Fast Orthogonal Projection Based on Kronecker Product

Xu Zhang[1,2], Felix X. Yu[2,3], Ruiqi Guo[3], Sanjiv Kumar[3], Shengjin Wang[1], Shih-Fu Chang[2]
[1]Tsinghua University, [2]Columbia University, [3] Google Research

Abstract

We propose a family of structured matrices to speed up orthogonal projections for high-dimensional data commonly seen in computer vision applications. In this, a structured matrix is formed by the Kronecker product of a series of smaller orthogonal matrices. This achieves $\mathcal{O}(d \log d)$ computational complexity and $\mathcal{O}(\log d)$ space complexity for d-dimensional data, a drastic improvement over the standard unstructured projections whose computational and space complexities are both $\mathcal{O}(d^2)$. We also introduce an efficient learning procedure for optimizing such matrices in a data dependent fashion. We demonstrate the significant advantages of the proposed approach in solving the approximate nearest neighbor (ANN) image search problem with both binary embedding and quantization. Comprehensive experiments show that the proposed approach can achieve similar or better accuracy as the existing state-of-the-art but with significantly less time and memory.

1. Introduction

Linear projection is one of the most widely used operations, fundamental to many algorithms in computer vision. Given a vector $\mathbf{x} \in \mathbb{R}^d$, and a projection matrix $\mathbf{R} \in \mathbb{R}^{k \times d}$, the linear projection computes $h(\mathbf{x}) \in \mathbb{R}^k$:

$$h(\mathbf{x}) = \mathbf{R}\mathbf{x}.$$

In the area of large-scale search and retrieval in computer vision, linear projection is usually followed by quantization to convert high dimensional image features into binary embeddings [15, 23, 34, 25] or product codes [16, 26]. These compact codes have been used to speed up search and reduce storage in image retrieval [32], feature matching [31], attribute recognition [28], and object categorization [6] among others. For example, the popularly used Locality Sensitive Hashing (LSH) technique applies a linear projection to the input data before converting it into a binary or a non-binary code. For instance, a k-bit binary code is simply given as,

$$h(\mathbf{x}) = \text{sign}(\mathbf{R}\mathbf{x}). \quad (1)$$

However, the projection operation becomes expensive as the input dimensionality d increases. In practice, to achieve high recall in retrieval tasks, it is often desirable to use long codes with a large k such that $k = \mathcal{O}(d)$ [22, 8, 29]. In this case, the space and computational complexity of projection is $\mathcal{O}(d^2)$, and such a high cost often becomes the bottleneck at both learning and prediction time. For instance, when $k = d = 50,000$, projection matrix alone takes 10 GB (single precision) and projecting one vector can take 800 ms on a single core.

In addition, in many applications, it is desired that the projection matrix is orthogonal. An orthogonal transformation preserves the Euclidean distance between points. And it can also distribute the variance more evenly across the dimensions. These properties are important to make several well-known techniques perform well on real-world data [10, 26]. Another motivation for orthogonal projection comes from the goal of learning maximally uncorrelated bits while learning data-dependent binary codes. One way to achieve that is by imposing orthogonal (or near orthogonal) constraints on the projections [10, 37]. In binary embedding, many independent empirical experiments [18, 15, 31, 10] have shown that imposing orthogonality constraint on the projection achieves better results for approximate nearest neighbor search. Ji *et al*. [19] provided theoretical analysis to support the use of orthogonal projection. In quantization, the orthogonality constraint of the projection matrix makes some critical optimization problems possible to solve [26, 35]. However, the main challenge is that the computational complexity of building an orthogonal matrix is $\mathcal{O}(d^3)$ while space and projection complexity is $\mathcal{O}(d^2)$, both of which are expensive for large d.

In order to speed up projection for high-dimensional data, researchers have studied various types of structured matrices, including Hadamard matrix, Toeplitz matrix and circulant matrix. The idea behind projecting with structured matrices instead of the traditional unstructured matrices is that one can exploit the structure to achieve better

This work was supported by the National Science and Technology Support Program under Grant No. 2013BAK02B04 and Initiative Scientic Research Program of Tsinghua University under Grant No. 20141081253. Felix X. Yu was supported in part by the IBM PhD Fellowship Award when working on this project.

space and computational complexity than quadratic. Projecting vectors with structured matrices has been studied in a variety of contexts. In dimensionality reduction, Alon and Chazelle [1] studied projection using a sparse Gaussian matrix paired with a Hadamard matrix. This was followed by Dasgupta *et al*. [5] who used a combination of permutation and diagonal matrices along with Hadamard matrix in LSH. These variants of Hadamard matrices were further used by Jacques *et al*. in compressed sensing [14] and Le *et al*. [21] in kernel approximation. These works utilize the well-known fast Johnson-Lindenstrauss transform to achieve $\mathcal{O}(d \log d)$ time complexity. Researchers have also used Toeplitz-structured matrices [2, 11] and circulant matrices [12, 37, 38, 4] for projection, which also obtains $\mathcal{O}(d \log d)$ time complexity.

However, the main problem with the commonly used structured matrices is that they are not orthogonal. Although the Hadamard matrix is orthogonal by itself, it is typically used in combination with other matrices (e.g., sparse Gaussian or diagonal/permutation matrices), which convert it into a non-orthogonal matrix. Besides this, there is another problem with using the Hadamard matrix directly: there are no free-parameters in Hadamard matrices. Thus, one cannot learn a Hadamard matrix in a data-dependent fashion.

In this work we introduce a very flexible family of orthogonal structured matrices formed by Kronecker product of small element matrices, leading to substantially reduced space and computational complexity. One can vary the number of free parameters in these matrices to adapt to the needs of a given application. The most related work to our proposed method is the bilinear projection [8], which is also orthogonal and faster than quadratic. We show that the bilinear method can be viewed as a special case of the proposed method. Moreover, our structure is more flexible and has lower computational complexity than $\mathcal{O}(d^{1.5})$ of the bilinear method. Table 1 summarizes the space and time complexity of the proposed method in comparison with other structured matrices.

1.1. Our Contributions

In this work, we propose a novel method to construct a family of orthogonal matrices by using the Kronecker product of a series of small orthogonal matrices. Formally, the Kronecker projection matrix is defined as

$$\mathbf{R} = \mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_M,$$

where $\mathbf{A}_j$, $j = 1, \cdots, M$ are small orthogonal matrices. We term them as the element matrices. Such Kronecker product matrices have the following unique advantages: 1) They satisfy orthogonality constraint and therefore preserve Euclidean distance in the original space; 2) Similar to Hadamard and circulant matrices, there exists a fast algorithm to compute Kronecker projection with a time complexity of $\mathcal{O}(d \log d)$; 3) Changing the sizes of the small orthogonal matrices, the resulting matrix has varying number of parameters (degrees of freedom), making it easier to control performance-speed trade-off; 4) The space complexity is $\mathcal{O}(\log d)$ in comparison to $\mathcal{O}(d)$ for most other structured matrices.

Method	Time	Space	Time (Learning)
Unstructured [10]	$\mathcal{O}(d^2)$	$\mathcal{O}(d^2)$	$\mathcal{O}(Nd^2)$
Bilinear [8]	$\mathcal{O}(d^{1.5})$	$\mathcal{O}(d)$	$\mathcal{O}(Nd^{1.5})$
Circulant [37]	$\mathcal{O}(d \log d)$	$\mathcal{O}(d)$	$\mathcal{O}(Nd \log d)$
Kronecker (ours)	$\mathcal{O}(d \log d)$	$\mathcal{O}(\log d)$	$\mathcal{O}(Nd \log^2 d)$

Table 1. Computational and space costs. d: data dimensionality, N: number of training samples. The space and time complexities of Kronecker projection are based on $\mathbf{A}_j \in \mathbb{R}^{d_e \times d_e}, \forall j$, where d_e is a small constant.

We study Kronecker projection in two application settings: binary embedding, and quantization (Section 2). We propose a randomized version (Section 4) and a data-dependent learned version (Section 5) of such matrices. We conduct extensive image retrieval experiments on ImageNet-32768, ImageNet-16384, and Flickr-16384 datasets (Section 6). The results show that with fixed number of bits, the method needs much less space and time than the state-of-the-art methods to achieve similar or better performance.

2. Background

We begin by reviewing the two settings where the fast orthogonal projection based on Kronecker Product is applied: binary embedding, and quantization.

2.1. Binary Embedding

Binary embedding methods map original vectors into k-bit binary vectors such that $h(\mathbf{x}) \in \{+1, -1\}^k$. Since datapoints are stored as binary codes, the storage cost is reduced significantly even when $k = \mathcal{O}(d)$. The approximate nearest neighbors are retrieved using Hamming distance in the binary code space, which can be computed very efficiently using table lookup, or the POPCNT instruction on modern computer architectures.

Locality Sensitive Hashing (LSH) is a popular method for generating binary codes that preserves cosine distance [3, 27] and typically uses randomized projections in (1) to generate binary codes. However, many works have shown the advantages of learning data-dependent binary codes by optimizing the projection matrix $\mathbf{R}$ in (1) instead of using the randomized ones [20, 25, 36, 24, 8]. Specifically, Iterative Quantization (ITQ) [10] showed that by using a PCA projection followed by a learned orthogonal projection, the resulting binary embedding outperforms non-orthogonal or randomized orthogonal projection in retrieval

experiments. The projection is learned by alternating between projecting datapoints and solving for projections via SVD. However, for high dimensional features, this becomes infeasible unless one radically reduces the dimensionality, which hurts performance. We found that the projections learned with Kronecker product have similar performance as ITQ, while being substantially more efficient.

2.2. Quantization

Quantization methods represent datapoints via a set of quantizers, which are typically obtained by vector quantization algorithms such as k-means. To search for nearest neighbors of a given query $\mathbf{q}$, its Euclidean distances to all datapoints in the database are computed, which are approximated by vector-to-quantizer distances. Furthermore, when the data is high dimensional, quantization is often carried out in subspaces independently. A commonly used set of subspaces is obtained simply by chunking the vectors, which leads to the Product Quantization (PQ) [16, 7]. Formally, the distance between the query vector $\mathbf{q}$ and a database point $\mathbf{x}$ is given as:

$$||\mathbf{q}-\mathbf{x}|| \approx \sqrt{\sum_{i=1}^{m} ||\mathbf{q}^{(i)} - \mu_i(\mathbf{x}^{(i)})||^2},$$

where m is the total number of subspaces, $\mathbf{x}^{(i)}$ and $\mathbf{q}^{(i)}$ are subvectors and $\mu_i(\mathbf{x}^{(i)})$ is the quantization function on subspace i. Because of its asymmetric nature, only the database points are quantized, not the query vector. Quantization methods have been shown to outperform LSH based method in many cases [16]. However, for quantization methods to work well, it is desirable that different subspaces have similar variance for the given data. One way to achieve that is by applying an orthogonal transformation $\mathbf{R}$ to the data. Thus,

$$||\mathbf{q}-\mathbf{x}|| = ||\mathbf{R}\mathbf{q}-\mathbf{R}\mathbf{x}|| \approx \sqrt{\sum_{i=1}^{m} ||(\mathbf{R}\mathbf{q})^{(i)} - \mu_i(\mathbf{R}\mathbf{x})^{(i)}||^2}. \quad (2)$$

Since the projection matrix $\mathbf{R}$ is orthogonal, it also preserves the Euclidean distance. Norouzi and Fleet [26] proposed Cartesian k-means (ck-means) where they showed that instead of using a random projection matrix, it can be learned from given data leading to improved retrieval results. However, the projection operation can be time consuming in high-dimensional spaces.

In summary, for both binary embedding and quantization, a fast projection that is both orthogonal and learnable is needed. This motivates us to use the Kronecker product to design such projections, as described in the following sections.

3. Kronecker Product and Projection

We start by introducing the Kronecker product and its properties [33]. Let $\mathbf{A}_1 \in \mathbb{R}^{k_1 \times d_1}$, and $\mathbf{A_2} \in \mathbb{R}^{k_2 \times d_2}$. The Kronecker product of $\mathbf{A}_1$ and $\mathbf{A}_2$ is $\mathbf{A}_1 \otimes \mathbf{A}_2 \in \mathbb{R}^{k_1 k_2 \times d_1 d_2}$ defined as

$$\mathbf{A}_1 \otimes \mathbf{A}_2 = \begin{bmatrix} a_1(1,1)\mathbf{A}_2 & \cdots & a_1(1,d_1)\mathbf{A}_2 \\ a_1(2,1)\mathbf{A}_2 & \cdots & a_1(2,d_1)\mathbf{A}_2 \\ \vdots & \ddots & \vdots \\ a_1(k_1,1)\mathbf{A}_2 & \cdots & a_1(k_1,d_1)\mathbf{A}_2 \end{bmatrix},$$

where $a_1(i,j)$ is the element of the i-th row, and j-th column of $\mathbf{A}_1$. The Kronecker product is also known as the tensor product or direct product. We introduce two operations: $\text{mat}(\mathbf{x}, a, b)$ reshapes a d dimensional vector to an $a \times b$ matrix ($ab = d$), and $\text{vec}(\cdot)$ forms a vector by column-wise stacking the matrix into a vector, and $\text{vec}(\text{mat}(\mathbf{x}, a, b)) = \mathbf{x}$.

We state two properties of Kronecker product which will be used later in the paper:

- $(\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{x} = \text{vec}(\mathbf{A}_2 \text{mat}(\mathbf{x}, d_2, d_1)\mathbf{A}_1^T)$.
- The Kronecker product preserves the orthogonality. That is, if $\mathbf{A}_1$ and $\mathbf{A}_2$ are both orthogonal, $\mathbf{A}_1 \otimes \mathbf{A}_2$ is also orthogonal.

We define Kronecker projection matrix $\mathbf{R} \in \mathbb{R}^{k \times d}$ as the Kronecker product of several *element matrices*.

$$\mathbf{R} = \mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_j \otimes \ldots \otimes \mathbf{A}_M = \otimes_{j=1}^{M} \mathbf{A}_j,$$

where $\mathbf{A}_j \in \mathbb{R}^{k_j \times d_j}$ with $\prod_{j=1}^{M} k_j = k$ and $\prod_{j=1}^{M} d_j = d$.

One advantage of forming a large matrix in this way is that the Kronecker projection can be computed with a fast algorithm. In order to simplify the discussion, we assume that matrix $\mathbf{R}$ is square *i.e.*, $k = d$, and all the element matrices are also square with the same order d_e. We use floating points operations (FLOPs) to give an accurate estimate of the computational cost of different methods [13].

Let the FLOPs to compute the Kronecker projection on a d-dimensional vector, with element matrices of the order d_e, be $f(d, d_e)$. According to the property of Kronecker product,

$$\mathbf{R}\mathbf{x} = (\otimes_{j=1}^{M} \mathbf{A}_j)\mathbf{x} = \text{vec}((\otimes_{j=2}^{M} \mathbf{A}_j)\text{mat}(\mathbf{x}, d/d_e, d_e)\mathbf{A}_1^T).$$

Performing $\text{mat}(\mathbf{x}, d/d_e, d_e)\mathbf{A}_1^T$ needs $d(2d_e - 1)$ FLOPs (dd_e multiplications and $dd_e - d$ additions). After that computing $(\otimes_{j=2}^{M} \mathbf{A}_j)\text{mat}(\mathbf{x}, d/d_e, d_e)\mathbf{A}_1^T$ turns out to be d_e smaller scale problems, each computing a Kronecker projection with feature dimension d/d_e, and element matrix of order d_e. Therefore,

$$f(d, d_e) = d(2d_e - 1) + d_e f(d/d_e, d_e).$$

Based on the above recursive relation, the FLOPs of performing Kronecker projection on a d-dimension vector is $d(2d_e - 1)\log_{d_e} d$.

As a special case, when all the element matrices have an order of 2 (*i.e.*, $d_e = 2$) the FLOPs is $3d\log_2(d)$. When $\mathbf{R}$ is composed by a single square matrix of order d, the Kronecker projection becomes the unstructured projection. And when $M = 2$, the proposed method is exactly the bilinear projection [8]. The unstructured projection ($d_e = d$) requires $2d^2 - d$ FLOPs, and the bilinear projection requires at least $2d(2\sqrt{d} - 1)$ FLOPs, since $d_e = \sqrt{d}$. Circulant projection needs one d-dim real to complex FFT, one d-dim complex to real IFFT, and one $d/2$-dim complex multiplication [37]. The cost is then $4d\log_2 d$ FLOPs for large d, based on the split-radix implementation. Consequently, the Kronecker projection with small d_e (such as 2) has the lowest computational cost.

Another appealing property of Kronecker projection is the flexibility of its structure: by controlling the size of $\mathbf{A}_j$, $j = 1, \cdots, M$, one can easily balance the number of parameters (therefore the capacity) of the model and the computational cost. There are $\log_{d_e} d$ element matrices, each with d_e^2 parameters. The number of parameters in Kronecker projection is $d_e^2 \log_{d_e} d$, which ranges from d^2 (when $d_e = d$) to $4\log_2 d$ (when $d_e = 2$)[1].

Although we have only discussed the case when $\mathbf{R}$ and all the element matrices are square, the analysis can be easily extended to the non-square cases. In practice, the sizes of the element matrices can be chosen by factorizing d and k. When d or k cannot be factorized as the product of small numbers: for the input feature, one can change the dimension by subsampling or padding zeros; for the output, one can always use a longer code and then subsample. In Section 4 and Section 5, we will discuss the generation of Kronecker projection. First we assume the projection matrix $\mathbf{R}$ to be square. The nonsquare case will be discussed in Section 5.4.

4. Randomized Kronecker Projection

Similar to unstructured projection, circulant projection, and bilinear projection *etc*., Kronecker projection can be generated randomly. Generating an unstructured orthogonal matrix of order d has time complexity $\mathcal{O}(d^3)$. Therefore it is not practical for high-dimensional data.

For the Kronecker projection, one needs to generate M (small) element orthogonal matrices, which is done by generating small random Gaussian matrices and then performing QR factorization. If the element matrices are of the size 2×2, the time complexity of generating randomized Kronecker projection of order d is only $\mathcal{O}(\log d)$.

[1]We assume storing all d_e^2 parameters of an element matrix. Note that an orthonormal matrix has only $d_e(d_e - 1)/2$ degrees of freedom, so the number of parameters can be further reduced by at least 50%.

To apply the randomized Kronecker projection in binary embedding and quantization, we replace the unstructured projection matrix ($\mathbf{R}$ in (1) and (2)) with the randomized Kronecker projection matrix.

5. Learning Kronecker Projection

The randomized projection is simple, but it does not utilize the data distributions. Similar to the previous works [26, 10, 8, 37], we provide an efficient algorithm to optimize Kronecker projection parameters. We first introduce the optimization problem in binary embedding (Section 5.1), and quantization (Section 5.2), and then show that both can be formulated as solving orthogonal procrustes problem for each element matrix. We term such a problem the Kronecker procrustes, and provide our solution in Section 5.3. We assume that the training data is given as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \in \mathbb{R}^{d \times N}$. Our analysis of Section 5.1 to Section 5.3 is based on the assumption that $k = d$. Section 5.4 extends the solution to $k \neq d$ cases.

5.1. Optimized Binary Embedding

We follow [10, 8] to minimize the binarization loss for binary embedding. The optimization problem can be written as,

$$\arg\min_{\mathbf{B},\mathbf{R}} \| \mathbf{B} - \mathbf{R}\mathbf{X} \|_F^2, \quad \text{s.t.} \quad \mathbf{R}\mathbf{R}^T = \mathbf{I}, \qquad (3)$$

where binary matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N] \in \{-1, 1\}^{d \times N}$, and $\mathbf{b}_i$ is the binary code of $\mathbf{x}_i$, *i.e.* $\mathbf{b}_i = \text{sign}(\mathbf{R}\mathbf{x}_i)$. Different from [10, 8], we impose Kronecker structure on $\mathbf{R}$. Gong *et al*. [10] propose to find a local solution of (3) by alternating minimization. When $\mathbf{R}$ is fixed, $\mathbf{B}$ is computed by a straightforward binarization by definition. When $\mathbf{B}$ is fixed, and $k = d$ (we will discuss $k < d$ case in Section 5.4), $\mathbf{R}$ is found by the orthogonal procrustes problem:

$$\arg\min_{\mathbf{R}} ||\mathbf{B} - \mathbf{R}\mathbf{X}||_F^2, \quad \text{s.t.} \quad \mathbf{R}^T\mathbf{R} = \mathbf{I}.$$

5.2. Optimized Quantization

For quantization, we consider the Cartesian K-Means (ck-means) [26] method which is the state-of-the-art. Note that the Product Quantization (PQ) [16] is a special case of ck-means where the orthogonal projection is randomized instead of optimized.

For ck-means, the input sample $\mathbf{x}$ is split into m subspaces, $\mathbf{x} = [\mathbf{x}^{(1)}; \mathbf{x}^{(2)}; ...; \mathbf{x}^{(m)}]$, and each subspace is quantized to h sub-centers. Here, we only consider the case when all the sub-center sets have the same fixed cardinality. Our method can be easily generalized in a way similar to [26] for varying cardinalities.

Let $\mathbf{p} = [\mathbf{p}^{(1)}; \mathbf{p}^{(2)}; ...; \mathbf{p}^{(m)}]$, where $\mathbf{p}^{(j)} \in \{0, 1\}^h$, $\| \mathbf{p}^{(j)} \|_1 = 1$. In other words, $\mathbf{p}^{(j)}$ is an indicator of which sub-center $\mathbf{x}^{(j)}$ is closest to. Let $\mathbf{C}^{(j)} \in \mathbb{R}^{d \times h}$ be the j-th sub-center matrix and $\mathbf{C} \in \mathbb{R}^{d \times mh}$ be a center matrix

which is formed by the concatenation (diagonal-wise) of all the sub-centers:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}^{(1)} & & \\ & \ddots & \\ & & \mathbf{C}^{(m)} \end{bmatrix}.$$

In ck-means, the center matrix $\mathbf{C}$ is parameterized by an orthogonal matrix $\mathbf{R} \in \mathbb{R}^{d\times d}$ and a block diagonal matrix $\mathbf{D} \in \mathbb{R}^{d\times mh}$. The optimization problem of ck-means can be written as,

$$\arg\min_{\mathbf{R},\mathbf{P},\mathbf{D}} \parallel \mathbf{X} - \mathbf{RDP} \parallel_F^2, \quad \text{s.t.} \quad \mathbf{R}^T\mathbf{R} = \mathbf{I}.$$

We impose the Kronecker structure on the orthogonal matrix $\mathbf{R}$ with a similar alternating procedure. When $\mathbf{R}$ is fixed, updating $\mathbf{D}$ and $\mathbf{P}$ is equivalent to vector quantization in each subspace with k-means. This is efficient because the number of centers is usually small since the number of clusters for each subspace is always set to a small number ($h = 256$ in [16, 26]). Updating $\mathbf{R}$ with fixed $\mathbf{D}$ and $\mathbf{P}$ is also an orthogonal procrustes problem.

$$\arg\min_{\mathbf{R}} ||\mathbf{X} - \mathbf{RDP}||_F^2, \quad \text{s.t.} \quad \mathbf{R}^T\mathbf{R} = \mathbf{I}.$$

5.3. Kronecker Procrustes

For both binary embedding and quantization, we need to solve an orthogonal procrustes problem with Kronecker structure, which we call Kronecker procrustes:

$$\begin{aligned} &\arg\min_{\mathbf{R}} \parallel \mathbf{RX} - \mathbf{B} \parallel_F^2, \\ &\text{s.t. } \mathbf{R} = \mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_M,\ \mathbf{A}_j^T\mathbf{A}_j = \mathbf{I},\ j = 1, \ldots, M. \end{aligned} \tag{4}$$

The above optimization is non-convex and quite challenging. We note that there exists a closed-form solution for the unstructured orthogonal procrustes problem, which requires computing SVD of $\mathbf{XB}^T$ and takes $\mathcal{O}(d^3)$ time. For Kronecker procrustes, there is no closed-form solution. We develop an efficient iterative method to update each element matrix sequentially to find a local solution. We start by rewriting $\parallel \mathbf{RX} - \mathbf{B} \parallel_F^2$ as,

$$\begin{aligned} &\parallel (\otimes_{j=1}^M \mathbf{A}_j)\mathbf{X} - \mathbf{B} \parallel_F^2 \\ =& \operatorname{tr}(((\otimes_{j=1}^M \mathbf{A}_j)\mathbf{X} - \mathbf{B})^T((\otimes_{j=1}^M \mathbf{A}_j)\mathbf{X} - \mathbf{B})) \\ =& \parallel \mathbf{X} \parallel_F^2 - 2\operatorname{tr}((\otimes_{j=1}^M \mathbf{A}_j)\mathbf{XB}^T) + \parallel \mathbf{B} \parallel_F^2. \end{aligned} \tag{5}$$

The second equality holds because Kronecker product preserves orthogonality. Thus, we need to maximize $\operatorname{tr}((\otimes_{j=1}^M \mathbf{A}_j)\mathbf{XB}^T)$. Using a property of trace, it can be expressed as,

$$\operatorname{tr}(\mathbf{B}^T(\otimes_{j=1}^M \mathbf{A}_j)\mathbf{X}) = \sum_{i=1}^N \mathbf{b}_i^T(\otimes_{j=1}^M \mathbf{A}_j)\mathbf{x}_i,$$

where $\mathbf{b}_i$ and $\mathbf{x}_i$ are the i-th column of matrix $\mathbf{B}$ and matrix $\mathbf{X}$ respectively. We solve this problem by updating one element matrix at a time, while keeping all others fixed. Without loss of generality, consider updating $\mathbf{A}_j$:

$$\begin{aligned} &\arg\min_{\mathbf{A}_j} \sum_{i=1}^N \mathbf{b}_i^T(\mathbf{A}_{pre} \otimes \mathbf{A}_j \otimes \mathbf{A}_{next})\mathbf{x}_i \\ &\text{s.t. } \mathbf{A}_j^T\mathbf{A}_j = \mathbf{I}, \end{aligned} \tag{6}$$

where $\mathbf{A}_{pre} = 1 \otimes (\otimes_{i=1}^{j-1}\mathbf{A}_i)$, and $\mathbf{A}_{next} = (\otimes_{i=j+1}^{M}\mathbf{A}_i) \otimes 1$. Let the dimension of $\mathbf{A}_{pre}$, $\mathbf{A}_{next}$ and $\mathbf{A}_j$ be $k_{pre} \times d_{pre}$, $k_{next} \times d_{next}$ and $k_j \times d_j$, respectively. Obviously, $d_{pre}d_jd_{next} = d$ and $k_{pre}k_jk_{next} = k$.

According to a property of Kronecker product, the objective function of $\mathbf{A}_j$ in (6) can be written as,

$$\sum_{i=1}^N \mathbf{b}_i^T \operatorname{vec}\big((\mathbf{A}_j \otimes \mathbf{A}_{next})\operatorname{mat}(\mathbf{x}_i, d_jd_{next}, d_{pre})\mathbf{A}_{pre}^T\big). \tag{7}$$

Let $\mathbf{G}_i = \operatorname{mat}(\mathbf{x}_i, d_jd_{next}, d_{pre})\mathbf{A}_{pre}^T$, and $\mathbf{F}_i = \operatorname{mat}(\mathbf{b}_i, k_jk_{next}, k_{pre})$. Then, (7) can be written as,

$$\sum_{i=1}^N \operatorname{tr}(\mathbf{F}_i^T(\mathbf{A}_j \otimes \mathbf{A}_{next})\mathbf{G}_i). \tag{8}$$

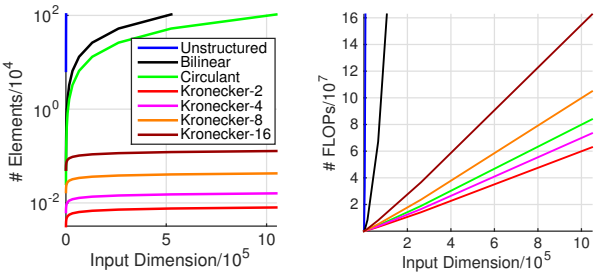
The above is the same as the bilinear optimization problem [8], which can be solved via polar decomposition, requiring SVD on a matrix of size $d_j \times k_j$, with computational complexity $\min(\mathcal{O}(d_j^2k_j), \mathcal{O}(k_j^2d_j))$.

When updating one element matrix, the computational cost comes from three different sources: **S1**. Calculating Kronecker projection of data with the fixed element matrices. **S2**. Calculating the product of projected data and codes. **S3**. Performing SVD to get the optimal element matrix. When the element matrices are large, the optimization bottleneck is SVD. When the element matrices are small, say 2×2, performing SVD can be seen as roughly a constant time operation. The main computational cost then comes from **S1** ($\mathcal{O}(Nd\log d)$) and **S2** ($\mathcal{O}(Nd)$). Since there are a total of $\log_{d_e} d$ element matrices, the computational complexity of the whole optimization is $\mathcal{O}(Nd\log^2 d)$.

In the optimization procedure, we use randomized Kronecker projection as initialization. In practice, we find that, for both binary embedding (Section 5.1), and quantization (Section 5.2), the objective decreases fast based on the proposed algorithm. Similar to [8, 37], a satisfactory solution can be found within a few iterations.

5.4. Learning with $k \neq d$

We have presented our algorithm in the case of $k = d$. The projection matrix $\mathbf{R}$ with $k \neq d$ can be formed by the Kronecker product of non-square row/column orthogonal element matrices. It is easy to show that Kronecker product also preserves the row/column orthogonality. When $k > d$, the orthogonal procrustes optimization problem can



(a) Storage (b) Time

Figure 1. Computational cost (measured by FLOPs count) and space cost for different types of projections. Kronecker-d_e means Kronecker projection formed by element matrices of order d_e.

be solved similarly to the $k = d$ case [30]. When $k < d$, $\mathbf{R}^T\mathbf{R} \neq \mathbf{I}$. Hence, the second equality in (5) does not hold. $\| \mathbf{RX} - \mathbf{B} \|_F^2$ becomes

$$\mathrm{tr}(\mathbf{X}^T\mathbf{R}^T\mathbf{RX}) - 2\,\mathrm{tr}(\mathbf{RXB}^T) + \| \mathbf{B} \|_F^2 .$$

We follow [9] and relax the problem by assuming that $\mathrm{tr}(\mathbf{X}^T\mathbf{R}^T\mathbf{RX})$ is independent of $\mathbf{R}$, same as in the $k \geq d$ case.

6. Experiments

6.1. Datasets and Methods

We evaluate our proposed Kronecker projection in approximate nearest neighbor search experiments on three high-dimensional datasets:

ImageNet-16384 contains 100k images sampled from ImageNet with each image represented by a 16,384 dimensional VLAD feature vector [17].

ImageNet-32768 is constructed the same way except each image represented by a 32,768 dimensional VLAD feature vector.

Flickr-16384 contains 100K images sampled from noisy internet image collection, represented by 16,384 dimensional VLAD feature vectors.

Following [8, 26, 37], we use 9,500 samples in training and 500 samples as queries for all three datasets. The features are ℓ_2 normalized.

We use the following baseline methods in the experiments: LSH [27], ITQ [10], BBE (bilinear binary embedding) [8], and CBE (circulant binary embedding) [37]. We call the proposed Kronecker projection based binary embedding *Kronecker Binary Embedding* (KBE), and KBE-d_e represents KBE using element matrices of order d_e. We use the suffix "-opt" and "-rand-orth" to denote the optimized and randomized versions of each projection type. Following [8], BBE-rand does not impose orthogonality. For quantization methods, we use PQ [16] and ckmeans [26] as baselines. KPQ is PQ with randomized Kronecker projection (with the element matrices of order 2). Kck-mean is ckmeans with optimized Kronecker projection (with the element matrices of order 2). In the approximate nearest neighbor retrieval experiments, for each query, we use its 10 nearest neighbors based on the ℓ_2 distance as the ground truth, and use recall as the evaluation metric. All the results are averaged over 10 runs.

6.2. Computational and Space Costs

Table 2 summarizes the required computations and space for different types of projections[2]. Table 2 also shows the real-world execution time (ms) and space cost (MB). Based on our implementation, the real-world costs approximately match the theoretical estimations. Figure 1 further shows the required computations and memory as a function of d. Kronecker projection provides significant speedup and space saving compared to the other methods. In addition, one advantage of Kronecker projection is its flexibility in trading off the model complexity with the computation time. In experiments, we used Kronecker projections with element matrices of the order 2 and 4 for binary embedding and order 2 for quantization, which gave satisfactory performance[3].

6.3. Approximate Nearest Neighbor Search

Low-dimensional data. We first test the proposed methods on a low-dimensional dataset ImageNet-256, which is constructed by randomly sampling 256 dimensions of ImageNet-16384. Such a study is required as many popular methods such as ITQ and ck-means are not practical for very high-dimensional data. The results are shown in Figure 2. Note that the quantization methods outperform binary embedding methods, but they come with the extra cost of center-based distance computation, which is more expensive than computing the Hamming distance [26].

Among all the quantization methods, ck-means outperforms PQ since it has an optimized orthogonal matrix. Replacing the orthogonal matrix by randomized Kronecker projection (KPQ) or optimized Kronecker projection (Kck-means) leads to very competitive performance.

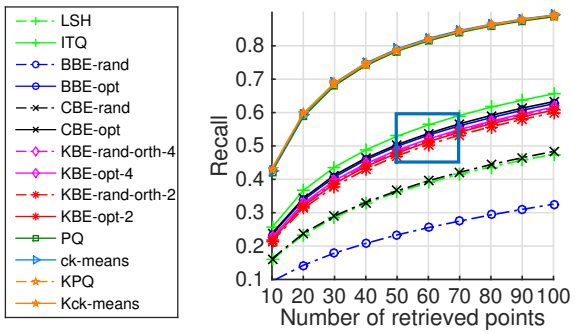
Among all the binary embedding methods, ITQ outperforms all the others, since it uses an optimized unstructured orthogonal matrix. BBE-opt, CBE-opt, KBE-2 and KBE-4 give similar performance, and they outperform LSH, CBE-rand, and BBE-rand. A zoomed-in view of ITQ, BBE-opt,

[2] Kronecker-d_e represents Kronecker projection with element matrices of order d_e. The computational and space costs (as a function of d) is computed based on the assumption that d is large, and $k = d$.

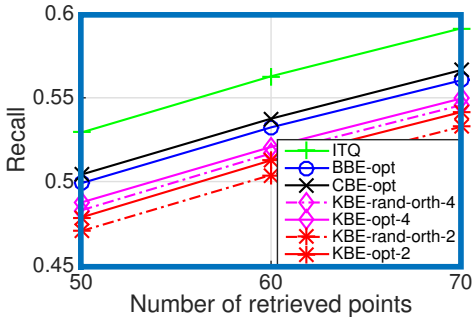
[3] The use of a few non-square matrices is sometimes required to match the input and output dimensions correctly. In the following experiments, for KBE-2, we use 2×2 and 2×4 element matrices. For KBE-4, we use 4×4, 4×8 and 2×2 element matrices.

	Unstructured		Bilinear		Circulant		Kronecker-2	
	Time	Space	Time	Space	Time	Space	Time	Space
d	$2d^2$	d^2	$4d^{1.5}$	$2d$	$4dlog_2d$	d	$3dlog_2d$	$4log_2d$
2^8	2.3e-1	2.5e-1	3.1e-2	2.0e-3	1.6e-2	9.8e-4	**1.1e-2**	**1.2e-4**
2^{10}	3.7	4.0	2.3e-1	7.8e-3	7.4e-2	3.9e-3	**5.6e-2**	**1.5e-4**
2^{12}	6.0e1	6.4e1	1.9	3.1e-2	3.8e-1	1.6e-2	**3.0e-1**	**1.8e-4**
2^{14}	9.5e2	1.0e3	1.5e1	1.2e-1	1.6	6.3e-2	**1.2**	**2.1e-4**
2^{16}	1.5e4	1.6e4	1.3e2	5.0e-1	8.4	2.5e-1	**6.4**	**2.4e-4**

Table 2. Execution time (ms) and space cost (single precision) (MB) of different types of projections. The result is based on a C implementation and a single core on a 2.6GHz Intel CPU.



(a) Retrieval performance.



(b) Zoomed-in view of Figure 2(a).

Figure 2. Retrieval performance on a low-dimensional dataset ImageNet-256.

CBE-opt and KBE is given in Figure 2(b). It shows that the performance of KBE can be improved by using more parameters (ITQ *aka.* KBE-256 > BBE *aka.* KBE-16 > KBE-4 > KBE-2). Yet, with only 48 and 32 parameters (KBE-4, KBE-2), we already have very competitive performance compared to ITQ (65,536 parameters) and BBE (512 parameters). The proposed optimization algorithm further improves the recall (KBE-opt-4 > KBE-rand-orth-4, KBE-opt-2 > KBE-rand-orth-2).

High-dimensional data. Figure 4 shows the retrieval performance with fixed number of bits on three high-dimensional datasets. One may notice that we did not compare with ITQ, ITQ with randomized unstructured orthogonal projection, ck-means, and ck-means with randomized unstructured orthogonal projection. The reason is that both the optimization and the generation of a randomized unstructured orthogonal matrix are prohibitively expensive for high-dimensional data (detailed in Section 4 and Section 5). Instead we can only do ck-means without rotation (which is exactly PQ), and ITQ without orthogonal constraint and optimization (which is exactly LSH).

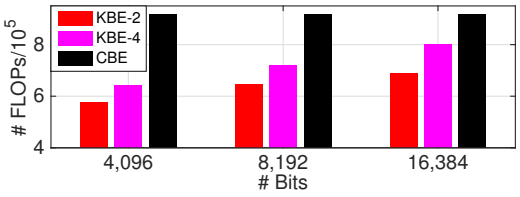
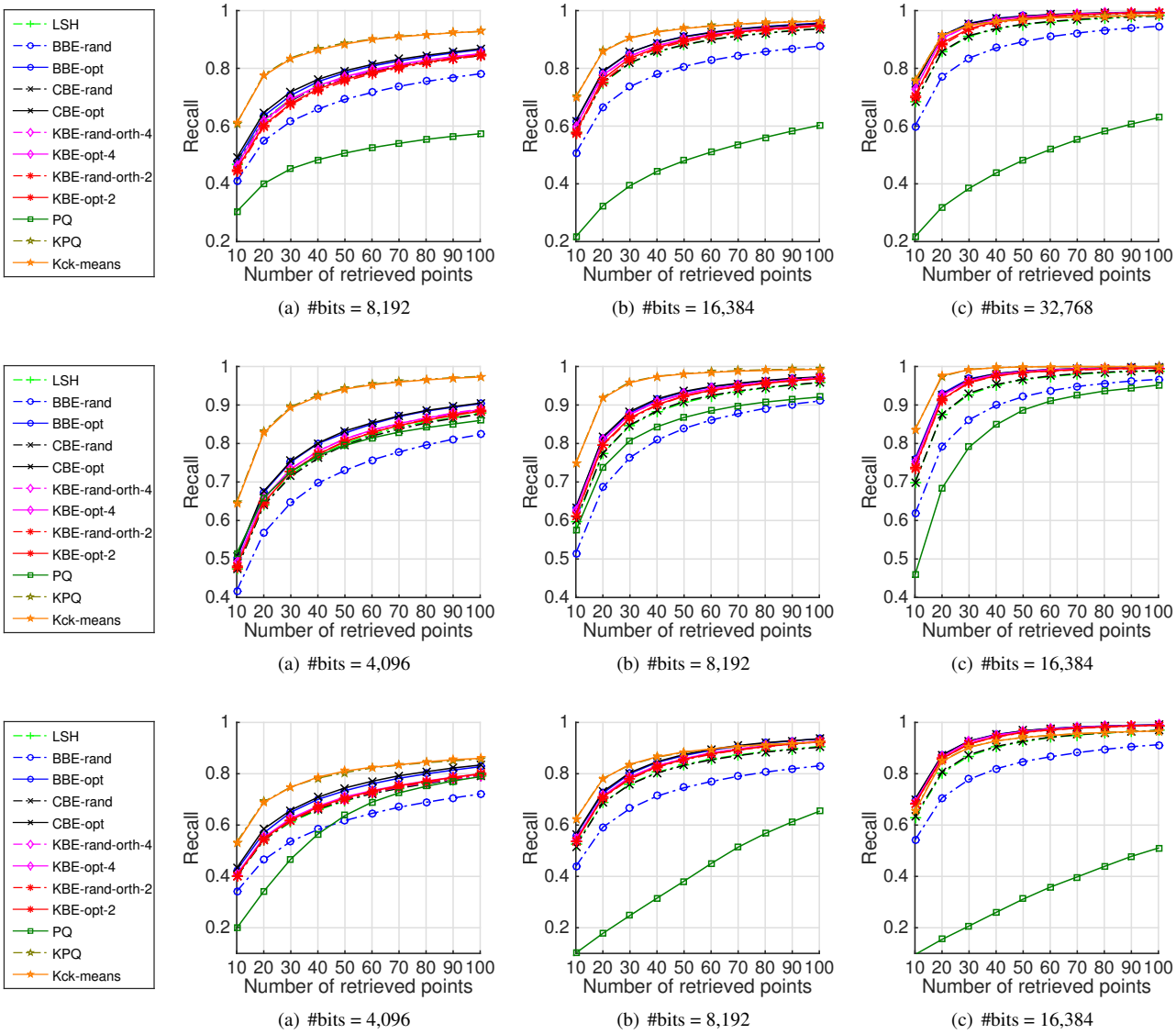


Figure 3. Number of FLOPs for projecting $d = 16,384$ dimensional data to create different number of bits.

One interesting finding is that in many cases PQ does not work well in such high-dimensional settings (also shown in [8]). The reason could be that the selection of the subspace is critical in PQ [17, 16]. Therefore [17] suggests using PCA followed by unstructured randomized orthogonal projection as pre-processing. However, such operations are impractical for very high-dimensional data. The proposed Kronecker projection makes orthogonal projections possible for high-dimensional data: both KPQ and Kck-means achieve very impressive performance.

For binary embedding, the results of KBE-opt-2, KBE-opt-4, KBE-rand-orth-2 and KBE-rand-orth-4 are competitive to CBE, which is the state-of-the-art. Beside the competitive performance, the proposed method has the advantage of reduced space and computational cost. This is most obvious in the cases when $k < d$. Due to the nature of the method, the space and computational costs of CBE for $k < d$ are identical to $k = d$ [37]. On the contrary, both the space and computational costs of KBE can be reduced. Figure 3 compares the computational cost of CBE and KBE with different structures and different number of bits.



(a) #bits = 8,192 (b) #bits = 16,384 (c) #bits = 32,768

(a) #bits = 4,096 (b) #bits = 8,192 (c) #bits = 16,384

(a) #bits = 4,096 (b) #bits = 8,192 (c) #bits = 16,384

Figure 4. Retrieval performance with fixed number of bits on ImageNet-32768 (first row), ImageNet-16384 (second row), and Flickr-16384 (third row). "#bits" is the number of bits used by each method.

For all settings, we found that the randomized Kronecker projection already gives competitive performance (this is especially true for high-dimensional data). It means that the orthogonal property of the projection plays an important role for guaranteeing the retrieval performance of ANN. And the proposed optimization algorithm can further improve the recall.

7. Conclusion

We proposed a special structured matrix to speed up orthogonal linear projections. The proposed method has $\mathcal{O}(d \log d)$ computational complexity and $\mathcal{O}(\log d)$ space complexity, dramatically lower than that of unstructured projection ($\mathcal{O}(d^2)$). The method is also very flexible in trading off the number of parameters and the computational cost. We successfully applied the Kronecker projection to binary embedding and quantization tasks for large-scale approximate image retrieval. We also found that the orthogonal property of the projection is important to the performance of ANN techniques. Comprehensive experiments showed that, with the same number of bits, the proposed method can achieve competitive or even better performance with much lower space and computational cost. The implementation of the method is available at `https://github.com/spongezhang/Kronecker_Projection.git`.

References

[1] N. Ailon and B. Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

[2] W. U. Bajwa, J. D. Haupt, G. M. Raz, S. J. Wright, and R. D. Nowak. Toeplitz-structured compressed sensing matrices. In *IEEE/SP 14th Workshop on Statistical Signal Processing*, 2007.

[3] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *ACM Symposium on Theory of Computing*, 2002.

[4] Y. Cheng, F. X. Yu, R. Feris, S. Kumar, A. Choudhary, and S.-F. Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *ICCV*, 2015.

[5] A. Dasgupta, R. Kumar, and T. Sarlós. Fast locality-sensitive hashing. In *SIGKDD*, 2011.

[6] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR*, 2013.

[7] A. Gersho and R. M. Gray. *Vector quantization and signal compression*. Springer Science & Business Media, 1992.

[8] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, 2013.

[9] Y. Gong, S. Kumar, V. Verma, and S. Lazebnik. Angular quantization-based binary codes for fast similarity search. In *Advances in Neural Information Processing Systems*, pages 1196–1204, 2012.

[10] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[11] J. Haupt, W. U. Bajwa, G. Raz, and R. Nowak. Toeplitz compressed sensing matrices with applications to sparse channel estimation. *IEEE Transactions on Information Theory*, 2010.

[12] A. Hinrichs and J. Vybíral. Johnson-lindenstrauss lemma for circulant matrices. *Random Structures & Algorithms*, 39(3):391–398, 2011.

[13] R. Hunger. *Floating point operations in matrix-vector calculus*. Munich University of Technology, Inst. for Circuit Theory and Signal Processing, 2005.

[14] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Trans. Info. Theory*, 2013.

[15] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[16] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 2011.

[17] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[18] H. Jégou, T. Furon, and J. Fuchs. Anti-sparse coding for approximate nearest neighbor search. In *ICASSP*, 2012.

[19] J. Ji, S. Yan, J. Li, G. Gao, Q. Tian, and B. Zhang. Batch-orthogonal locality-sensitive hashing for angular similarity. *IEEE TPAMI*, 2014.

[20] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

[21] Q. Le, T. Sarlós, and A. Smola. Fastfood–approximating kernel expansions in loglinear time. In *ICML*, 2013.

[22] P. Li, A. Shrivastava, J. Moore, and A. C. Konig. Hashing algorithms for large-scale learning. In *NIPS*, 2011.

[23] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, 2012.

[24] M. Norouzi, D. Fleet, and R. Salakhutdinov. Hamming distance metric learning. In *NIPS*, 2012.

[25] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[26] M. Norouzi and D. J. Fleet. Cartesian k-means. In *CVPR*. IEEE, 2013.

[27] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, 2009.

[28] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*. 2012.

[29] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

[30] P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[31] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *IEEE TPAMI*, 2012.

[32] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

[33] C. F. Van Loan. The ubiquitous kronecker product. *Journal of Computational and Applied Mathematics*, 123(1):85–100, 2000.

[34] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE TPAMI*, 2012.

[35] J. Wang, J. Wang, J. Song, X.-S. Xu, H. T. Shen, and S. Li. Optimized cartesian k-means. *IEEE TKDD*, 27(1):180–192, Jan 2015.

[36] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[37] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *ICML*, 2014.

[38] F. X. Yu, S. Kumar, H. Rowley, and S.-F. Chang. Compact nonlinear maps and circulant extensions. *arXiv preprint arXiv:1503.03893*, 2015.