Composite Quantization for Approximate Nearest Neighbor Search

Ting Zhang[*] ZTING@MAIL.USTC.EDU.CN
University of Science and Technology of China, Hefei, P.R. China

Chao Du[*] DUCHAO0726@GMAIL.COM
Tsinghua University, Beijing, P.R. China

Jingdong Wang JINGDW@MICROSOFT.COM
Microsoft Research, Beijing, P.R. China

Abstract

This paper presents a novel compact coding approach, composite quantization, for approximate nearest neighbor search. The idea is to use the composition of several elements selected from the dictionaries to accurately approximate a vector and to represent the vector by a short code composed of the indices of the selected elements. To efficiently compute the approximate distance of a query to a database vector using the short code, we introduce an extra constraint, constant inter-dictionary-element-product, resulting in that approximating the distance only using the distance of the query to each selected element is enough for nearest neighbor search. Experimental comparison with state-of-the-art algorithms over several benchmark datasets demonstrates the efficacy of the proposed approach.

1. Introduction

Nearest neighbor (NN) search has been a fundamental research topic in machine learning, computer vision, and information retrieval (Shakhnarovich et al., 2006). The goal of NN search, given a query $\mathbf{q}$, is to find a vector $\mathrm{NN}(\mathbf{q})$ whose distance to the query is the smallest from the N d-dimensional database vectors.

The straightforward solution, linear scan, is to compute the distances to all the database vectors whose time cost is $O(Nd)$ and is not practical for large scale high-dimensional cases. Multi-dimensional indexing methods, such as the k-d tree (Friedman et al., 1977), have been developed to speed up exact search. However, for high-dimensional cases it turns out that such approaches are not much more efficient (or even less efficient) than linear scan.

Therefore, there have been a lot of interests in algorithms that perform approximate nearest neighbor (ANN) search. Many data structures and algorithms are developed to eliminate the number of distance computations. For example, the algorithm with a k-d tree is used to find ANNs by limiting the search time. Exploiting random multiple k-d trees and priority search (Silpa-Anan & Hartley, 2008) result in good performance in terms of accuracy and efficiency. FLANN (Muja & Lowe, 2009) finds a good configuration of random k-d trees and hierarchial k-means trees to optimize the performance. Other tree structures, such as the cover tree (Beygelzimer et al., 2006) and the trinary-projection tree (Jia et al., 2010; Wang et al., 2014), and so on, have also been designed. Neighborhood graph (Arya & Mount, 1993; Wang & Li, 2012; Wang et al., 2013b;c) is also adopted for ANN search.

The algorithms that convert the database vectors into short codes have been attracting a lot of attention recently as their storage cost is small, making the in-memory search feasible, and the distance computation cost is also reduced. The seminal work on locality sensitive hashing (LSH) (Gionis et al., 1999) uses random projections to compute Hamming embeddings, and is later followed by a lot of extensions, such as Mahalanobis distance (Jain et al., 2008), kernelization (Kulis & Grauman, 2009), complementary hashing (Xu et al., 2011), and so on. Subsequent efforts have been devoted to learn good hashing functions, such as learnt binary reconstruction (Kulis & Darrells, 2009), semantic hashing (Salakhutdinov & Hinton, 2009), shift kernel hashing (Raginsky & Lazebnik, 2009), spectral hash-

[*]This work was done when Ting Zhang and Chao Du were interns at Microsoft Research, Beijing, P.R. China.

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

ing (Weiss et al., 2008), graph-based hashing (Liu et al., 2011), iterative quantization (Gong & Lazebnik, 2011), isotropic hashing (Kong & Li, 2012), minimal loss hashing (Norouzi & Fleet, 2011), order preserving hashing (Wang et al., 2013a), and so on. The Hamming embedding algorithms are only able to produce a few distinct distances resulting in limited ability and flexibility of distance approximation.

Product quantization (PQ), a data compression technique in signal processing, is recently used for efficient nearest neighbor search (Jégou et al., 2011a). The idea is to decompose the space into a Cartesian product of low-dimensional subspaces and to quantize each subspace separately. A vector is then represented by a short code composed of its subspace quantization indices. The distance between a query and a database vector is approximated by using the query and the short code corresponding to the database vector. Specifically, it first computes the distance of the query to the quantized center of the database vector in each subspace, and then sums up the distances together, where the distance lookup table is used for fast distance evaluation. The advantage over Hamming embedding methods is that the number of possible distances is significantly higher and hence the distance approximation is more accurate. It is shown in (Jégou et al., 2011a) that PQ achieves a much better search performance with comparable efficiency than hamming embedding algorithms. Cartesian k-means (Norouzi & Fleet, 2013) improves product quantization by finding better subspace partitioning and achieves a better search performance.

In this paper, we propose a novel approach, composite quantization, to convert vectors to compact codes. The idea is to approximate a vector using the composition of several elements selected from several dictionaries and to represent this vector by a short code composed of the indices of the selected elements. The advantage is that the vector approximation, and accordingly the distance approximation of a query to the database vector, is more accurate, yielding more accurate nearest neighbor search. To efficiently evaluate the distance between a query and the short code representing the database vector, we introduce an extra constraint, called constant inter-dictionary-element-product, i.e., the summation of the inner products of all pairs of elements that are used to approximate the vector but from different dictionaries is constant. As a result, the approximate distance can be calculated from the distance of the query to each selected element, which is efficient by using a few distance table lookups. We present an alternative optimization algorithm to simultaneously find the dictionaries and the compact codes. In addition, we show that production quantization (Jégou et al., 2011a) and Cartesian k-means (Norouzi & Fleet, 2013) can be regarded as constrained versions of our approach. Experimental results over several datasets demonstrate the proposed approach achieves state-of-the-art performances.

2. Formulation

Given a query vector $\mathbf{q} \in \mathbb{R}^d$ and a set of N d-dimensional vectors $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the nearest neighbor search problem aims to find the item $\mathrm{NN}(\mathbf{q})$ from $\mathcal{X}$ such that its distance to the query vector is minimum. In this paper, we study the approximate nearest neighbor search problem under the Euclidean distance and propose a compact coding approach, called composite quantization.

The idea is to approximate a vector $\mathbf{x}$ by the composition of several (M) elements $\{\mathbf{c}_{1k_1}, \mathbf{c}_{2k_2}, \ldots, \mathbf{c}_{Mk_M}\}$, each of which is selected from a dictionary with K elements, $\mathcal{C}_m = \{\mathbf{c}_{m1}, \ldots, \mathbf{c}_{mK}\}$, e.g., $\mathbf{c}_{1k_1}$ is the k_1th element in dictionary $\mathcal{C}_1$, and to represent a vector by a short code composed of the indices of the selected elements, resulting in a compact representation of length $M \log K$ with each dictionary element coded by $\log K$ bits.

Let $\bar{\mathbf{x}} = \sum_{m=1}^{M} \mathbf{c}_{mk_m}$ be the approximation of the vector $\mathbf{x}$. The accuracy of nearest neighbor search relies on the degree of the distance approximation, i.e., how small is the difference between the distance of a vector $\mathbf{q}$ to the vector $\mathbf{x}$ and the distance to the approximation $\bar{\mathbf{x}}$. According to the triangle inequality, $|\|\mathbf{q}-\mathbf{x}\|_2 - \|\mathbf{q}-\bar{\mathbf{x}}\|_2| \leqslant \|\mathbf{x}-\bar{\mathbf{x}}\|_2$, minimizing the distance approximation error can be transformed to minimizing the vector approximation error, which is formulated as follows,

$$\min_{\mathbf{c}_{mk_m^n} \in \mathcal{C}_m} \sum_{n=1}^{N} \|\mathbf{x}_n - \sum_{m=1}^{M} \mathbf{c}_{mk_m^n}\|_2^2, \quad (1)$$

where $\mathbf{c}_{mk_m^n}$ is the element selected from the dictionary $\mathcal{C}_m$ for the nth database vector $\mathbf{x}_n$.

Given the approximation $\bar{\mathbf{x}}$, the distance of a query $\mathbf{q}$ to the approximation $\bar{\mathbf{x}}$ is $\|\mathbf{q}-\bar{\mathbf{x}}\|_2 = \|\mathbf{q} - \sum_{m=1}^{M} \mathbf{c}_{mk_m}\|_2$. It is time-consuming to reconstruct the approximate vector $\bar{\mathbf{x}}$ (taking $O(Md)$) and then compute the distance between $\mathbf{q}$ and $\bar{\mathbf{x}}$ (taking $O(d)$). Considering the expansion,

$$\|\mathbf{q} - \sum_{m=1}^{M} \mathbf{c}_{mk_m}\|_2^2 = \sum_{m=1}^{M} \|\mathbf{q} - \mathbf{c}_{mk_m}\|_2^2 - (M-1)\|\mathbf{q}\|_2^2 + \sum_{i=1}^{M} \sum_{j=1, j\neq i}^{M} \mathbf{c}_{ik_i}^T \mathbf{c}_{jk_j}, \quad (2)$$

we can see that, given the query $\mathbf{q}$, the second term $-(M-1)\|\mathbf{q}\|_2^2$ in the right-hand side is constant for all the database vectors and hence unnecessary to compute for nearest neighbor search. The first term $\sum_{m=1}^{M} \|\mathbf{q} - \mathbf{c}_{mk_m}\|_2^2$ is the summation of the distances of the query to the selected dictionary elements and can be efficiently computed using a few (M) distance table lookups, where the distance table is precomputed and stores the distances of the query to the dictionary elements. Similarly, we can build a table storing

the inner products between dictionary elements and compute the third term using $O(M^2)$ distance table lookups. This results in the distance computation cost is changed from $O(Md)$ to $O(M^2)$, which is still a little large.

To further reduce the computation cost, we introduce an extra constraint, letting the third term be a constant, i.e., $\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\mathbf{c}_{ik_i}^T\mathbf{c}_{jk_j}=\epsilon$, called constant inter-dictionary-element-product. As a result, we only need to compute the first term for nearest neighbor search, and the computation cost is reduced to $O(M)$.

In summary, the optimization problem is formulated as

$$\begin{aligned}\min_{\{\mathbf{C}_m\},\{\mathbf{b}_n\},\epsilon}\quad & \sum_{n=1}^{N}\|\mathbf{x}_n-[\mathbf{C}_1\mathbf{C}_2\cdots\mathbf{C}_M]\mathbf{b}_n\|_2^2 \quad (3)\\ \text{s.t.}\quad & \sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}\mathbf{b}_{ni}^T\mathbf{C}_i^T\mathbf{C}_j\mathbf{b}_{nj}=\epsilon\\ & \mathbf{b}_n=[\mathbf{b}_{n1}^T\mathbf{b}_{n2}^T\cdots\mathbf{b}_{nM}^T]^T\\ & \mathbf{b}_{nm}\in\{0,1\}^K,\ \|\mathbf{b}_{nm}\|_1=1\\ & n=1,2,\cdots,N, m=1,2,\cdots,M.\end{aligned}$$

Here, $\mathbf{C}_m$ is a matrix of size $d\times K$, and each column corresponds to an element of the mth dictionary $\mathcal{C}_m$. $\mathbf{b}_n$ is the composition vector, and its subvector $\mathbf{b}_{nm}$ is an indicator vector with only one entry being 1 and all others being 0, showing which element is selected from the mth dictionary for composite quantization.

3. Optimization

The problem formulated in 3 is a mixed-binary-integer program, which consists of three groups of unknown variables: dictionaries $\{\mathbf{C}_m\}$, composition vectors $\{\mathbf{b}_n\}$, and ϵ. In addition to the binary-integer constraint over $\{\mathbf{b}_n\}$, there are quadratic equality constraints over $\{\mathbf{C}_m\}$, $\sum_{i\neq j}^{M}\mathbf{b}_{ni}^T\mathbf{C}_i^T\mathbf{C}_j\mathbf{b}_{nj}=\epsilon$. We propose to adopt the quadratic penalty method and add a penalty function that measures the violation of the quadratic equality constraints into the objective function,

$$\begin{aligned}\phi(\{\mathbf{C}_m\},\{\mathbf{b}_n\},\epsilon)&=\sum_{n=1}^{N}\|\mathbf{x}_n-\mathbf{C}\mathbf{b}_n\|_2^2\\ &+\mu\sum_{n=1}^{N}(\sum_{i\neq j}^{M}\mathbf{b}_{ni}^T\mathbf{C}_i^T\mathbf{C}_j\mathbf{b}_{nj}-\epsilon)^2, \quad (4)\end{aligned}$$

where μ is the penalty parameter, $\mathbf{C}=[\mathbf{C}_1\mathbf{C}_2\cdots\mathbf{C}_M]$ and $\sum_{i\neq j}^{M}=\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}$.

3.1. Algorithm

We use the alternative optimization technique to iteratively solve the problem. Each iteration alternatively updates $\{\mathbf{b}_n\}$, ϵ and $\{\mathbf{C}_m\}$. The details are given below.

Update $\{\mathbf{b}_n\}$. It can be easily seen that $\mathbf{b}_n$, the composition indicator of a vector $\mathbf{x}_n$, given $\{\mathbf{C}_m\}$ and ϵ fixed, is independent to $\{\mathbf{b}_t\}_{t\neq n}$ for all the other vectors. Then the optimization problem is decomposed to N subproblems,

$$\begin{aligned}&\phi_n(\mathbf{b}_n)\\ &=\|\mathbf{x}_n-\mathbf{C}\mathbf{b}_n\|_2^2+\mu(\sum_{i\neq j}^{M}\mathbf{b}_{ni}^T\mathbf{C}_i^T\mathbf{C}_j\mathbf{b}_{nj}-\epsilon)^2, \quad (5)\end{aligned}$$

where there are three constraints: $\mathbf{b}_n$ is a binary vector, $\|\mathbf{b}_{nm}\|_1=1$, and $\mathbf{b}_n=[\mathbf{b}_{n1}^T\mathbf{b}_{n2}^T\cdots\mathbf{b}_{nM}^T]^T$. Generally, this optimization problem is NP-hard. We notice that the problem is essentially a high-order MRF (Markov random field) problem. We again use the alternative optimization technique like the iterated conditional modes algorithm that is widely used to solve MRFs, and solve the M subvectors $\{\mathbf{b}_{nm}\}$ alternatively. Given $\{\mathbf{b}_{nl}\}_{l\neq m}$ fixed, we update $\mathbf{b}_{nm}$ by exhaustively checking all the elements in the dictionary $\mathcal{C}_m$, finding the element such that the objective value is minimized, and accordingly setting the corresponding entry of $\mathbf{b}_{nm}$ to be 1 and all the others to be 0.

Update ϵ. We can see that the objective function is a quadratic function with respect to ϵ. Given $\mathbf{C}$ and $\{\mathbf{b}_n\}$ fixed, it is easy to get the optimal solution to ϵ,

$$\epsilon=\frac{1}{N}\sum_{n=1}^{N}\sum_{i\neq j}^{M}\mathbf{b}_{ni}^T\mathbf{C}_i^T\mathbf{C}_j\mathbf{b}_{nj}. \quad (6)$$

Update $\{\mathbf{C}_m\}$. Fixing $\{\mathbf{b}_n\}$ and ϵ, the problem is an unconstrained nonlinear optimization problem with respect to $\mathbf{C}$. There are many algorithms for such a problem. We choose the quasi-Newton algorithm and specifically the L-BFGS algorithm, the limited-memory version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. It only needs a few vectors to represent the approximation of the Hessian matrix instead of storing the full Hessian matrix as done in the BFGS algorithm. We adopt the publicly available implementation of L-BFGS[†]. The partial-derivative with respect to $\mathbf{C}_m$, the input to the L-BFGS solver, is computed as follows,

$$\begin{aligned}&\frac{\partial}{\partial\mathbf{C}_m}\phi(\{\mathbf{C}_m\},\{\mathbf{b}_n\},\epsilon)\\ &=\sum_{n=1}^{N}[2(\sum_{l=1}^{M}\mathbf{C}_l\mathbf{b}_{nl}-\mathbf{x}_n)\mathbf{b}_{nm}^T+\\ &4\mu(\sum_{i\neq j}^{M}\mathbf{b}_{ni}^T\mathbf{C}_i^T\mathbf{C}_j\mathbf{b}_{nj}-\epsilon)(\sum_{l=1,l\neq m}^{M}\mathbf{C}_l\mathbf{b}_{nl})\mathbf{b}_{nm}^T]. \quad (7)\end{aligned}$$

3.2. Implementation details

The proposed algorithm is warm-started by using the solution of a relatively easy problem, which is formed by dropping the constant inter-dictionary-element-product con-

[†]http://www.ece.northwestern.edu/~nocedal/lbfgs.html

straint, i.e., the problem in Equation 1, for the initialization. The easy problem is also solved by alterative optimization, iteratively updating $\mathbf{C}$ and $\{\mathbf{b}_n\}$. The scheme of updating $\{\mathbf{b}_n\}$ is almost the same to the above scheme with dropping ϵ-related terms. Updating $\mathbf{C}$ is relatively easy. Given $\{\mathbf{b}_n\}$ fixed, the objective function is quadratic with respect to $\mathbf{C}$, and there is a closed-form solution: $\mathbf{C} = \mathbf{X}\mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$, where $\mathbf{X}$ is a matrix with each column corresponding to a database vector, and $\mathbf{B}$ is a matrix composed of the composition vectors of all the database vectors, $\mathbf{B} = [\mathbf{b}_1\ \mathbf{b}_2\ \cdots\ \mathbf{b}_n]$.

The penalty method usually needs to solve a series of unconstrained problems by increasing the penalty parameter μ into infinity to make the constraint completely satisfied. In our case, we find that the inter-dictionary-element-product is not necessarily exactly constant and the search performance is not affected if the deviation of the inter-dictionary-element-product from a constant is relatively small compared with the distortion error. Therefore, our algorithm instead relaxes this constraint and selects the parameter μ via validation. The validation dataset is a subset of the database (selecting a subset is only for validation efficiency, and it is fine that the validation set is a subset of the learning set as the validation criterion is not the objective function value but the search performance). The best parameter μ is chosen so that the average search performance by regarding the validation vectors as queries and finding $\{5, 10, 15, \cdots, 100\}$ nearest neighbors from all the database vectors is the best.

3.3. Analysis

We present the time complexity of each iteration. At the beginning of each iteration, we first compute inner product tables, $\{\mathbf{c}_{ir}^T\mathbf{c}_{js} | i \neq j, r, s = 1, 2, \cdots, K\}$, between the dictionary elements, taking $O(M^2K^2d)$, so that computing $\mathbf{b}_{ni}^T\mathbf{C}_i^T\mathbf{C}_j^T\mathbf{b}_{nj}$ can be completed by one table lookup. The time complexities of the three updates are given as follows. (1) It takes $O(MKdT_b)$ with T_b being the number of iterations ($= 3$ in our implementation achieving satisfactory results) to update $\mathbf{b}_n$, i.e., optimize the objective function in 5, and thus the time complexity of updating $\{\mathbf{b}_n\}$ is $O(NMKdT_b)$. (2) It takes $O(NM^2)$ to update ϵ, which can be easily seen from Equation 6. (3) The main cost of updating $\{\mathbf{C}_m\}$ lies in computing the partial derivatives and the objective function value that are necessary in L-BFGS. For clarity, we drop the complexity terms that are independent of N and can be neglected for a large N. Then, the time complexity for updating $\{\mathbf{C}_m\}$ is $O(NMdT_lT_c)$ with T_c being the number of iterations ($= 10$ in our implementation) and T_l (set to 5 in our implementation) being the number of line searches in L-BFGS.

The objective function value at each iteration in the algorithm always weakly decreases. It can be validated that the objective function value is lower-bounded (not smaller than 0). The two points indicate the convergence of our algorithm. The theoretic analysis of the rate of convergence is not easy, while the empirical results show that the algorithm takes a few iterations to converge. Figure 1 shows an example convergence curve.

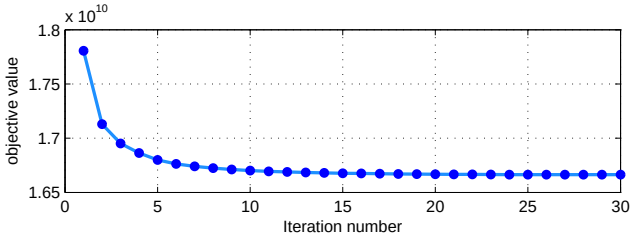

Figure 1. Convergence curve of our algorithm. The vertical axis represents the objective function value of Equation 4 and the horizontal axis corresponds to the number of iterations. The objective function value at the initialization is about 2.3×10^{12} and not shown for clarity. The curve is obtained from the result over a representative dataset $1M$ SIFT with 64 bits.

4. Discussions

Relation to k-means and sparse coding. Composite quantization, when only one dictionary is used (i.e., $M = 1$), is degraded to the k-means approach. Compared with k-means, composite quantization is able to produce a larger number of quantized centers (K^M) using a few dictionary elements (KM), resulting in that the composite quantizer can be indexed in memory for large scale quantized centers.

Composite quantization is also related to coding with block sparsity (Yuan & Lin, 2006), in which the coefficients are divided into several blocks and the sparsity constraints are imposed in each block separately. Composite quantization can be regarded as a sparse coding approach, where the coefficients that can be only valued by 0 or 1 are divided into M groups, for each group the non-sparsity degree is 1, and an extra constraint, constant inter-dictionary-element-product, is considered.

Connection with product quantization and Cartesian k-means. Product quantization (Jégou et al., 2011a) decomposes the space into M low dimensional subspaces and quantizes each subspace separately. A vector $\mathbf{x}$ is decomposed into M subvectors, $\{\mathbf{x}^1, \cdots, \mathbf{x}^M\}$. Let the quantization dictionaries over the M subspaces be $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_M$ with $\mathcal{C}_m$ being a set of centers $\{\mathbf{c}_{m1}, \cdots, \mathbf{c}_{mK}\}$. A vector $\mathbf{x}$ is represented by the concatenation of M centers, $[\mathbf{c}_{1k_1^*}^T\mathbf{c}_{2k_2^*}^T \cdots \mathbf{c}_{mk_m^*}^T \cdots \mathbf{c}_{Mk_M^*}^T]^T$, where $\mathbf{c}_{mk_m^*}$ is the one nearest to $\mathbf{x}^m$ in the mth quantization dictionary.

Rewrite each center $\mathbf{c}_{mk}$ as a d-dimensional vector $\tilde{\mathbf{c}}_{mk}$ so

Table 1. The description of the datasets.

	BASE SET	QUERY SET	DIM
MNIST	60,000	10,000	784
LABELME22K	20,019	2,000	512
$1M$ SIFT	1,000,000	10,000	128
$1M$ GIST	1,000,000	1,000	960
$1B$ SIFT	1,000,000,000	10,000	128

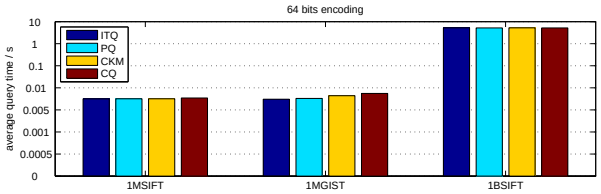

Figure 2. Average query time on $1M$ SIFT, $1M$ GIST and $1B$ SIFT.

that $\tilde{\mathbf{c}}_{mk} = [\mathbf{0}^T \cdots (\mathbf{c}_{mk})^T \mathbf{0}^T]^T$, i.e., all entries are zero except that the subvector corresponding to the mth subspace is equal to $\mathbf{c}_{mk}$. The approximation of a vector $\mathbf{x}$ using the concatenation $\mathbf{x} = [\mathbf{c}_{1k_1^*}^T \ \mathbf{c}_{2k_2^* c}^T \cdots \mathbf{c}_{Mk_M^*}^T]^T$ is then equivalent to the composition $\mathbf{x} = \sum_{m=1}^M \tilde{\mathbf{c}}_{mk_m^*}$. Similarly, it can also be shown that there is a same equivalence in Cartesian k-means (Norouzi & Fleet, 2013).

The above analysis indicates that both product quantization and Cartesian k-means can be regarded as a constrained version of composition quantization, with the orthogonal dictionary constraint: $\mathbf{C}_i^T \mathbf{C}_j = \mathbf{0}$, $i \neq j$, which guarantees that the constant inter-dictionary-element-product constraint in our approach holds. In addition, unlike product quantization and Cartesian k-means in which each dictionary (subspace) is formed by d/M dimensions, our approach when $\epsilon = 0$ is able to automatically decide how many dimensions belong to one dictionary.

An upper-bound minimization view. Let us introduce several notations: $d(\mathbf{q}, \mathbf{x}) = \|\mathbf{q} - \mathbf{x}\|_2$; $d(\mathbf{q}, \bar{\mathbf{x}}) = \|\mathbf{q} - \bar{\mathbf{x}}\|_2 = \|\mathbf{q} - \sum_{m=1}^M \mathbf{c}_{mk_m}\|_2$; $\tilde{d}(\mathbf{q}, \bar{\mathbf{x}}) = (\sum_{m=1}^M \|\mathbf{q} - \mathbf{c}_{mk_m}\|_2^2)^{1/2}$; $\hat{d}(\mathbf{q}, \mathbf{x}) = (\|\mathbf{q} - \mathbf{x}\|_2^2 + (M-1)\|\mathbf{q}\|_2^2)^{1/2}$ ($\hat{d}(\mathbf{q}, \bar{\mathbf{x}}) = (\|\mathbf{q} - \bar{\mathbf{x}}\|_2^2 + (M-1)\|\mathbf{q}\|_2^2)^{1/2}$) which is the square root of the summation of the square of the true (approximate) Euclidean distance and a query-dependent term $(M-1)\|\mathbf{q}\|_2^2$ that is a constant for the search with a specific query. Let $\delta = \sum_{i \neq j} \mathbf{c}_{ik_i}^T \mathbf{c}_{jk_j}$. By definition, we have $\hat{d}(\mathbf{q}, \bar{\mathbf{x}}) = (\tilde{d}^2(\mathbf{q}, \bar{\mathbf{x}}) + \delta)^{1/2}$.

Our approach uses $\tilde{d}(\mathbf{q}, \bar{\mathbf{x}})$ as the distance approximation and essentially aims to use it to approximate $\hat{d}(\mathbf{q}, \mathbf{x})$. Ideally, if $\tilde{d}(\mathbf{q}, \bar{\mathbf{x}}) = \hat{d}(\mathbf{q}, \mathbf{x})$, the search accuracy would be 100%. In general, the absolute difference $|\tilde{d}(\mathbf{q}, \bar{\mathbf{x}}) - \hat{d}(\mathbf{q}, \mathbf{x})|$ is expected to be small to guarantee high search accuracy. We have the following theorem:

Theorem 1. *The reconstruction error of the distances is upper-bounded:* $|\tilde{d}(\mathbf{q}, \bar{\mathbf{x}}) - \hat{d}(\mathbf{q}, \mathbf{x})| \leq \|\mathbf{x} - \bar{\mathbf{x}}\|_2 + |\delta|^{1/2}$.

This theorem suggests a solution to minimizing the distance reconstruction error by minimizing the upper-bound: $\min \|\mathbf{x} - \bar{\mathbf{x}}\|_2 + |\delta|^{1/2}$. With the assumption $\delta = \epsilon$ for $\forall \mathbf{x} \in \mathcal{X}$, this minimization problem is transformed to a constrained optimization problem: $\min \|\mathbf{x} - \bar{\mathbf{x}}\|_2$ subject to $\delta = \epsilon$. Accumulating the distortion errors over all the database vectors yields formulation 3.

5. Experiments

We perform the ANN experiments on five datasets: MNIST[‡] (LeCun et al., 2001), $784D$ grayscale images of handwritten digits; LabelMe22K (Russell et al., 2008) a corpus of images expressed as $512D$ GIST descriptors; $1M$ SIFT (Jégou et al., 2011a), consisting of $1M$ $128D$ SIFT features as base vectors, $100K$ learning vectors and $10K$ queries; $1M$ GIST (Jégou et al., 2011a), containing $1M$ $960D$ global GIST descriptors as base vectors, $500K$ learning vectors and $1K$ queries; and $1B$ SIFT (Jégou et al., 2011b), composed of $1B$ SIFT features as base vectors, $100M$ learning vectors and $10K$ queries. The detailed description is presented in Table 1.

We compare our approach, composite quantization (CQ), with several state-of-the-art methods: product quantization (PQ) (Jégou et al., 2011a), Cartesian k-means (CKM) (Norouzi & Fleet, 2013). It is already shown that PQ and CKM achieve better search accuracy than hashing algorithms with the same code length and comparable search efficiency. Thus, we report one result from a representative hashing algorithm, iterative quantization (ITQ) (Gong & Lazebnik, 2011). All the results were obtained with the implementations generously provided by their respective authors. Following (Jégou et al., 2011a), we use the structured ordering for $1M$ GIST and the natural ordering for $1M$ SIFT and $1B$ SIFT to get the best performance for PQ. We choose $K = 256$ as the dictionary size which is an attractive choice because the resulting distance lookup tables are small and each subindex fits into one byte (Jégou et al., 2011a; Norouzi & Fleet, 2013).

To find ANNs, all the algorithms perform linear scan search using asymmetric distance: to compare a query with a database vector, PQ, CKM and CQ conduct a few distance table lookups and additions, and ITQ uses asymmetric hamming distance for better search accuracy proposed in (Gordo & Perronnin, 2011). ITQ, PQ, CKM and CQ takes the same time for linear scan. Their costs of computing the distance lookup table are slightly different, and

[‡]http://yann.lecun.com/exdb/mnist/

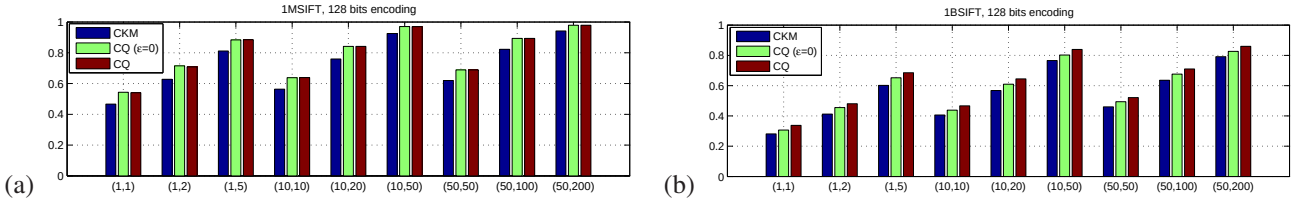

Figure 3. Illustrating the effect of ϵ on (a) $1M$ SIFT and (b) $1B$ SIFT. (T, R) means recall@R when searching for T nearest neighbors.

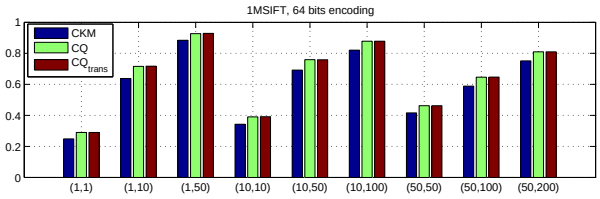

Figure 4. Illustrating the effect of translation. (T, R) means recall@R when searching for T nearest neighbors.

can be negligible as they are very low compared with the cost for linear scan. For instance, the cost of computing the distance lookup table in our approach only takes around 2% of the cost of linear scan on $1M$ SIFT. Figure 2 shows the average query times on the $1M$ SIFT, $1M$ GIST and $1B$ SIFT datasets, from which one can observe that the costs are similar for the four methods.

The search quality is measured using recall@R. For each query, we retrieve its R nearest items and compute the ratio of R to T, i.e., the fraction of T ground-truth nearest neighbors are found in the retrieved R items. The average recall score over all the queries is used as the measure. The ground-truth nearest neighbors are computed over the original features using linear scan. In the experiments, we report the performance with T being 1, 10, and 50. The observed conclusions remain valid for other T.

5.1. Empirical analysis

The effect of ϵ. The proposed approach learns the variable ϵ, inter-dictionary-element-product. Alternatively, one can simply set it to be zero, $\epsilon = 0$, indicating that the dictionaries are mutually orthogonal like splitting the spaces into subspaces as done in product quantization and Cartesian k-means. The average distortion error in the case of learning ϵ potentially can be smaller than that in the case of letting $\epsilon = 0$ as learning ϵ is more flexible, and thus the search performance with learnt ϵ can be better. The experimental results over the $1M$ SIFT and $1B$ SIFT datasets under the two schemes, shown in Figure 3, validate this point: the performances over $1M$ SIFT are similar and the performance when ϵ is not limited to be zero over $1B$ SIFT is much better.

The effect of translating the vector. One potential extension of our approach is to introduce an offset, denoted $\mathbf{t}$, to translate $\mathbf{x}$. Introducing the offset does not increase the storage cost as it is a global parameter. The objective function with such an offset is as follows: $\min_{\mathbf{C},\mathbf{t},\mathbf{b}_1,\cdots,\mathbf{b}_N} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{t} - \mathbf{C}\mathbf{b}_n\|_2^2$. Our experiments indicate that this introduction does not influence the performance too much. An example result on $1M$ SIFT with 64 bits is shown in Figure 4. The reason might be that the contribution of the offset to the quantization distortion reduction is relatively small compared with that from the composition of selected dictionary elements.

5.2. Results

Figure 5 shows the comparison on MNIST and LabelMe22K. One can see that the vector approximation algorithms, our approach (CQ), CKM, and PQ outperform ITQ. It is as expected because the information loss in Hamming embedding used in ITQ is much larger. PQ also performs not so good because it does not well exploit the data information for subspace partitioning. Our approach performs the best, which validates the aforementioned analysis. The improvement seems a little small, but it is actually significant as the datasets are relatively small and the search is relatively easy.

Figure 6 shows the results of large scale datasets: $1M$ SIFT and $1M$ GIST using codes of 64 bits and 128 bits for searching 1, 10, and 50 nearest neighbors. It can be seen that the gain obtained by our approach (CQ) is significant for $1M$ SIFT. For example, the recall@10 with $T = 1$ for Cartesian k-means and ours are 63.83% and 71.59% on 64 bits. In other words, the improvement is 7.76% and the relative improvement reaches 12%. The reason of the relatively small improvement on $1M$ GIST over Cartesian k-means might be that Cartesian k-means already achieves very large improvement over product quantization and the improvement space is relatively small.

Figure 7 shows the performance for a very large dataset, $1B$ SIFT. Similar to (Norouzi & Fleet, 2013), we use the first $1M$ learning vectors for efficient training. It can be seen that our approach gets the best performance and the improvement is consistently significant. For example, the recall@100 from our approach on the $1B$ base set with 64

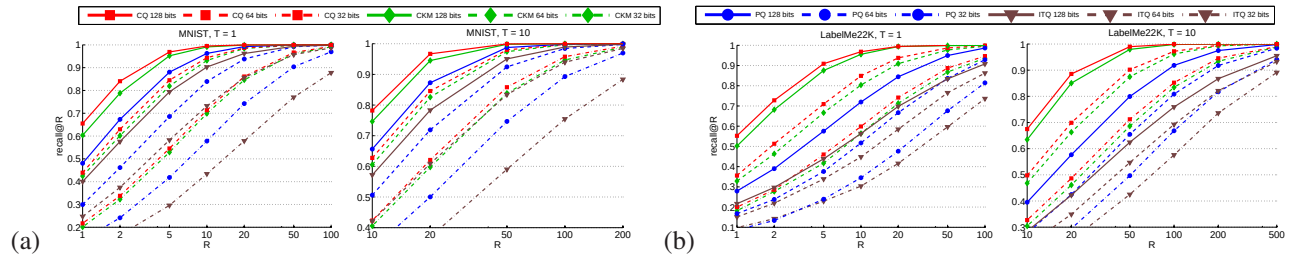

Figure 5. The performance for different algorithms on (a) MNIST and (b) LabelMe22K for searching various numbers of ground truth nearest neighbors ($T = 1, 10$).

Table 2. The performance of object retrieval over the holiday data set in terms of MAP.

	#BIT	ITQ	PQ	CKM	CQ
FISHER	32	0.4132	0.5040	0.5373	0.5500
	64	0.5334	0.5476	0.5775	0.6221
	128	0.5883	0.5794	0.5978	0.6339
VLAD	32	0.4378	0.5132	0.5453	0.5777
	64	0.5371	0.5740	0.5974	0.6320
	128	0.6074	0.5861	0.6092	0.6442

Table 3. The performance of object retrieval over the UKBench data set in terms of scores.

	#BITS	ITQ	PQ	CKM	CQ
FISHER	32	2.1155	2.2031	2.6060	2.7401
	64	2.6320	2.6181	2.8943	3.0093
	128	2.8808	2.8507	3.0387	3.1539
VLAD	32	2.0935	2.2140	2.6312	2.7455
	64	2.6174	2.6290	2.9246	3.0459
	128	2.8946	2.8776	3.0688	3.1854

bits for $T = 1$ is 70.12% while from CKM it is 64.57%. Besides the performance over all the $1B$ database vectors, we also show the performance on a subset of $1B$ base vectors, the first $10M$ database vectors. As we can see, the performance on $1B$ vectors is worse than that on $10M$ vectors, which is reasonable as searching over a larger dataset is more difficult. The notable observation is that the improvement of our approach over Cartesian k-means on the larger dataset, $1B$ database vectors, is much more significant than that on $10M$ database vectors.

Besides, we plot the curve in terms of recall vs. the code length, as depicted in Figure 8 on a representative dataset, $1M$ SIFT, to explicitly show how the improvement of our approach over other approaches changes under various code lengths. As expected, the performances of all the algorithms improve along with the increase of the code length. Notably, the improvement of our approach over the other methods, gets more significant as the code length is larger. In contrast, the improvement of CKM over PQ is reduced when the code becomes longer. This shows that the superiority of our approach in the longer code is stronger.

5.3. Application to object retrieval

We report the results of the applications to object retrieval. In object retrieval, images are represented as an aggregation of local descriptors, often thousands of dimension. We evaluate the performances over the 4096-dimensional Fisher vectors (Perronnin & Dance, 2007) and the 4096-dimensional VLAD vectors (Jégou et al., 2010) extracted from the INRIA Holidays data set (Jégou et al., 2008) that

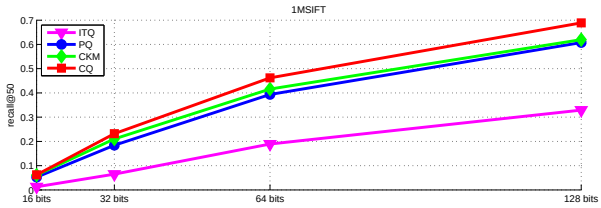

Figure 8. Illustrating the effect of the code length on $1M$ SIFT when searching for 50 nearest neighbors.

contains 500 query and 991 relevant images, and the UKBench data set (Nistér & Stewénius, 2006) that contains 2550 groups of 4 images each (totally 10200 images).

The search performances in terms of mean average precision (MAP) (Jégou et al., 2008) for the holiday data set and the score (Nistér & Stewénius, 2006) for the UKBench data set are shown in Table 2 and Table 3. It can be seen that our method performs the best, which is because our approach (CQ) produces better vector approximation.

6. Conclusion

In this paper, we present a compact coding approach, composite quantization, to approximate nearest neighbor search. The superior search accuracy stems from that it exploits the composition of dictionary elements to approximate a vector, yielding smaller distortion errors. The search efficiency is guaranteed by making the inter-dictionary-element-product constant and discarding its computation. Empirical results on different datasets suggest that the pro-

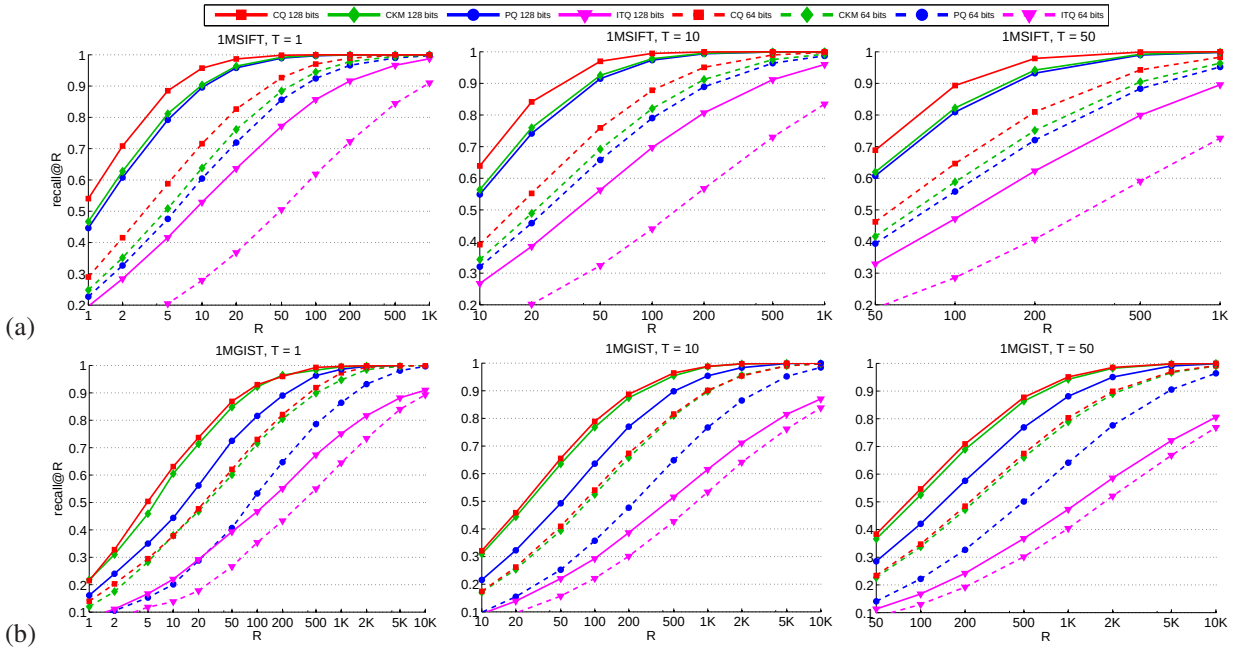

Figure 6. The performance for different algorithms on (a) $1M$ SIFT and (b) $1M$ GIST ($T = 1, 10, 50$).

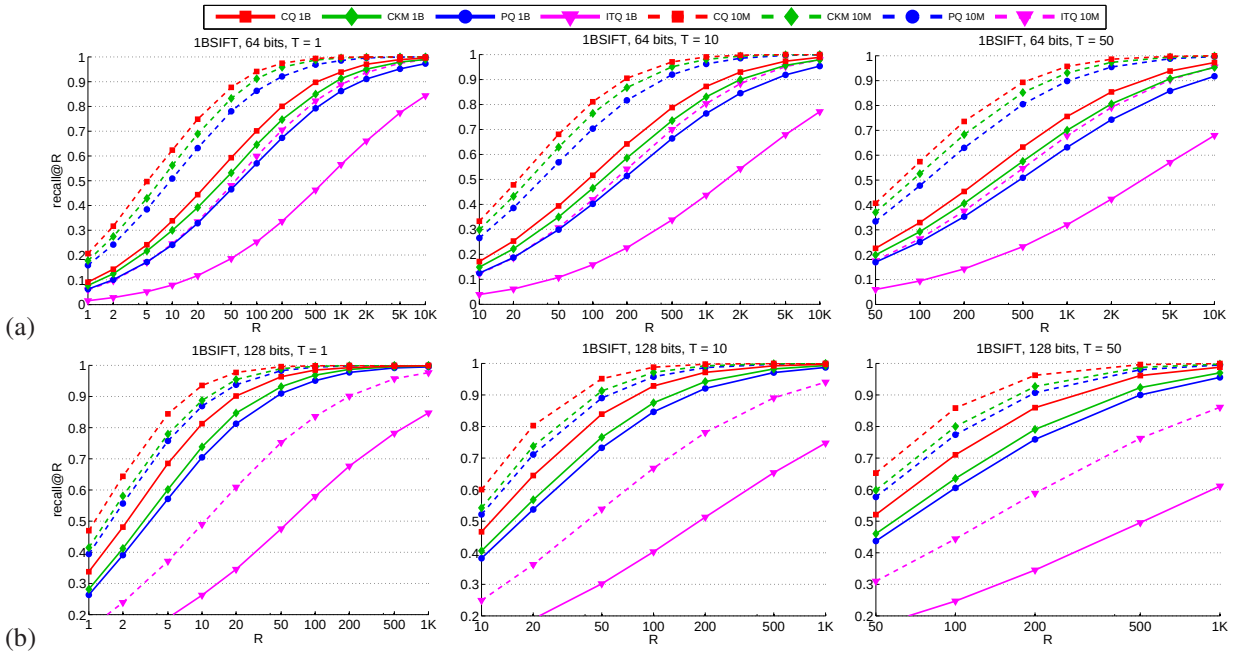

Figure 7. The performance for different algorithms on $1B$ SIFT with (a) 64 bits and (b) 128 bits ($T = 1, 10, 50$).

posed approach outperforms existing methods.

Acknowledgements

This work was partially supported by the National Basic Research Program of China (973 Program) under Grant 2014CB347600.

References

Arya, S. and Mount, D. M. Approximate nearest neighbor queries in fixed dimensions. In *SODA*, pp. 271–280, 1993.

Beygelzimer, A., Kakade, S., and Langford, J. Cover trees for nearest neighbor. In *ICML*, pp. 97–104, 2006.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

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

Gong, Y. and Lazebnik, S. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pp. 817–824, 2011.

Gordo, A. and Perronnin, F. Asymmetric distances for binary embeddings. In *CVPR*, pp. 729–736, 2011.

Jain, P., Kulis, B., and Grauman, K. Fast image search for learned metrics. In *CVPR*, 2008.

Jégou, H., Douze, M., and Schmid, C. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV (1)*, pp. 304–317, 2008.

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

Jégou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011a.

Jégou, H., Tavenard, R., Douze, M., and Amsaleg, L. Searching in one billion vectors: Re-rank with source coding. In *ICASSP*, pp. 861–864, 2011b.

Jia, Y., Wang, J., Zeng, G., Zha, H., and Hua, Xian-Sheng. Optimizing kd-trees for scalable visual descriptor indexing. In *CVPR*, pp. 3392–3399, 2010.

Kong, W. and Li, W. Isotropic hashing. In *NIPS*, pp. 1655–1663, 2012.

Kulis, B. and Darrells, T. Learning to hash with binary reconstructive embeddings. In *NIPS*, pp. 577–584, 2009.

Kulis, B. and Grauman, K. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pp. 306–351. IEEE Press, 2001.

Liu, W., Wang, J., Kumar, S., and Chang, S. Hashing with graphs. In *ICML*, pp. 1–8, 2011.

Muja, M. and Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP (1)*, pp. 331–340, 2009.

Nistér, D. and Stewénius, H. Scalable recognition with a vocabulary tree. In *CVPR (2)*, pp. 2161–2168, 2006.

Norouzi, M. and Fleet, D. J. Minimal loss hashing for compact binary codes. In *ICML*, pp. 353–360, 2011.

Norouzi, M. and Fleet, D. J. Cartesian k-means. In *CVPR*, pp. 3017–3024, 2013.

Perronnin, F. and Dance, C. R. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

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

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008.

Salakhutdinov, R. and Hinton, G. E. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.

Shakhnarovich, G., Darrell, T., and Indyk, P. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT press, 2006.

Silpa-Anan, C. and Hartley, R. Optimised kd-trees for fast image descriptor matching. In *CVPR*, 2008.

Wang, J. and Li, S. Query-driven iterated neighborhood graph search for large scale indexing. In *ACM Multimedia*, pp. 179–188, 2012.

Wang, J., Wang, J., Yu, N., and Li, S. Order preserving hashing for approximate nearest neighbor search. In *ACM Multimedia*, pp. 133–142, 2013a.

Wang, J., Wang, J., Zeng, G., Gan, R., Li, S., and Guo, B. Fast neighborhood graph search using cartesian concatenation. *CoRR*, abs/1312.3062, 2013b.

Wang, J., Wang, J., Zeng, G., Gan, R., Li, S., and Guo, B. Fast neighborhood graph search using cartesian concatenation. In *ICCV*, pp. 2128–2135, 2013c.

Wang, J., Wang, N., Jia, Y., Li, J., Zeng, G., Zha, H., and Hua, X.-S. Trinary-projection trees for approximate nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(2):388–403, 2014.

Weiss, Y., Torralba, A. B., and Fergus, R. Spectral hashing. In *NIPS*, pp. 1753–1760, 2008.

Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., and Yu, N. Complementary hashing for approximate nearest neighbor search. In *ICCV*, pp. 1631–1638, 2011.

Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.