Sparse Composite Quantization

Ting Zhang [1]* Guo-Jun Qi[2] Jinhui Tang[3] Jingdong Wang[4]
[1]University of Science and Technology of China, P.R. China [2]University of Central Florida, USA
[3]Nanjing University of Science and Technology, P.R. China [4]Microsoft Research, P.R. China

Abstract

The quantization techniques have shown competitive performance in approximate nearest neighbor search. The state-of-the-art algorithm, composite quantization, takes advantage of the compositionabity, i.e., the vector approximation accuracy, as opposed to product quantization and Cartesian k-means. However, we have observed that the runtime cost of computing the distance table in composite quantization, which is used as a lookup table for fast distance computation, becomes nonnegligible in real applications, e.g., reordering the candidates retrieved from the inverted index when handling very large scale databases. To address this problem, we develop a novel approach, called sparse composite quantization, which constructs sparse dictionaries. The benefit is that the distance evaluation between the query and the dictionary element (a sparse vector) is accelerated using the efficient sparse vector operation, and thus the cost of distance table computation is reduced a lot. Experiment results on large scale ANN retrieval tasks ($1M$ SIFTs and $1B$ SIFTs) and applications to object retrieval show that the proposed approach yields competitive performance: superior search accuracy to product quantization and Cartesian k-means with almost the same computing cost, and much faster ANN search than composite quantization with the same level of accuracy.

1. Introduction

Nearest neighbor (NN) search has wide applications in pattern classification, computer vision, and information retrieval, such as the K-NN classifier, similar image search, object instance search, and similar document search. Unfortunately, exact NN search is often impractical in the large-scale high-dimensional case because the computing cost is very high, causing unaffordable latency to return the search result. Thus, many research efforts have been devoted to approximate nearest neighbor (ANN) search.

*This work was done when Ting Zhang was an intern at Microsoft Research, Beijing, P.R. China.

This paper concerns the quantization algorithms, a category of compact coding approaches, for ANN search with high-dimensional data, which shows competitive search accuracy with tractable storage and search cost. Following product quantization [13] that divides the space into partitions and conducts k-means separately over each partition, the extensions with optimized space partitions, Cartesian k-means [22] and optimized product quantization [6], have been proposed. The recently proposed composite quantization approach [33] introduces a new framework generalizing those algorithms.

The acceleration obtained by these ANN algorithms stems from the ability of efficiently computing the distance between a query and a database vector by looking up a distance table in the online query stage before scanning the whole database. Previous approaches mainly focus on improving the accuracy of the distance approximation, while paying little attention to reducing the cost of the online distance table construction. The construction cost becomes a significant contribution to the ANN search cost in real applications, e.g., reordering the candidates retrieved from inverted index when handling very large databases, especially when we use composite quantization for higher search accuracy.

To handle this issue, we introduce a novel approach, sparse composite quantization, which generalizes the composite quantization approach by allowing the words in the dictionaries to be sparse. The key advantage of this approach is that the distance between the query and every dictionary word is evaluated very efficiently using sparse vector multiplication, which significantly reduces the search time. As a consequence, the proposed approach constructs the distance lookup table as fast as the most efficient algorithm, product quantization, while achieving a higher accuracy than product quantization and Cartesian k-means, and also a similar or even higher accuracy when compared to composite quantization. On the challenging search task with 1 billion SIFT vectors, the proposed approach outperforms the state-of-the-art algorithms in terms of both search efficiency and search accuracy. In particular, compared with composite quantization that achieves the highest recall performance, the proposed approach has successfully reduced the search time by at least 7% up to 36% in many cases, while the reduction of recall is limited within less than 3%.

2. Related work

The approximate nearest neighbor search algorithms are in general developed from two main aspects: (1) comparing the query with only a small subset of database vectors through special data structures, such as k-d trees [4], FLANN [20], and neighborhood graph [27], and (2) accelerating the distance computation through compact codes, such as hashing [8, 32, 19, 26], quantization [13, 22, 33]. The real applications, e.g., handling very large scale databases, often combine the two categories of solutions together to achieve satisfactory search quality in terms of search accuracy, search efficiency, and space cost. The proposed approach belongs to the second category comprehensively surveyed in [28]. It has been shown that the quantization algorithms [13, 22, 28] achieve better search quality than hashing algorithms with Hamming distance, even with optimized or asymmetric distances [9, 29]. In this section, we present a brief review of the quantization algorithms.

Hypercubic quantization, such as iterative quantization [8], isotropic hashing [17], harmonious hashing [31], angular quantization [7], can be regarded as a variant of scalar quantization by optimally rotating the data space and performing binary quantization along each dimension in the rotated space, with the quantization centers fixed at -1 and 1 (or equivalently 0 and 1). Such a way of fixing quantization centers puts a limit on the number of possible distances in the coding space, which also limits the accuracy of distance approximation even using optimized distances [9, 29]. Therefore, the overall search performance is not comparable to product quantization and Cartesian k-means.

Product quantization [13] divides the data space into (e.g., M) disjoint subspaces. Accordingly, each database vector is divided into M subvectors, and the whole database is also split into M sub-databases. A number of clusters are obtained by conducting k-means over each sub-database. Then a database vector is approximated by concatenating the nearest cluster center of each subspace, yielding a representation with a short code containing the indices of the nearest cluster centers. The computation of the distance between two vectors is accelerated by looking up a precomputed table.

Cartesian k-means [22] (or optimized product quantization [6]) improves the compositionabilty, i.e. vector approximation accuracy, by finding an optimal feature space rotation and then performing product quantization over the rotated space. Additive quantization [1, 5] further improves the compositionabilty by approximating a database vector using the summation of dictionary words selected from different dictionaries, whose idea is similar to structured vector quantization [10] (a.k.a., multi-stage vector quantization and residual quantization). It has been applied to data compression [1] and inner product similarity search [5], yet is not suitable for search with Euclidean distance due to the lack of the acceleration of distance computation.

The state-of-the-art approach, composite quantization [33], approximates a database vector using the summation of dictionary words, and introduces the orthogonality condition between dictionaries, resulting in keeping the efficiency of computing the approximate distance with slight sacrifice in compositionabilty (the vector approximation accuracy). Our approach goes in this direction, and addresses the high cost problem in computing the distance table.

There are other attempts to improve product quantization in the other ways, such as distance-encoded product quantization [11] and locally optimized product quantization [16], which can also be combined with our approach in the same way. The inverted multi-index [2] applies product quantization to build an inverted index for searching a very large scale database, with the ability of efficiently retrieving the candidates from a large number of inverted lists. Bilayer product quantization [3] improves the efficiency of distance computation within the inverted multi-index framework. We will also apply the proposed approach to inverted multi-index to show its effectiveness.

3. Formulation

Given a database containing N data vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ of dimension D, the proposed approach 1) learns M dictionaries $\{\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_M\}$, with each containing a few number of (assumed to be K without loss of generality) D-dimensional vectors (words) denoted by $\mathbf{C}_m = [\mathbf{c}_{m1}\ \mathbf{c}_{m2} \cdots \mathbf{c}_{mK}]$; 2) approximates each database vector using a composite vector $\bar{\mathbf{x}} = \sum_{m=1}^{M} \mathbf{c}_{mk_m}$, where $\mathbf{c}_{mk_m}$ is the k_mth vector in the mth dictionary $\mathbf{C}_m$; and 3) represents the database vector $\mathbf{x}$ using a short code of length $M \log K$, a sequence of indices $(k_1, k_2, \cdots, k_M)$.

The distance between a query vector $\mathbf{q}$ and a database vector $\mathbf{x}$ is approximated as below,

$$\|\mathbf{q} - \mathbf{x}\|_2^2 \approx \|\mathbf{q} - \bar{\mathbf{x}}\|_2^2 = \|\mathbf{q} - \sum_{m=1}^{M} \mathbf{c}_{mk_m}\|_2^2 \qquad (1)$$

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

Composite quantization [33] simplifies the approximate distance computation by introducing a constant constraint on inter-dictionary- element-product $\sum_{i=1}^{M} \sum_{j=1, j\neq i}^{M} \mathbf{c}_{ik_i}^\top \mathbf{c}_{jk_j} =$ constant. Then it is enough to compute the first term $\sum_{m=1}^{M} \|\mathbf{q} - \mathbf{c}_{mk_m}\|_2^2$ to search for the nearest neighbors, which furthermore is accelerated by looking up a distance table storing the distances between the query and the words of all the M dictionaries, and thus takes $O(M)$ in computing time.

The distance table computation consists of MK distance computations: $\{\|\mathbf{q} - \mathbf{c}_{mk}\|_2^2 \mid m = 1, 2, \cdots, M; k = 1, 2, \cdots, K\}$, which takes $O(MKD)$ time. Compared with the approximate distance computation cost over all N data vectors, $O(MN)$, the distance table cost is negligible if $N \gg KD$. Nonetheless, considering a real application, where we reorder the candidates retrieved from inverted index [2] for 128-dimensional SIFT features with a typical setting, $K = 256$ and $M = 8$, the number of retrieved candidates N (e.g., $= 100{,}000$) is not much greater than (about 3 times) KD ($\approx 32{,}000$) and such cost of distance table computation becomes significant and nonnegligible.

Our idea, to accelerate the distance table computation, is inspired by the property: the cost of computing the Euclidean distance between two vectors, $\|\mathbf{q} - \mathbf{c}\|_2^2$, can be accelerated if the vector $\mathbf{c}$ is very sparse and $\|\mathbf{q}\|_2^2$ is already known, hence the complexity in general becomes $O(\|\mathbf{c}\|_0)$, i.e., linear in the number of non-zero entries. Rather than enforcing each dictionary word being sparse (i.e., let each $\|\mathbf{c}_{mk}\|_0$ be $\frac{D}{M}$, into which we will show that product quantization can be cast), we impose a global sparsity constraint over all the dictionary words together, e.g., $\sum_{m=1}^{M}\sum_{k=1}^{K} \|\mathbf{c}_{mk}\|_0 \leqslant KD$.

To this end, the objective is to minimize the vector approximation error, with the constant inter-dictionary-element-product constraint and the sparsity regularization. We formally formulate the objective function as follows,

$$\min_{\{\mathbf{C}_m\},\{\mathbf{y}_n\},\xi} \quad \sum_{n=1}^{N} \|\mathbf{x}_n - [\mathbf{C}_1\mathbf{C}_2\cdots\mathbf{C}_M]\mathbf{y}_n\|_2^2 \tag{2}$$

$$\text{s. t.} \quad \sum_{m=1}^{M}\sum_{k=1}^{K} \|\mathbf{c}_{mk}\|_0 \leqslant S \tag{3}$$

$$\sum_{m=1}^{M}\sum_{m'=1, m'\neq m}^{M} \mathbf{y}_{nm}^\top \mathbf{C}_m^\top \mathbf{C}_{m'} \mathbf{y}_{nm'} = \xi \tag{4}$$

$$n = 1, 2, \cdots, N. \tag{5}$$

Here S is a positive constant indicating the sparsity degree. $\mathbf{y}_n = [\mathbf{y}_{n1}^\top \mathbf{y}_{n2}^\top \cdots \mathbf{y}_{nM}^\top]^\top$, and $\mathbf{y}_{nm}$ is a binary vector with only one entry valued as 1 and all others as 0. It is meant that the kth word of the dictionary $\mathbf{C}_m$ is selected to form the approximation of $\mathbf{x}_n$ if the kth entry y_{nmk} is equal to 1. $[\mathbf{C}_1\mathbf{C}_2\cdots\mathbf{C}_M]\mathbf{y}_n$ is equivalent to the definition of the approximation $\bar{\mathbf{x}} = \sum_{m=1}^{M} \mathbf{c}_{mk_m}$.

4. Optimization

The constrained problem 2 contains both continuous variables, ($\{\mathbf{C}_m\}$ and ξ) and binary variables ($\{\mathbf{y}_n\}$), and includes two constraints, the equality constraint 4 and the L_0 sparsity constraint 3, which make it uneasy to solve. We present a two-step approximate solution. The first step replaces the L_0 norm with the L_1 norm and transforms the problem to an unconstrained problem,

$$\phi(\{\mathbf{y}_n\}, \xi, \mathbf{C}) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{C}\mathbf{y}_n\|_2^2 + \lambda \sum_{m=1}^{M}\sum_{k=1}^{K} \|\mathbf{c}_{mk}\|_1 + \mu \sum_{n=1}^{N} \Big(\sum_{m'\neq m}^{M} \mathbf{y}_{nm}^\top \mathbf{C}_m^\top \mathbf{C}_{m'} \mathbf{y}_{nm'} - \xi\Big)^2, \tag{6}$$

where $\mathbf{C} = [\mathbf{C}_1\mathbf{C}_2\cdots\mathbf{C}_M]$ and $\sum_{m'\neq m}^{M} = \sum_{m=1}^{M}\sum_{m'=1,m'\neq m}^{M}$.

In the second step, we optimize another transformed problem by dropping off the sparsity term,

$$\phi(\{\mathbf{y}_n\}, \xi, \mathbf{C}) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{C}\mathbf{y}_n\|_2^2 + \mu \sum_{n=1}^{N} \Big(\sum_{m'\neq m}^{M} \mathbf{y}_{nm}^\top \mathbf{C}_m^\top \mathbf{C}_{m'} \mathbf{y}_{nm'} - \xi\Big)^2, \tag{7}$$

and introducing a new constraint: $c = 0, \forall c \in \mathcal{C}'$, where $\mathcal{C}'$ is a subset of $\mathcal{C} = \{c_{mkd} \mid m = 1, 2, \cdots, M; k = 1, 2, \cdots, K; d = 1, 2, \cdots, D\}$ of all the entries in the dictionary $\mathbf{C}$, such that the subset $\mathcal{C} \setminus \mathcal{C}'$ only contains those entries whose absolute values are the S largest from the optimal solution $\mathbf{C}^*$ to problem 6. Hence the new constraint ensures that there are at most S nonzero entries in $\mathbf{C}$, which satisfies the constraint 3. It can be shown that this construction of $\mathcal{C}'$ is equivalent to solving a constrained problem, $\min_{\mathbf{C}} \|\mathbf{C} - \mathbf{C}^*\|_F^2$ subject to $\sum_{m=1}^{M}\sum_{k=1}^{K} \|\mathbf{c}_{mk}\|_0 \leqslant S$.

We solve both the problems 6 and 7 using an alternative optimization algorithm in which each iteration alternatively updates $\{\mathbf{y}_n\}$, ξ, and $\mathbf{C}$. The whole pipeline is very similar to the one given in [33] and the difference lies in updating $\mathbf{C}$. The outline is given in Algorithm 1. The following first presents brief descriptions for updating $\{\mathbf{y}_n\}$ and ξ and then describes how to update $\mathbf{C}$ for the two problems 6 and 7, respectively.

Update $\{\mathbf{y}_n\}$. With $\mathbf{C}$ and ξ fixed, we can see that the optimization problem can be decomposed into N sub-problems,

$$\phi_n(\mathbf{y}_n) = \|\mathbf{x}_n - \mathbf{C}\mathbf{y}_n\|_2^2 + \mu\Big(\sum_{m'\neq m}^{M} \mathbf{y}_{nm}^\top \mathbf{C}_m^\top \mathbf{C}_{m'} \mathbf{y}_{nm'} - \xi\Big)^2,$$

each solving M sub-vectors $\{\mathbf{y}_{nm}\}_{m=1}^{M}$ in cycle: with $\{\mathbf{y}_{nm'}\}_{m'=1,m'\neq m}^{M}$ fixed, we exhaustively check all the elements in the dictionary $\mathbf{C}_m$, find the element such that the objective value is minimized and accordingly set the corresponding entry of $\mathbf{y}_{nm}$ to be 1 and all the others to be 0.

Algorithm 1 The sparse composite quantization algorithm

```
Input: $\mathbf{X}, \lambda, \mu$
Output: $\mathbf{C}, \{\mathbf{y}_n\}, \xi$
1:  for each iteration do
2:      Update $\xi$;
3:      for each $c_{mkd} \in \mathcal{C}$ do
4:          Compute $\alpha, \beta$;
5:          $c^{\star}_{mkd} = S_{\lambda/\alpha}(-\frac{\beta}{\alpha})$;
6:      end for
7:      Update $\{\mathbf{y}_n\}$;
8:  end for
9:  Construct $\mathcal{C}'$;
10: for each iteration do
11:     Update $\xi$;
12:     for each $c_{mkd} \notin \mathcal{C}'$ do
13:         Compute $\alpha, \beta$;
14:         $c^{\star}_{mkd} = -\frac{\beta}{\alpha}$;
15:     end for
16:     Update $\{\mathbf{y}_n\}$;
17: end for
```

Update ξ. With $\{\mathbf{y}_n\}$ and $\mathbf{C}$ fixed, it can be easily shown that the optimal solution to the objective function with respect to ξ is,

$$\xi = \frac{1}{N}\sum\nolimits_{n=1}^{N}(\sum\nolimits_{m'\neq m}^{M}\mathbf{y}_{nm}^{\top}\mathbf{C}_m^{\top}\mathbf{C}_{m'}\mathbf{y}_{nm'}). \quad (8)$$

Update C for the problem 6. The objective function with respect to $\mathbf{C}$ is an unconstrained non-linear and non-differentiable optimization problem. We again use the alternative optimization technique to iteratively update each entry in $\mathbf{C}$: update c_{mkd} by fixing all the other entries except c_{mkd} in $\mathcal{C}$. Here c_{mkd} is the dth entry in the kth vector $\mathbf{c}_{mk}$ of the mth dictionary. The objective function with respect to c_{mkd} is a unary function consisting of two quadratic terms and an L_1 sparse term and written as a minimization problem,

$$\varphi(c_{mkd}) = \sum_{n\in\mathcal{N}_{mk}}[(x_{nd} - a_{nd} - c_{mkd})^2 + \mu(2a_{nd}c_{mkd} + b_{nd})^2] + \lambda|c_{mkd}|, \quad (9)$$

where $\mathcal{N}_{mk} = \{n|\mathbf{c}_{mk_{mn}} = \mathbf{c}_{mk}\}$, $a_{nd} = \sum_{m'=1,m'\neq m}^{M} c_{m'k_{m'n}d}$, $b_{nd} = \sum_{m'\neq m}^{M}\mathbf{c}_{mk_{mn}}^{\top}\mathbf{c}_{m'k_{m'n}} - \xi - 2a_{nd}c_{mkd}$, both a_{nd} and b_{nd} are independent of c_{mkd}. We solve the problem using the soft-thresholding technique,

$$c^{\star}_{mkd} = S_{\lambda/\alpha}(-\frac{\beta}{\alpha}) = \begin{cases} -\frac{\beta}{\alpha} + \lambda/\alpha & \frac{\beta}{\alpha} \geqslant \frac{\lambda}{\alpha} \\ 0 & |\frac{\beta}{\alpha}| < \frac{\lambda}{\alpha} \\ -\frac{\beta}{\alpha} - \lambda/\alpha & \frac{\beta}{\alpha} \leqslant -\frac{\lambda}{\alpha} \end{cases}, \quad (10)$$

where $\alpha = \sum_{n\in\mathcal{N}_{mk}}(2 + 8\mu a_{nd}^2)$, and $\beta = \sum_{n\in\mathcal{N}_{mk}}(2a_{nd} - 2x_{nd} + 4\mu a_{nd}b_{nd})$.

Table 1. Detailed description of datasets.

	Base set	Query set	Dim
MNIST	60,000	10,000	784
SIFT1M	1,000,000	10,000	128
Tiny1M	1,000,000	100,000	384
SIFT1B	1,000,000,000	10,000	128

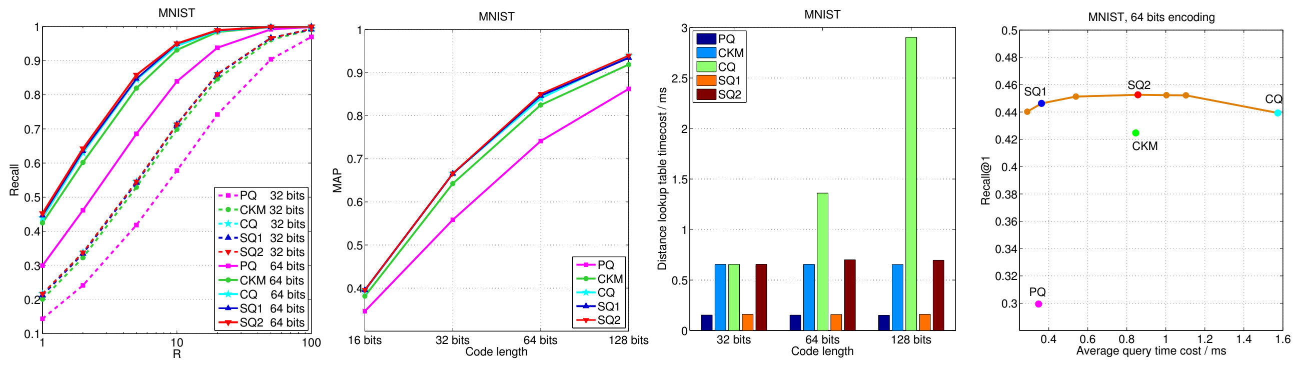
Update C for the problem 7. The solution is very simple. Dropping the sparse regularization term in 9, we have the optimal solution, $c^{\star}_{mkd} = -\frac{\beta}{\alpha}$, if $c_{mkd} \notin \mathcal{C}'$. According to the constraint, $c = 0$ if $c \in \mathcal{C}'$.

We initialize the algorithm using the solution of product quantization as a warm start. There are two parameters: the penalty parameter μ and the regularization parameter λ. Both are selected by validation, where we use a small subset of the learning set as the validation set. The best parameters μ and λ are chosen such that the mean average search accuracy using the validation set as the query set when searching for $\{5, 10, 15, ..., 100\}$ nearest neighbors is the best.

5. Discussions

Let us look at a special sparsity constraint for sparse composite quantization. We divide each dictionary vector $\mathbf{c}_{mk}$ into M subvectors, $\mathbf{c}_{mk} = [\mathbf{c}_{mk}^{1\,\top}, \mathbf{c}_{mk}^{2\,\top}, \cdots, \mathbf{c}_{mk}^{M\,\top}]^{\top}$, with the dimension of each subvector $\mathbf{c}_{mk}^r$ equal to s_m, resulting in $\sum_{m=1}^{M} s_m = D$. We change the sparsity constraint into a stronger one: $\{\|\mathbf{c}_{mk}^m\|_0 \leqslant s_m; \mathbf{c}_{mk}^r = \mathbf{0}, r \neq m, r = 1, 2, \cdots, M \mid m = 1, 2, \cdots, M; k = 1, 2, \cdots, K\}$. This is equivalent to that each dictionary $\mathbf{C}_m$ lies in a disjoint subspace, which means that sparse composite quantization is degenerated to product quantization. Naturally, the optimal distortion error when $S = KD$ in our sparse composite quantization is not larger than product quantization as the optimized solution to product quantization is a feasible solution to sparse composite quantization. This gives an evidence that sparse composite quantization in general produces better search quality than product quantization.

Cartesian k-means [22] (optimized product quantization [6]) improves the compositionablity (the vector approximation accuracy) by finding an optimal space rotation and performing product quantization over the rotated space. Our approach can benefit from the optimized space rotation by simply formulating the objective function as $\sum_{n=1}^{N}\|\mathbf{R}^{\top}\mathbf{x}_n - [\mathbf{C}_1\mathbf{C}_2\cdots\mathbf{C}_M]\mathbf{y}_n\|_2^2$ with $\mathbf{R}$ being the rotation matrix. In particular, we study the performance of sparse composite quantization with the sparsity degree reduced (the value S increased) and the overall query time cost same to Cartesian k-means.



(a) (b) (c) (d)

Figure 1. Comparison for the results of different algorithms on the MNIST dataset in terms of (a) recall@R with 32 bits and 64 bits, (b) MAP vs. the code length, (c) time cost of distance lookup table construction vs. the code length, (d) recall@1 vs. average query time cost with 64 bits encoding.

6. Experiments

6.1. Setup

Datasets. The ANN search experiments include three parts. First, we demonstrate the performance over three middle and large scale datasets. MNIST [18] includes $60K$ $784D$ raw pixel features describing grayscale images of handwritten digits as a base set, and $10K$ features as the queries. SIFT$1M$ [13] is composed of $1M$ $128D$ SIFT vectors as a base set and $10K$ queries. Tiny$1M$ [30] consists of $1M$ $384D$ GIST descriptors randomly sampled from the $80M$ images [25] as a base set and $100K$ queries from the remaining images. Second, we show the performance over a very large scale dataset using the inverted multi-index: SIFT$1B$ [15] comprises $1B$ SIFT features as a base set and $10K$ queries. The description is summarized in Table 1. Last, we apply our approach to the object retrieval problem over the INRIA Holidays dataset [12] and the UKBench dataset [21].

Compared methods. We compare the proposed approach – sparse composite quantization (SQ), against three state-of-the-art methods: product quantization (PQ), Cartesian k-means (CKM) and composite quantization (CQ). It is already shown in [33] that PQ, CKM, and CQ, with a smaller code length, achieve better search accuracy than hashing algorithms with the same query time. Thus, we do not report the results for hashing algorithms. All the results were obtained with the implementations provided by the authors or carefully re-implemented by us. We follow [13] and choose $K = 256$ in our experiments. All the algorithms conduct linear scan search using asymmetric distance for middle and large scale datasets except $1B$ SIFT for which we will show how to do search in Section 6.3. We report the results from our approach with different sparsity degrees, which includes two representative methods: SQ1 ($S = KD$) that almost takes the same query time with PQ and SQ2 ($S = \min(KD + D^2, MKD)$) that almost takes the same query time with CKM.

Evaluation. The search quality is evaluated using two measures: recall and mean average precision (MAP). Recall is defined as: for each query, we retrieve its top-ranking items and check whether the ground-truth nearest neighbor is found in the retrieved items, and the average recall score over all the queries is used as the measure. We report the recall performance at various number R of retrieved items, recall@R. The MAP score is reported by regarding the 100 nearest ground-truth neighbors as relevant answers to the query. The average precision for a query is computed as $\sum_{t=1}^{N} P(t)\Delta(t)$, where $P(t)$ is the precision at cut-off t in the ranked list and $\Delta(t)$ is the change in recall from items $t-1$ to t. We report the mean of average precisions over all the queries under different code lengths.

In terms of search efficiency, we report two measures. The first one is the time taken in constructing the distance lookup table vs. the code length, which aims to indicate that the proposed approach is able to reduce the construction cost. Second, we vary the sparsity degree in the proposed approach (i.e., vary S) and report the recall performance vs. the query cost (including both the table construction cost and the linear scan search cost) to show the superiority of our approach over PQ and CKM.

6.2. Results for medium and large scale search

Figure 1 plots the results on the MNIST dataset. It can be observed from Figures 1(a) and 1(b) that our approaches (SQ1 and SQ2) perform better than PQ and CKM in terms of recall@R and MAP under different code lengths (32 bits, 64 bits, and 128 bits). As shown in Figure 1(c), the time cost of constructing the distance lookup table for SQ1 (SQ2) is very close to PQ (CKM). In particular, one can clearly see from Figure 1(d), which shows the results of our approach with various sparsity degrees, that the proposed approach

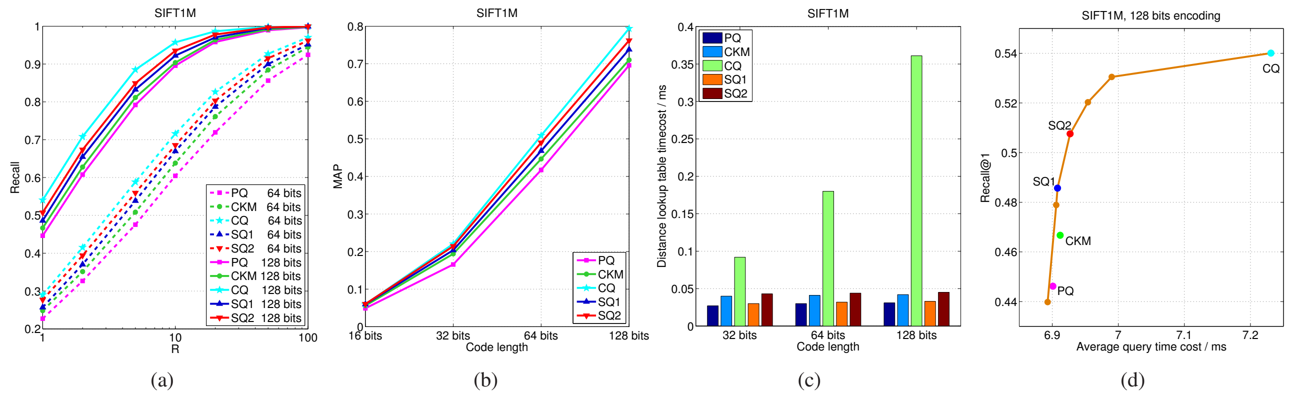


Figure 2. Comparison for the results of different algorithms on the SIFT1M dataset in terms of (a) recall@R with 64 bits and 128 bits, (b) MAP vs. the code length, (c) time cost of distance lookup table construction vs. the code length, (d) recall@1 vs. average query time cost with 128 bits encoding.

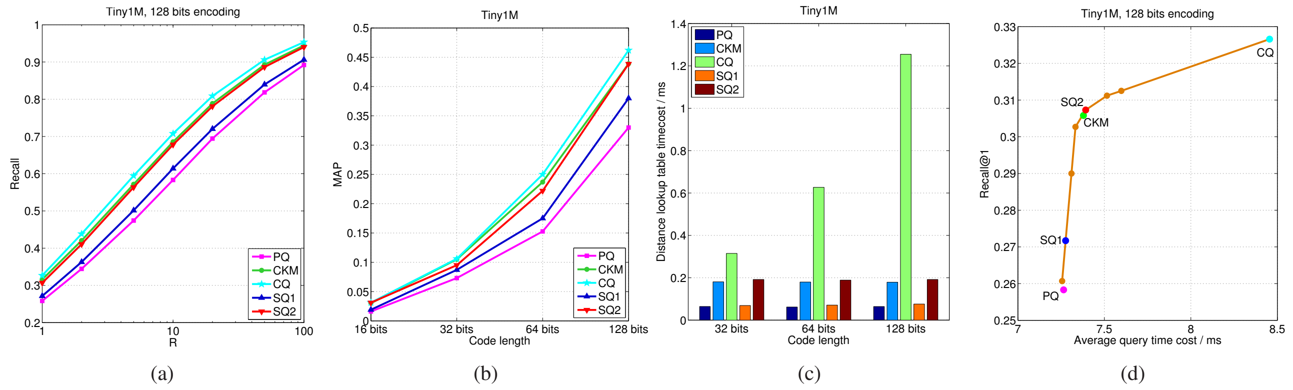


Figure 3. Comparison for the results of different algorithms on the Tiny1M dataset in terms of (a) recall@R with 128 bits, (b) MAP vs. the code length, (c) time cost of distance lookup table construction vs. the code length, (d) recall@1 vs. average query time cost with 128 bits encoding.

performs much better – with the exactly same query cost as PQ and CKM, it improves them by $\sim 15\%$ (PQ) and $\sim 3\%$ (CKM) in terms of recall@1. In comparison with CQ, our approaches surprisingly get almost the same performance as shown in Figures 1(a) and 1(b) which indicates that the introduced sparsity in some sense removes the redundancy of the dictionary elements, making it much efficient than CQ shown in Figures 1(c) and 1(d).

Figure 2 shows the results for the SIFT1M dataset. The natural order is used for PQ to achieve the best performance as suggested in [13]. The proposed approaches (both SQ1 and SQ2) show the consistent superiority to PQ and CKM in terms of recall@R and MAP, which can be seen from Figures 2(a) and 2(b). For instance, the recall@10 score of SQ1 (SQ2) with 64 bits on the SIFT1M dataset is 66.98% (68.62%), about 6% (8%) better than 60.45% of PQ and 3% (5%) larger than 63.83% of CKM. The MAP scores of SIFT1M on 64 bits for SQ1 and PQ are 0.468 and 0.417, and the relative improvement is about 12.2%. The scores for SQ2 and CKM are 0.490 and 0.447, and the improvement reaches 9.61%. In terms of distance table construction cost vs. the code length as shown in Figure 2(c), the proposed approach is also very competitive, taking negligible cost compared with PQ and CKM. From the comparison in terms of recall@1 vs. query time shown in Figure 2(d), even if with the exactly same cost of search time, the proposed approach still achieves an improvement of 1.4% over PQ and of 2.0% over CKM.

Figure 3 shows the results for the Tiny1M dataset. One can see that the proposed approach (SQ1) also consistently outperforms PQ in terms of the four evaluation schemes. Under the same code length, the performance of our approach (SQ2) is slightly worse than CKM in terms of recall and MAP (Figures 3(a) and 3(b)). The reason is that CKM is very close to CQ (equivalently to the proposed approach without the sparsity constraint), which is also observed in [33]. This does not suggest that CKM is better than the proposed approach. Figure 3(d) shows that in some sparsity constraint, the proposed approach performs almost the same as CKM. In particular, the proposed approach is more flexible and is able to adapt to various needs of query cost limits.

Table 2. Comparison of Multi-D-ADC system with different quantization algorithms in terms of recall@R with R being $1, 10, 100$, time cost (in millisecond) with database vector reconstruction ($T1$), time cost (in millisecond) without database vector reconstruction but through distance lookup tables ($T2$). L is the length of the candidate list reranked by the system.

Alg.	L	R@1	R@10	R@100	$T1$	$T2$
BIGANN, 1 billion SIFTs, 64 bits per vector						
PQ	10000	0.158	0.479	0.713	6.2	4.1
CKM		0.181	0.525	0.751	11.9	4.6
CQ		0.195	0.558	0.765	15.7	7.1
SQ1		0.184	0.530	0.736	7.3	4.3
SQ2		0.191	0.546	0.754	8.6	4.5
PQ	30000	0.172	0.507	0.814	13.2	9.8
CKM		0.193	0.556	0.851	30.3	10.1
CQ		0.200	0.597	0.869	42.6	12.9
SQ1		0.192	0.571	0. 849	15.8	9.9
SQ2		0.198	0.586	0.860	19.9	10.0
PQ	100000	0.173	0.517	0.862	37.4	30.5
CKM		0.195	0.568	0.892	95.8	31.6
CQ		0.204	0.612	0.920	125.9	33.4
SQ1		0.194	0.584	0.903	43.7	30.9
SQ2		0.199	0.597	0.907	58.6	31.2
BIGANN, 1 billion SIFTs, 128 bits per vector						
PQ	10000	0.312	0.673	0.739	7.0	5.5
CKM		0.357	0.718	0.772	12.4	5.8
CQ		0.379	0.738	0.781	29.0	7.9
SQ1		0.347	0.702	0.755	8.2	5.6
SQ2		0.368	0.725	0.773	9.5	5.7
PQ	30000	0.337	0.765	0.883	15.8	14.1
CKM		0.380	0.811	0.903	32.7	14.4
CQ		0.404	0.833	0.906	76.4	16.8
SQ1		0. 372	0.802	0.890	18.9	14.3
SQ2		0.392	0.821	0.904	25.8	14.4
PQ	100000	0.345	0.809	0.964	48.7	43.3
CKM		0.389	0.848	0.970	107.6	44.9
CQ		0.413	0.877	0.975	242.3	47.3
SQ1		0.381	0.852	0.969	59.3	43.6
SQ2		0.401	0.858	0.971	77.4	43.9

6.3. Results for very large scale search

We follow the inverted multi-index framework to solve the ANN search problem over $1B$ 128-dimensional SIFT features vectors [15]. The inverted multi-index structure use product quantization for clustering to generalize the inverted index that is typically formed by performing clustering on the database vectors and storing the list of vectors that lie in a cluster. The multi-sequence algorithm is introduced to efficiently produce a sequence of multi-index entries ordered by the increasing distances between the query and the product-quantization centers, with the aim of retrieving the NN candidates. In addition, to reduce the memory cost, the database vectors are represented with short codes, e.g., through product quantization [15], that is used to rerank the candidates retrieved from the inverted multi-index. We follow the Multi-D-ADC scheme [15] to apply the compact coding algorithm to the residual displacement between each vector and its closest coarse quantization center.

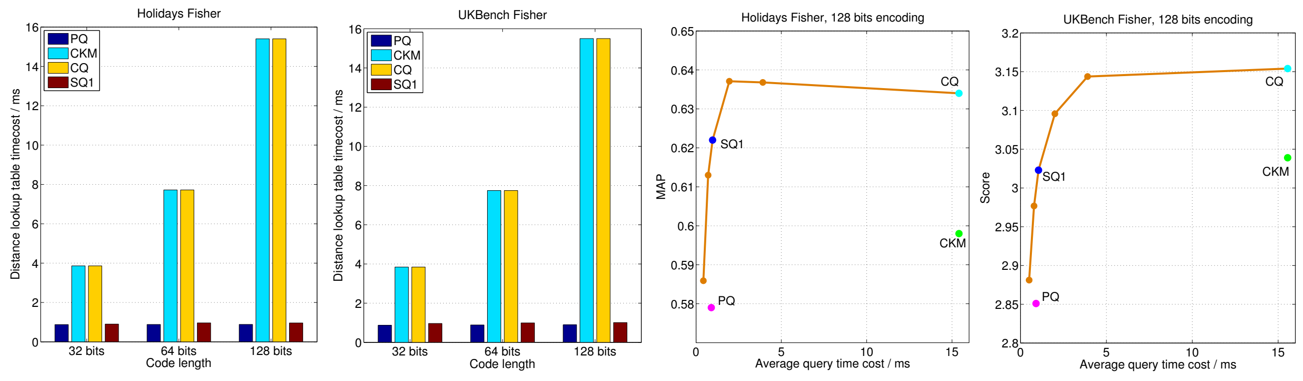
The proposed approach can be applied to build inverted multi-index (coarse quantization) and compact code representation (fine quantization). On the candidate retrieval stage, we directly apply the scheme of accelerated distance table computation before performing the multi-sequence algorithm for candidate retrieval. On the reranking stage, there are two ways for distance computation: with reconstructing the database vector, and without reconstructing the database vector but through looking up distance tables like [3]. Here we show how to accelerate the distance computation without reconstructing the database vector.

We adopt two dictionaries $\mathbf{C}_1$ and $\mathbf{C}_2$, suggested by [3] for coarse quantization, and represent the dictionaries for fine quantization by $\{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_M\}$. Let the approximation of a vector $\mathbf{x}$ be $\bar{\mathbf{x}} = \sum_{i=1}^{2} \mathbf{c}_{ik_i} + \sum_{j=1}^{M} \mathbf{r}_{jk_j}$. The acceleration idea is inspired by [3], and illustrated below. Expanding the approximated distance computation,

$$\|\mathbf{q} - (\sum_{i=1}^{2} \mathbf{c}_{ik_i} + \sum_{j=1}^{M} \mathbf{r}_{jk_j})\|_2^2 \tag{11}$$

$$\begin{aligned} &= \|\mathbf{q}\|_2^2 + \sum_{i=1}^{2} \|\mathbf{c}_{ik_i}\|_2^2 + \sum_{j=1}^{M} \|\mathbf{r}_{jk_j}\|_2^2 \\ &\quad - 2\sum_{i=1}^{2} \mathbf{q}^\top \mathbf{c}_{ik_i} - 2\sum_{j=1}^{M} \mathbf{q}^\top \mathbf{r}_{jk_j} + 2\sum_{i=1}^{2}\sum_{j=1}^{M} \mathbf{c}_{ik_i}^\top \mathbf{r}_{jk_j} \\ &\quad + 2\mathbf{c}_{1k_1}^\top \mathbf{c}_{2k_2} + 2\sum_{j=1}^{M}\sum_{m=1, m\neq j}^{M} \mathbf{r}_{jk_j}^\top \mathbf{r}_{mk_m}, \end{aligned} \tag{12}$$

we can see that the right-hand-side contains 8 terms. The first term only depends on the query, and is not necessary to compute. The second and the third terms are the summation of the L_2 norms of the selected quantization centers, where the norms are offline computed, and hence the complexity is $O(M)$. In the fourth term the inner products have been computed in the candidate retrieval stage, and this complexity is $O(1)$. The fifth term takes $O(M)$ time if the inner products between the query and the dictionary elements for the fine quantizer are precomputed. The sixth term takes $O(M)$ when the inner products between the elements of the coarse and fine dictionaries are precomputed. The last two terms are omitted because they are approximately equal to a constant. In summary, the online time complexity is $O(M)$ with precomputing the inner product table storing the inner products between the query and the fine dictionary elements.



(a) (b) (c) (d)

Figure 4. Time cost of distance lookup table construction vs. the code length on (a) the Holidays dataset and (b) the UKBench dataset and the performance (MAP and score) vs. the average query time cost with 128 bits encoding on (c) the Holidays dataset and (d) the UKBench dataset.

Table 3. The performance over the Holidays dataset in terms of MAP using different length of codes encoding.

	♯Bits	PQ	CKM	CQ	SQ
Fisher	32	0.504	0.537	0.550	0.569
	64	0.548	0.578	0.622	0.613
	128	0.579	0.598	0.634	0.622
VLAD	32	0.513	0.545	0.578	0.580
	64	0.574	0.598	0.632	0.630
	128	0.586	0.609	0.644	0.639

Table 4. The performance over the UKBench dataset in terms of scores using different length of codes encoding.

	♯Bits	PQ	CKM	CQ	SQ
Fisher	32	2.203	2.606	2.740	2.693
	64	2.618	2.894	3.009	2.902
	128	2.851	3.039	3.154	3.023
VLAD	32	2.214	2.631	2.746	2.695
	64	2.629	2.925	3.046	2.933
	128	2.878	3.069	3.185	3.056

We compare the proposed approach with three methods: PQ, CKM, and CQ and use them to train the coarse and fine quantizers. Following [15], for all the approaches, we use $K = 2^{14}$ to build coarse dictionaries and $M = 8, 16$ for compact code representation. The performance comparison in terms of recall@R ($R = 1, 10, 100$), $T1$ (query time cost for the scheme with database vector reconstruction), $T2$ (query time cost for the scheme without database vector reconstruction but through distance lookup tables) with respect to the length of the retrieved candidate list L ($L = 10000, 30000, 100000$) is summarized in Table 2. It can be seen from the $T1$ and $T2$ columns that like PQ and CKM, our approaches, SQ1 and SQ2, are both accelerated, and according to the $T2$ column, the query costs of our approaches (SQ1 and SQ2) are almost the same to that of the most efficient approach PQ. Considering the overall performance in terms of the query cost for the accelerated scheme ($T2$) and the recall, one can see that SQ2 performs the best. For example, the query cost of SQ2 is 36% smaller than that of CQ when retrieving 10000 candidates with 64 bits and the recall decreases less than 2.2%. In other cases, the recall decrease is always less than 3% while the query cost is saved at least 7%.

6.4. Application to object retrieval.

We have also evaluated our method on the application of compact codes [24, 6] to object retrieval over two datasets: the INRIA Holidays dataset that contains 500 queries and 991 corresponding relevant images and the UKBench dataset that contains 10200 images of 2550 groups with each four images. We follow [12, 21] and evaluate the performance over the INRIA Holidays dataset using mean average precision (MAP) and performance over the UKBench dataset using the score indicating how many of the other images are in the top-4 rank where one image of each group is used as query. We use 4096-dimensional Fisher vectors [23] and VLAD vectors [14] as the image descriptors.

The search results are shown in Tables 3 and 4. One can observe that the performance of our approach SQ ($S = KD$) is very competitive, overall better than CKM and PQ and very close to CQ. In terms of the search efficiency as shown in Figure 4, it can be observed that our approach and PQ are the most efficient.

7. Conclusion

This paper presents a compact coding approach to approximate nearest neighbor search, called sparse composite quantization. The proposed approach generalizes the composite quantization by sparsifying the elements in the dictionaries resulting in the efficient construction of a lookup table storing distances between the query and the dictionary words. The proposed approach in the applications to approximate nearest neighbor search achieves superior performance in terms of search accuracy and search efficiency.

Acknowledgements

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

References

[1] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, pages 931–939, 2014. 2

[2] A. Babenko and V. S. Lempitsky. The inverted multi-index. In *CVPR*, pages 3069–3076, 2012. 2, 3

[3] A. Babenko and V. S. Lempitsky. Improving bilayer product quantization for billion-scale approximate nearest neighbors in high dimensions. *CoRR*, abs/1404.1831, 2014. 2, 7

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975. 2

[5] C. Du and J. Wang. Inner product similarity search using compositional codes. *CoRR*, abs/1406.4966, 2014. 2

[6] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2946–2953, 2013. 1, 2, 4, 8

[7] Y. Gong, S. Kumar, V. Verma, and S. Lazebnik. Angular quantization-based binary codes for fast similarity search. In *NIPS*, pages 1205–1213, 2012. 2

[8] 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. Pattern Anal. Mach. Intell.*, 35(12):2916–2929, 2013. 2

[9] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik. Asymmetric distances for binary embeddings. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(1):33–47, 2014. 2

[10] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. 2

[11] J.-P. Heo, Z. Lin, and S.-E. Yoon. Distance encoded product quantization. In *CVPR*, pages 2139–2146, 2014. 2

[12] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV (1)*, pages 304–317, 2008. 5, 8

[13] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011. 1, 2, 5, 6

[14] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, 2010. 8

[15] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: Re-rank with source coding. In *ICASSP*, pages 861–864, 2011. 5, 7, 8

[16] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2329–2336, 2014. 2

[17] W. Kong and W.-J. Li. Isotropic hashing. In *NIPS*, pages 1655–1663, 2012. 2

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001. 5

[19] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012. 2

[20] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP (1)*, pages 331–340, 2009. 2

[21] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR (2)*, pages 2161–2168, 2006. 5, 8

[22] M. Norouzi and D. J. Fleet. Cartesian k-means. In *CVPR*, pages 3017–3024, 2013. 1, 2, 4

[23] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 8

[24] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3384–3391, 2010. 8

[25] A. B. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008. 5

[26] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, 2012. 2

[27] J. Wang and S. Li. Query-driven iterated neighborhood graph search for large scale indexing. In *ACM Multimedia*, pages 179–188, 2012. 2

[28] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *CoRR*, abs/1408.2927, 2014. 2

[29] J. Wang, J. Wang, J. Song, X.-S. Xu, H. T. Shen, and S. Li. Optimized cartesian k-means. *CoRR*, abs/1405.4054, 2014. 2

[30] J. Wang, N. Wang, Y. Jia, J. Li, G. Zeng, H. Zha, and X.-S. Hua. Trinary-projection trees for approximate nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(2):388–403, 2014. 5

[31] B. Xu, J. Bu, Y. Lin, C. Chen, X. He, and D. Cai. Harmonious hashing. In *IJCAI*, 2013. 2

[32] F. X. Yu, S. Kumar, Y. Gong, and S. Chang. Circulant binary embedding. In *ICML*, pages 946–954, 2014. 2

[33] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *ICML (2)*, pages 838–846, 2014. 1, 2, 3, 5, 6