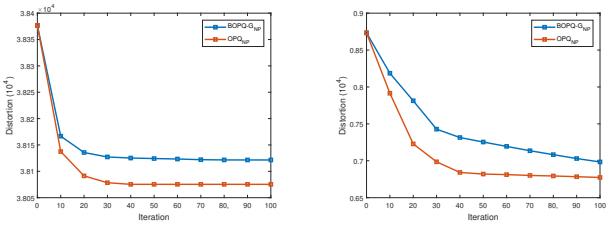
$$\begin{aligned} \min_{\hat{\mathbf{c}}, \boldsymbol{R}_1^{(v)}, \boldsymbol{R}_2^{(v)}} \quad & \sum_{t^{-1}(\mathbf{z}) \in \mathcal{Z}^{(v)}} \|t^{-1}(\boldsymbol{R}_1^{(v)\top}\mathbf{z}\boldsymbol{R}_2^{(v)}) - \hat{\mathbf{c}})\|^2 \\ \text{s.t.} \quad & \boldsymbol{R}_1^{(v)\top}\boldsymbol{R}_1^{(v)} = \boldsymbol{I}, \boldsymbol{R}_2^{(v)\top}\boldsymbol{R}_2^{(v)} = \boldsymbol{I}, \\ & \hat{\mathbf{c}} = q(t^{-1}(\boldsymbol{R}_1^{(v)\top}\mathbf{z}\boldsymbol{R}_2^{(v)})), \\ & \hat{\mathbf{c}} \in \hat{\mathcal{C}} = \hat{\mathcal{C}}_1 \times \ldots \times \hat{\mathcal{C}}_M, \\ & v = 1, \ldots, V. \end{aligned} \tag{22}$$

Within each residual cell $\mathcal{Z}^{(v)}$, the local rotation matrices $\boldsymbol{R}_1^{(v)}$ and $\boldsymbol{R}_2^{(v)}$ can be optimized in a similar way to the algorithms described in the previous subsection. The non-parametric solution is comparably slow because it needs iterative optimization. Also, its objective value is dependent on the initialization, and all the bilinear projection pairs should be consistent. Based on these reasons, we only focus on the parametric solution of BOPQ-L, which is described in Algorithm 2.

In the encoding phase, for a new data point $\boldsymbol{x}$, we first find its nearest centroid of the coarse clusters, then calculate its residual according to Eq. (20). After that, it is projected to a new space using the local bilinear rotation. Finally, $\boldsymbol{x}$ is encoded based on the pre-trained codebook $\hat{\mathcal{C}}$.

TABLE II: Complexities of different optimized PQ methods

Method	Time	Space
OPQ_{NP}	$O(T(d_1^2 d_2^2 + d_1 d_2 nK + d_1^3 d_2^3))$	$O(d_1^2 d_2^2)$
OPQ_{P}	$O(d_1^2 d_2^2 + d_1^3 d_2^3 + d_1 + d_2)$	$O(d_1^2 d_2^2)$
LOPQ	$O(V(d_1^2 d_2^2 + d_1^3 d_2^3 + d_1 + d_2))$	$O(V d_1^2 d_2^2)$
$\text{BOPQ-G}_{\text{NP}}$	$O(T(d_1^2 + d_2^2 + d_1 d_2 nK + d_1^3 + d_2^3))$	$O(d_1^2 + d_2^2)$
BOPQ-G_{P}	$O(d_1^2 + d_2^2 + d_1^3 + d_2^3 + d_1 + d_2)$	$O(d_1^2 + d_2^2)$
BOPQ-L	$O(V(d_1^2 + d_2^2 + d_1^3 + d_2^3 + d_1 + d_2))$	$O(V(d_1^2 + d_2^2))$



(a) SIFT1M ($M = 4$) (b) GIST1M ($M = 15$)

Fig. 1: The convergence curves of OPQ_{NP} and $\text{BOPQ-G}_{\text{NP}}$ on SIFT1M and GIST1M dataset

C. Complexity analysis of BOPQ

Since our proposed BOPQ adopts two small matrices to rotate original data, it can significantly reduce both time and memory cost when optimizing PQ codes. When a D dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^D$ is mapped to a matrix $\mathbf{x} \in \mathbb{R}^{d_1 \times d_2}$ that satisfies $D = d_1 d_2$, the projection time can be reduced from $O(d_1^2 d_2^2)$ to $O(d_1^2 + d_2^2)$. In full projection, the time complexity of SVD is $O(d_1^3 d_2^3)$, which can be reduced to $O(d_1^3 + d_2^3)$. When there are M subspaces, n training samples, and $K \times M$ codewords, one iteration of K-means requires $O(\frac{DnK}{M})$, so the overall complexity of K-means is $O(DnK)$. For parametric solutions OPQ_{P}, LOPQ, BOPQ-G_{P} and BOPQ-L, eigenvalue allocation generally needs a linear time complexity, i.e., $O(d_1 + d_2)$. Besides of the advantage in time, using two smaller matrices to project the high-dimensional feature data can effectively reduce the memory cost from $O(d_1^2 d_2^2)$ to $O(d_1^2 + d_2^2)$. Suppose the number of iterations is T, the optimizing time and space complexities are summarized in Table II.

IV. Experiments and Analysis

The performance of our proposed bilinear optimized PQ is evaluated in three applications: ANN search, image retrieval, and video classification.

A. Datasets and tasks

The experiment of ANN search is conducted on SIFT1M and GIST1M datasets[2]. The typical ANN search is to apply an encoding strategy to represent data points as compact codes, which allows fitting millions or billions of data in memory. The task of ANN search is to find the ground-truth nearest neighbors in the original Euclidean feature space. In the on-line search phase, each query is first encoded, then the distances to the data in the database are efficiently computed then ranked in ascending order. In SIFT1M dataset, there are 10k, 100k, and 1m data samples for query, training, and reference, respectively. The GIST1M contains 1k queries, 500k training samples, and 1m samples in the database, and it has a higher dimensionality of 960.

We test different encoding approaches of image retrieval on a combination of two datasets: Holiday[3] + Flickr1M [4]. In Holiday dataset, 1,419 images are corresponding to 500 different scenes. It is divided into two parts for the query (500 images) and reference (919 images) respectively. All samples in Flickr1M are completely irrelevant to the images in Holiday dataset, so we just mix the 919 images from the Holiday dataset with Flickr1M, to form a very large image database for the evaluation of retrieval. This task aims to retrieve the "relevant" or "semantically correct" neighbors. In image processing, we used VLfeat [5] toolkit to extract SIFT interesting points and applied K-means to generate a codebook with 128 codewords. Then we represented each image as a $128 \times 128 = 16,384d$ feature vector by VLAD aggregation. It is commonly known that high dimensional feature vectors usually contribute more accuracies in retrieval tasks. In the off-line training phase, we randomly selected 100,000 samples from reference images to train the encoding models and applied them to store all data.

B. Baselines and settings

The baselines are different encoding approaches, including PQ based methods and binary embedding methods as follows:

- **PQ** [10]: The original PQ method without any re-order of dimensions or optimization. The codebook is generated by K-means in different decomposed feature spaces;
- **Optimized PQ (OPQ)** [2]: Using a global rotation matrix to minimize data distortion. We use both non-parametric solution OPQ_{NP} and parametric solution OPQ_{P}.
- **Locally Optimized PQ (LOPQ)**[12]: Using different rotation matrices to locally minimize data distortion and quantize the residuals in different cells. Here we adopt single-indexing to encode the feature vectors using a unique encoding function.
- **Iterative Quantization (ITQ)** [6]: Applying the PCA and iterative optimization to generate binary codes;
- **Bilinear Projection-based Binary Codes (BPBC)** [5]: Applying PCA and a bilinear projection to generate binary codes;
- **Globally Bilinear Optimized PQ (BOPQ-G)**: Our proposed global bilinear optimized PQ, including the non-parametric solution $\text{BOPQ-G}_{\text{NP}}$ and parametric solution BOPQ-G_{P}.
- **Locally Bilinear Optimized PQ (BOPQ-L)**: Our proposed locally bilinear optimized PQ that utilizes different bilinear rotations and quantizes the local residuals. The local bilinear rotation matrices are optimized in a parametric way, i.e., eigenvalue allocation.

[2]http://corpus-texmex.irisa.fr/

[3]http://lear.inrialpes.fr/ jegou/data.php

[4]http://www.multimedia-computing.org/wiki/Flickr1M

[5]http://www.vlfeat.org/

TABLE IV: The comparison of recall on SIFT1M dataset

M	@	PQ	OPQ_{NP}	OPQ_{P}	LOPQ	BOPQ-G_{NP}	BOPQ-G_{P}	BOPQ-L
4	1	0.055	0.073	0.051	0.079	0.067	0.046	0.052
	5	0.152	0.194	0.150	0.218	0.181	0.124	0.145
	10	0.221	0.274	0.219	0.272	0.263	0.175	0.213
8	1	0.222	0.232	0.221	0.258	0.228	0.191	0.207
	5	0.472	0.487	0.470	0.587	0.477	0.412	0.413
	10	0.605	0.615	0.589	0.588	0.608	0.525	0.515

TABLE V: The comparison of recall on GIST1M dataset

M	@	PQ	OPQ_{NP}	OPQ_{P}	LOPQ	BOPQ-G_{NP}	BOPQ-G_{P}	BOPQ-L
15	1	0.082	0.125	0.123	0.134	0.124	0.129	0.130
	5	0.170	0.299	0.296	0.304	0.264	0.256	0.274
	10	0.225	0.388	0.386	0.380	0.370	0.338	0.366
30	1	0.126	0.187	0.186	0.213	0.161	0.193	0.195
	5	0.305	0.489	0.486	0.565	0.343	0.379	0.421
	10	0.411	0.594	0.592	0.618	0.482	0.494	0.533

TABLE III: The dimensions of bilinear projection matrices

Dataset	D	d_1	d_2
SIFT1M	128	8	16
GIST1M	960	30	32
Holiday+Flickr1M	16,384	128	128
TRECVID MED LCD	65,536	256	256
TRECVID MED fc_6	131,072	256	512
TRECVID MED fc_7	65,536	256	256

In PQ based methods, given the dimension D and the number of bits n_b assigned to each subspace, the number of subspace M is D/n_b. The number of centroids K in each subspace and M play a critical role in data representation. The higher values of K and M can better reflect the data distribution, leading to better accuracy in both retrieval and classification tasks. However, it has a higher storage requirement, and a slower computation. For the fair comparison, we set $K = 256$ without any change in the experiment, and set different values of M to observe the performance. Given the dimension D, each codeword of PQ can represent a D/M-bits sub-vector. For all PQ based ANN search and retrieval tasks in the experiment, we adopt asymmetric distance calculation (ADC) since it has a slightly better performance than symmetric distance calculation (SDC).

Since PQ and binary embedding have different encoding strategies, the parameter setting also varies. For ITQ and BPBC, we set the numbers of bits to 128, 256 and 512, respectively. In both PQ and binary embedding experiments, we use exhaustive search.

In our proposed BOPQ methods, the dimensions of the two rotation matrices d_1 and d_2 can be automatically computed in our program to make them as close as possible. The dimensions of the original features and bilinear matrices on these datasets are summarized in Table III.

We evaluated the ANN search and image retrieval performance via the *recall* measure, which is the proportion of queries having their top-ranked positions in the test set. Also, we used the Mean Average Precision (MAP) to evaluate the retrieval and classification performance.

Our experiment was conducted on a server with Intel(R) Xeon (R) E5-2690 v2 3.00GHz CPU with 20 processors, 256GB RAM, and Windows Server 2012 R2 operating system.

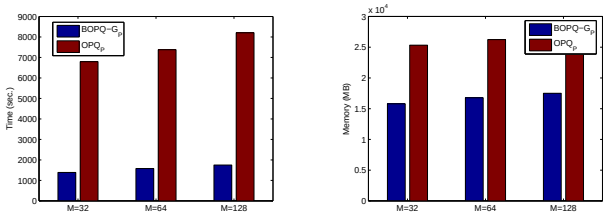
In order to facilitate the comprehensive understanding of our work, we have hosted the demo Matlab code of the BOPQ solutions on the website of our research group [6].

C. ANN search on SIFT1M and GIST1M datasets

Similar to OPQ_{NP}, BOPQ-G_{NP} is optimized iteratively to minimize the data distortion. We ran the OPQ_{NP} and BOPQ-G_{NP}, and recorded the objective values. Fig. 1(a) and Fig. 1(b) display the convergence curves. Like many other alternative solutions [5], [6], the two optimization models converge to local minima. The bilinear projection has more constraints and the equation in Eq. (7) can be hardly satisfied. Thus, its minimized data distortion is still higher than that of full-rotation. Besides, in each iterative step, the bilinear projection matrices are updated in a smaller scale, so BOPQ-G_{NP} has a lower convergence rate.

In Table IV and V, we show the comparison of ANN search results on the two datasets when the different number of subspaces were set. The SIFT descriptors have a low dimensionality of 128, so it seems unnecessary to use bilinear projections to optimize the PQ codes because the full-rotation matrix does not need too much memory and the computation is fast. In fact, three parametric solutions: OPQ_{P}, BOPQ-G_{P} and BOPQ-L even degraded the search accuracy compared to the original PQ model, while the only one parametric solution: LOPQ, and all non-parametric solutions: OPQ_{NP} and BOPQ-G_{NP}, boosted the search performance on this dataset. The reason for such phenomenon may be that the SIFT descriptor does not follow the Gaussian distribution, so eigenvalue allocation has no effect on minimizing the data distortion. However, when we did not assume the Gaussian distribution and adopted the non-parametric solutions, both OPQ_{NP} and BOPQ-G_{NP} achieved better results. On the GIST1M dataset, all optimized PQ methods can boost the search results compared to the original PQ. Also, the difference between the non-parametric solutions (OPQ_{NP} and BOPQ-G_{NP}) and parametric solutions (OPQ_{P} and BOPQ-G_{P}) is quite insignificant, which signifies the GIST data approximately follows the Gaussian distribution.

[6] http://xxx.xxx.xxx.xxx



(a) Time (b) Memory

Fig. 2: Comparison of training times and memory costs

Since the bilinear projection is a special case of full projection, and it has a stricter constraint, the data distortion values of BOPQ-G_{NP} are higher than that of OPQ_{NP} (see Fig. 1). In fact, the full rotation matrix R can be hardly decomposed to two smaller matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ as described in Eq. (7) , and this is the reason why the proposed BOPQ-G and BOPQ-L generally have a slightly lower accuracy than OPQ and LOPQ.

D. Image retrieval on Holiday+Flickr1M dataset

In this experiment, we first evaluated the running times on the server. The proposed BOPQ is specially designed for optimizing high-dimensional PQ codes. At the same time, the computation complexity is greatly reduced. We set the number of subspaces M to 32, 64 and 128, i.e., each codeword in these settings can represent a 512-bits, a 256-bits and a 128-bits code, respectively, and recorded the time and memory in optimization. Since the SVD and eigenvalue decomposition basically have the same computational complexity, here we only compare the parametric solutions. The time and memory consumption for non-parametric solutions are just multiple times than the parametric ones, which depends on the number of iterations. Fig.2 shows the training time and memory comparison between OPQ_P and BOPQ-G_P, where the blue and red bars represent the bilinear projection and full-rotation method respectively. Compared with the full-rotation method, applying bilinear projection on the 16,384d data, it takes less than 1/6 time and about a half memory of the full-rotation method in optimization.

Next, we evaluated the retrieval performance on the combined dataset. Fig. 3 and Fig.4 shows the recall curves and MAP results given the different number of quantized subspaces, respectively. We also plotted the Precision-Recall curves as shown in Fig.5.

From these figures, we can see that PQ based methods consistently outperform the hashing based encoding. The reason behind such phenomenon is that PQ methods have much less information loss when training the encoding models. Among different PQ methods, the original PQ has the lowest accuracy, which could be further optimized by a proper rotation in minimizing the data distortion. The full-rotation based optimization, LOPQ and OPQ_{NP}, still achieve the best performance on this dataset. Although our proposed BOPQ-G_{NP}, BOPQ-G_P and BOPQ-L are slightly inaccurate than OPQ and LOPQ, they still have very promising performance. The reason is the stricter constraint of bilinear projection, and the

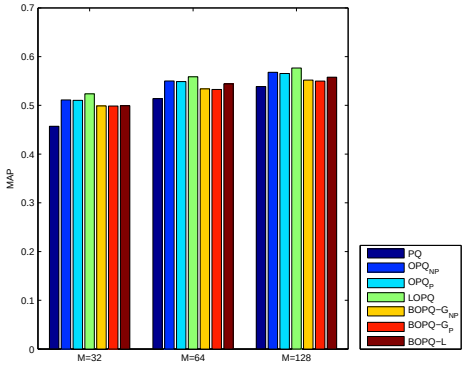


Fig. 4: The MAP comparison of Holiday+Flickr1M dataset

performance disparity of OPQ and BOPQ is consistent with the counterparts of binary embedding models ITQ and BPBC.

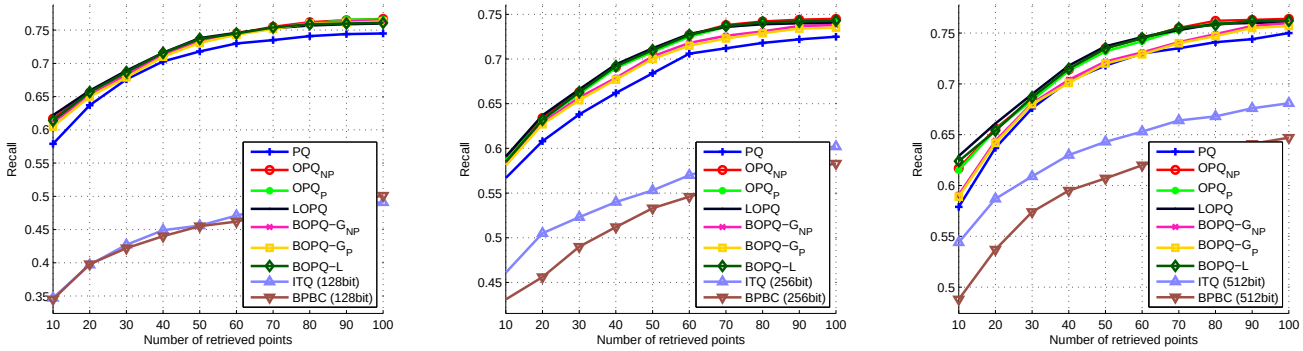
In the encoding of high-dimensional data, the disparity of retrieval accuracy between full-rotation and bilinear projection is much more insignificant than that on lower dimensional data such as SIFT1M and GIST1M. After applying VLAD aggregation on the raw visual descriptor, high-dimensional feature vectors naturally have a matrix structure, because the basic idea of VLAD is to quantize raw features into a matrix rather than a vector. This signifies the matrix structure in high-dimensional data is more natural and obvious, so bilinear optimization is more practical when there are limited computational resources.

An interesting phenomenon in Fig.3 is that the locally optimized PQ methods, i.e., LOPQ and BOPQ-L, have higher recall values than OPQ and BOPQ-G at top-ranked locations. However, when the number of retrieved points increases, the recalls of LOPQ and BOPQ-L are surpassed by OPQ and BOPQ-G. This is because, in the retrieval phase, the nearest neighbors are always searched in the locally projected area. Since the whole search space is segmented into several cells (coarse clusters) before generating the codebook, the closest data points are allocated to different areas.

E. Video classification on TRECVID MED datasets

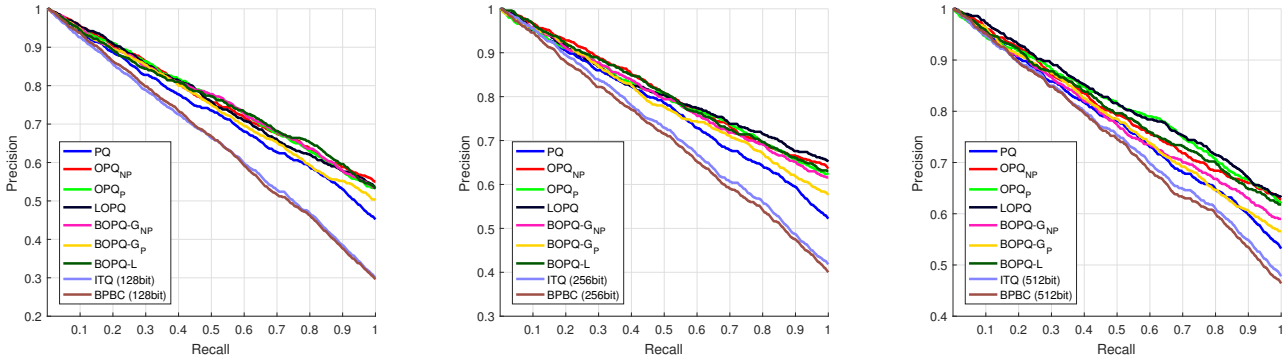
Finally, we demonstrated the effectiveness of the PQ compression strategy for video classification on TRECVID MED datasets. We used the CNN feature provided by Xu et al. [36], which consists of the VLAD quantization on the 5th pooling layer ($pool_5$ with 65,536d), the 6th and 7th fully-connected layers (fc_6 with 131,072d and fc_7, 65,536d). Applying the PQ encoding methods can greatly reduce the memory cost, while maintains the similar detection accuracies. In this experiment each codeword was used to represent a 32-bits length subvector, so the storage cost is only 1/32 of the visual features in original space.

Note that the locally optimized PQ methods, including LOPQ and BOPQ-L, are not available to compress the visual feature data for classification tasks. The reason for this is because all feature vectors are first allocated to their nearest coarse clusters (cells), then they are quantized based on their



(a) $M = 32$ (b) $M = 64$ (c) $M = 128$

Fig. 3: The recall curves on Holiday+Flickr1M dataset



(a) $M = 32$ (b) $M = 64$ (c) $M = 128$

Fig. 5: The ROC curves on Holiday+Flickr1M dataset

residuals within the cell. As a result, the compact PQ codes actually reflect the distances between the residuals of data and their centroids, and they cannot be recovered to the data in the original feature space. Based on these facts, we only compare the classification performance of different PQ methods in the global case.

The experiment of video event detection was conducted two subsets: MEDTest 13 100Ex and MEDTest 14 100Ex, and SVM with two different kernel types: linear kernel and χ^2 kernel, to evaluate the classification performance. When applying the SVM to train the classifiers and predict the test videos, we just used the compact PQ codes and the generated codebook to recover the feature vectors.

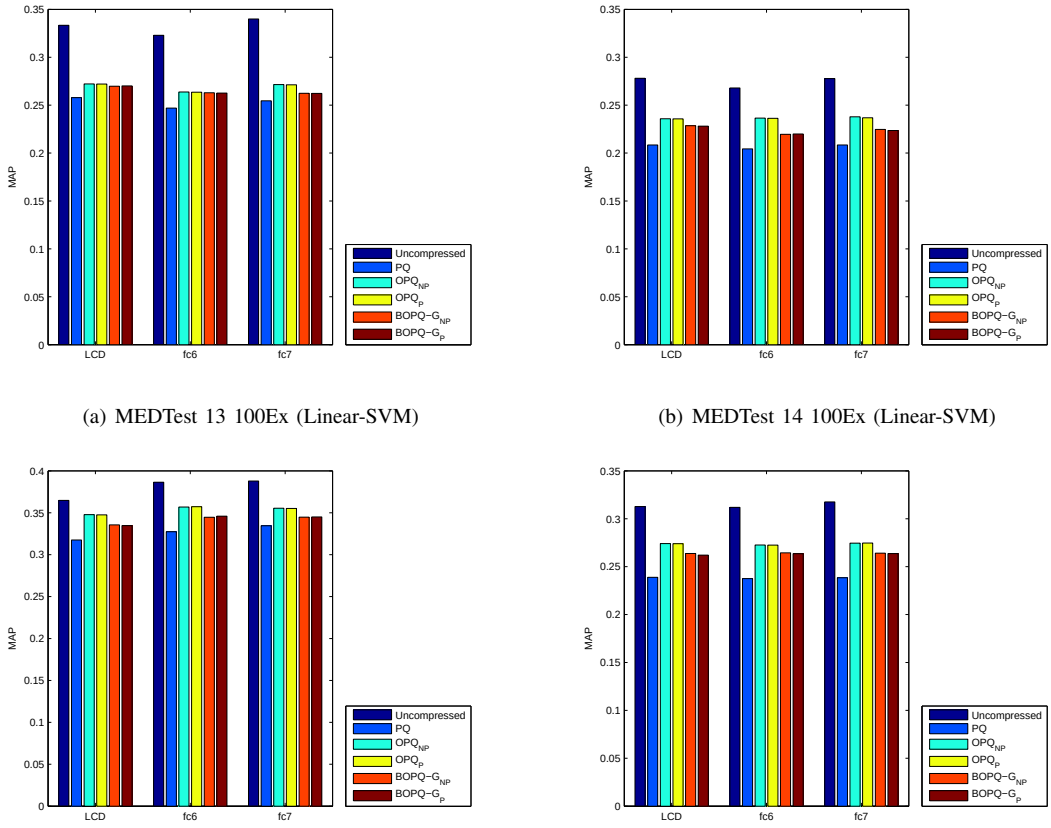
The classification results are reported in Fig. 6. If we simply apply the original PQ method to encode high-dimensional features, it incurs about 7% loss in MAP over the original CNN features. When a proper full-rotation matrix is used to optimize PQ codes, the loss of MAP can be reduced to 5% on this dataset. However, it is very expensive to obtain the full-rotation matrices since the dimensionality is extremely high. Applying our proposed bilinear projection methods can also achieve the similar results while having much less time and memory.

F. Discussion

We illustrated a few optimization strategies for product quantization, and they could be applied in different scenarios of content analysis tasks. Applying a full rotation matrix (OPQ_{NP} and OPQ_{P}) is suitable for indexing comparably low-dimensional data, while our proposed bilinear rotation (BOPQ_{NP} and BOPQ_{P}) is an alternative solution for high-dimensional data. It can considerably reduce the computational cost, but with a tolerable information loss. Both OPQ and BOPQ-G can be applied in information retrievals and other vision content analysis tasks, such as detection and recognition. The globally optimized PQ codes can be recovered by the codebook and rotation matrices. Their local extensions, LOPQ and BOPQ-L, are specifically designed for information retrieval especially when we care more about the top-ranked search results. However, they are not suitable for other learning tasks, because they quantize the residuals for each local cluster, and the original feature vectors cannot be recovered.

V. Conclusion

In this paper, we presented a novel bilinear rotation formulation for PQ optimization, which can exploit the natural matrix structure of visual feature vectors. We provided both non-parametric and parametric solutions to globally optimize



(a) MEDTest 13 100Ex (Linear-SVM)

(b) MEDTest 14 100Ex (Linear-SVM)

(c) MEDTest 13 100Ex (χ^2-SVM)

(d) MEDTest 14 100Ex (χ^2-SVM)

Fig. 6: The MAP comparisons on TRECVID MED dataset

the PQ codes using bilinear projections for different visual content analysis tasks. Also, we extended it to locally bilinear optimized PQ specifically for visual retrieval tasks. The accuracies of our proposed bilinear optimized PQ are very close to the full-projection ones when dealing with very high-dimensional data while being much more efficient in training encoding functions. As recent research shows the high-dimensional descriptor can lead to better performance in many visual content analysis tasks, it is very practical to apply our approach to such kind of data.

ACKNOWLEDGEMENTS

This work was supported in part by the Australian Research Council Project FT130101530, Project DP150103008 and in part by National Natural Science Foundation of China under Project 61502081 and Project 61632007 and in part by the Fundamental Research Funds for the Central Universities under Project ZYGX2014Z007.

REFERENCES

[1] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[2] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 36(4):744–755, 2014.

[3] A. Gersho and R. M. Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[4] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[5] Y. Gong, S. Kumar, H. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *IEEE CVPR*, pages 484–491, 2013.

[6] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

[7] R. Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[8] J.-P. Heo, Z. Lin, and S.-E. Yoon. Distance encoded product quantization. In *IEEE CVPR*, pages 2131–2138, 2014.

[9] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[10] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[11] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE CVPR*, pages 3304–3311, 2010.

[12] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *IEEE CVPR*, pages 2321–2328, 2014.

[13] T. Liu and D. Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):447–461, 2016.

[14] Y. Luo, Y. Yang, F. Shen, Z. Huang, P. Zhou, and H. T. Shen. Robust discrete code modeling for supervised hashing. *Pattern Recognition*, 2017. doi:10.1016/j.patcog.2017.02.034.

[15] Y. Matsui, T. Yamasaki, and K. Aizawa. Pqtable: Fast exact asymmetric distance neighbor search for product quantization using hash tables. In *IEEE ICCV*, 2015.

[16] N. M. Nasrabadi and R. A. King. Image coding using vector quantization: A review. *IEEE Trans. on Communications*, 36(8):957–971, 1988.

[17] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in hamming space with multi-index hashing. In *IEEE CVPR*, pages 3108–3115, 2012.

[18] F. Perronnin, Y. Liu, J. Snchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *IEEE CVPR*, pages 3384–3391, 2010.

[19] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Bilinear classifiers for visual recognition. In *NIPS*, pages 1482–1490, 2009.

[20] J. Sanchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *IEEE CVPR*, pages 1665–1672, 2011.

[21] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of Computer Vision*, 105(3):222–245, 2013.

[22] F. Shen, Y. Mu, Y. Yang, W. Liu, L. Liu, J. Song, and H. T. Shen. Classification by retrieval: Binarizing data and classifier. In *ACM SIGIR*, 2017.

[23] F. Shen, C. Shen, Q. Shi, A. van den Hengel, Z. Tang, and H. T. Shen. Hashing on nonlinear manifolds. *IEEE Transactions on Image Processing*, 24(6):1839–1851, 2015.

[24] F. Shen, Y. Yang, L. Liu, W. Liu, D. Tao, and H. T. Shen. Asymmetric binary coding for image search. *IEEE Transactions on Multimedia*, 2017. doi:10.1109/TMM.2017.2699863.

[25] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao. A fast optimization method for general binary code learning. *IEEE Transactions on Image Processing*, 25(12):5610–5621, 2016.

[26] J. Song, L. Gao, Y. Yan, D. Zhang, and N. Sebe. Supervised hashing with pseudo labels for scalable multimedia retrieval. In *ACM MM*, pages 827–830, 2015.

[27] E. Spyromitros-Xioufis, S. Papadopoulos, I. Y. Kompatsiaris, G. Tsoumakas, and I. Vlahavas. A comprehensive study over vlad and product quantization in large-scale image retrieval. *IEEE Transactions on Multimedia*, 16(6):1713–1728, 2014.

[28] D. Tao, Y. Guo, M. Song, Y. Li, Z. Yu, and Y. Y. Tang. Person reidentification by dual-regularized kiss metric learning. *IEEE Transactions on Image Processing*, 25(6):2726–2738, 2016.

[29] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *IEEE CVPR*, pages 3360–3367, 2010.

[30] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 2017.

[31] X. Wang, T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Supervised quantization for similarity search. In *IEEE CVPR*, pages 2018–2026, 2016.

[32] C. Xu, T. Liu, D. Tao, and C. Xu. Local rademacher complexity for multi-label learning. *IEEE Transactions on Image Processing*, 25(3):1495–1507, 2016.

[33] C. Xu, D. Tao, and C. Xu. Large-margin multi-view information bottleneck. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1559–1572, 2014.

[34] C. Xu, D. Tao, and C. Xu. Multi-view intact space learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(12):2531–2544, 2015.

[35] X. Xu, F. Shen, Y. Yang, H. T. Shen, and X. Li. Learning discriminative binary codes for large-scale cross-modal retrieval. *IEEE Transactions on Image Processing*, 26(5):2494–2507, 2017.

[36] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *IEEE CVPR*, pages 1798–1807, 2015.

[37] Y. Yang, F. Shen, Z. Huang, H. T. Shen, and X. Li. Discrete nonnegative spectral clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2017. doi:10.1109/TKDE.2017.2701825.

[38] Y. Yang, F. Shen, H. T. Shen, H. Li, and X. Li. Robust discrete spectral hashing for large-scale image semantic indexing. *IEEE Transactions on Big Data*, 1(4):162–171, 2015.

[39] Y. Yang, Z.-J. Zha, Y. Gao, X. Zhu, and T.-S. Chua. Exploiting web images for semantic video indexing via robust sample-specific loss. *IEEE Transactions on Multimedia*, 16(6):1677–1689, 2014.

[40] J. Ye, R. Janardan, and Q. Li. Two-dimensional linear discriminant analysis. In *NIPS*, pages 1569–1576, 2004.

[41] J. Yu, Y. Rui, and D. Tao. Click prediction for web image reranking using multimodal sparse coding. *IEEE Trans. on Image Processing*, 23(5):2019–2032, 2014.

[42] Z. Yu, F. Wu, Y. Yang, Q. Tian, J. Luo, and Y. Zhuang. Discriminative coupled dictionary hashing for fast cross-media retrieval. In *ACM SIGIR*, pages 395–404, 2014.