Efficient Indexing of Billion-Scale datasets of deep descriptors

Artem Babenko
Yandex
Moscow Institute of Physics and Technology
artem.babenko@phystech.edu

Victor Lempitsky
Skolkovo Institute of Science and Technology
(Skoltech)
lempitsky@skoltech.ru

Abstract

Existing billion-scale nearest neighbor search systems have mostly been compared on a single dataset of a billion of SIFT vectors, where systems based on the Inverted Multi-Index (IMI) have been performing very well, achieving state-of-the-art recall in several milliseconds. SIFT-like descriptors, however, are quickly being replaced with descriptors based on deep neural networks (DNN) that provide better performance for many computer vision tasks.

In this paper, we introduce a new dataset of one billion descriptors based on DNNs and reveal the relative inefficiency of IMI-based indexing for such descriptors compared to SIFT data. We then introduce two new indexing structures, the Non-Orthogonal Inverted Multi-Index (NO-IMI) and the Generalized Non-Orthogonal Inverted Multi-Index (GNO-IMI). We show that due to additional flexibility, the new structures are able to adapt to DNN descriptor distribution in a better way. In particular, extensive experiments on the new dataset demonstrate that these data structures provide considerably better trade-off between the speed of retrieval and recall, given similar amount of memory, as compared to the standard Inverted Multi-Index.

1. Introduction

Modern image retrieval systems[19] have to search through billion-scale databases in several milliseconds to respond to user queries. This imposes strict demands on scalability and efficiency of the underlying nearest-neighbor search algorithms. As exhaustive search is mostly infeasible at this scale, approximate nearest neighbor (ANN) methods that restrict the part of a dataset that is being considered in response to a query have to be used.

State-of-the-art ANN algorithms of this kind [13, 2, 9, 14, 5] avoid exhaustive search by organizing dataset points using an *indexing structure*. For a given query the indexing structure allows to retrieve a *short-list* of candidate points that are likely to be close to this query. The candidate points are then reranked according to their distances using such methods as Asymmetric Distance Computation (ADC) [12]. In general, the reranking runtime is approximately linear in the size of the short-list. Consequently, the ability to obtain compact short-lists with high enough recall leads to fast reranking and overall time efficiency.

One of the first billion-scale retrieval systems for billion-scale datasets have been presented by Jegou et al. [13]. Their system called IVFADC divides data space into separate *cells*, corresponding to Voronoi regions obtained via k-means clustering. Given a query, IVFADC returns the short list with the points from the cells that are the closest to it. The Inverted Multi-Index (IMI)[2] generalizes IVFADC by decomposing the dataspace into several subspaces and splitting each subspace into cells independently. The system [14], based on the IMI indexing structure and the Product Quantization (PQ) compression provides state-of-the-art performance on the SIFT1B dataset of one billion SIFT descriptors [13], which invariably serves as the testbed for such systems.

Recently, the focus in computer vision has shifted from SIFT descriptors to descriptors produced by deep neural networks (DNN). Several works [17, 7, 11, 18, 4] have shown that DNN-based descriptors significantly improve the state-of-the-art on common retrieval benchmarks. Furthermore, [7] have shown that deep descriptors can be compressed by PCA with almost no loss in performance.

Despite such shift to deep descriptors, to the best of our knowledge, billion-scale ANN with deep descriptors has not been investigated. One potential problem with relying on the SIFT1B dataset is its particular suitability for IMI-based systems. This is because, the SIFT construction procedure uses different halves of a SIFT vector to describe disjoint regions of an image patch. Thus, the correlations between the two halves are likely to be small and the decomposition in the IMI is justified. Deep descriptors do not possess such structure and therefore the good performance of IMI might not be achieved.

This paper seeks to develop a good indexing structure for deep descriptors. We begin with experiments illustrating that for this type of data the Inverted Multi-Index indeed

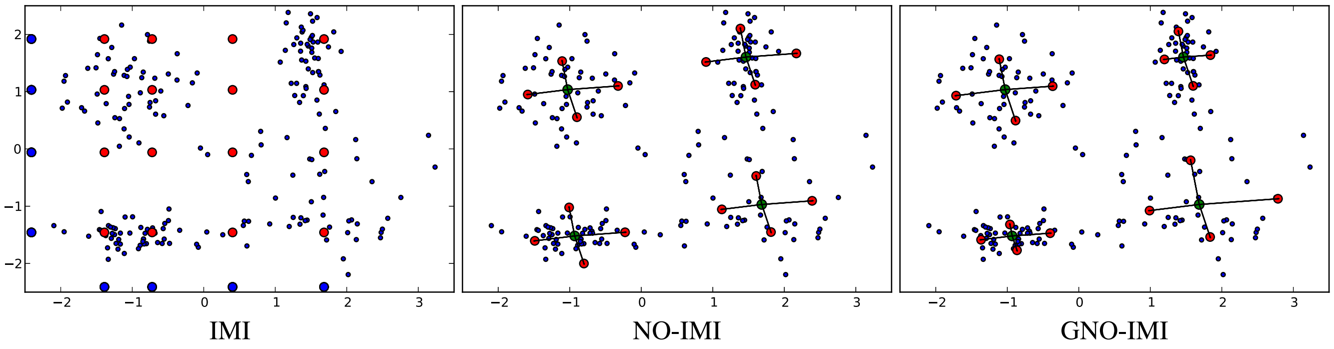

Figure 1. The cell centroids (red points) produced by different indexing structures for the same sample of two-dimensional data (blue points). For all structures parameter K was set to four, hence 16 centroids were produced. The left plot corresponds to the IMI structure and large blue points on the axes denote the codewords of the underlying PQ decomposition. The middle and right plots correspond to the NO-IMI and the GNO-IMI structures respectively. On the both plots green points correspond to the "first-order centroids" $S_1, \ldots, S_4$. The GNO-IMI centroids represent the actual data distribution more accurately than the other structures.

produces a particularly large number of redundant cells that do not contain any points. This deficiency arises from the space decomposition, implicitly performed in the IMI. We conclude that for deep data the correlations between subspaces are significant and independent codebooks for each subspace cannot represent the global data distribution adequately.

We then introduce two new indexing structures, the Non-Orthogonal Inverted Multi-Index (NO-IMI) and the Generalized Non-Orthogonal Inverted Multi-Index (GNO-IMI). The NO-IMI forms index cells centroids as sums of two vectors from two non-orthogonal codebooks, following the well-known residual vector quantization (RVQ) scheme [8], thus avoiding any decomposition into orthogonal subspaces. As a result, the NO-IMI provides more reasonable index cells with the centroids representing actual data distribution more accurately. The GNO-IMI pushes the representation accuracy even further by forming cell centroids as linear combinations of codewords. As the extra linear coefficients are learned from the data, the GNO-IMI exhibits higher efficiency while indexing complex data as compared to the NO-IMI.

We demonstrate that better indexing by the (G)NO-IMI results in more compact and precise short-lists and, therefore, more efficient retrieval. Overall, we show that for the same time budget the new structures provide higher recall compared to the existing top-performing schemes. Our evaluation is based on a new dataset of one billion deep descriptors with hold-out query and learning sets. The dataset is the first publicly available billion-scale dataset of deep descriptors.

2. Related work

We briefly describe several ideas from the previous work that are essential to the description of the our structures. Along the way, we introduce notation for this description.

Product quantization (PQ) is a lossy compression scheme for high-dimensional vectors [12]. PQ encodes each vector $x \in \mathbf{R}^D$ as a concatenation of M codewords from M $\frac{D}{M}$-dimensional codebooks $C_1, \ldots, C_M$, each containing K codewords. In other words, PQ decomposes a vector into M separate subvectors and applies vector quantization (VQ) to each subvector, while using a separate codebook. As a result each vector x is encoded by a tuple of codewords indices $[i_1, \ldots, i_M]$ and approximated by $x \approx [C_1(i_1), \ldots, C_M(i_M)]$. Fast Euclidean distance computation becomes possible via efficient ADC procedure [12] using lookup tables:

$$\|q - x\|^2 \approx \|q - [C_1(i_1), \ldots, C_M(i_M)]\|^2 = \sum_{m=1}^{M} \|q_m - C_m(i_m)\|^2 \quad (1)$$

where q_m — mth subvector of a query q. This sum can be calculated in M additions and lookups given that distances from query subvectors to codewords are precomputed.

From the geometry viewpoint, PQ effectively partitions the original vector space into K^M cells, each being a Cartesian product of M lower-dimensional cells. Such product-based approximation works better if the $\frac{D}{M}$-dimensional components of vectors have independent distributions. The degree of dependence is affected by the choice of the splitting, and can be further improved by orthogonal transformation applied to vectors as preprocessing. Two recent works have therefore looked into finding an optimal transformation [10, 16]. Following one of them, the modification of PQ corresponding to such pre-processing transformation is referred below as *Optimized Product Quantization* (OPQ).

Non-orthogonal quantizations. Several works [8, 3, 21, 6] have proposed modifications of PQ that do not decompose the dataspace into orthogonal subspaces. In fact, they generalize the idea of Product Quantization by approximating each vector by a sum of M codewords instead of concatenation. In this case the ADC procedure is still efficient while the approximation accuracy is increased.

The first approach, Residual Vector Quantization [8], quantizes original vectors, and then iteratively quantizes the approximation residuals from the previous iteration. [8] also introduces the IVFRVQ system that allows to perform non-exhaustive search. IVFRVQ prunes large portions of database points based on the distances from a query to the "coarse approximations", produced by the first several quantizers. Another approach, Additive Quantization (AQ) [3], is the most general as it does not impose any constraints on the codewords from the different codebooks. Usually, AQ provides the smallest compression errors, however, it is much slower than other methods, especially for large M. Composite Quantization (CQ) [21] learns codebooks with a fixed value of scalar product between the codewords from different codebooks. Finally, Tree Quantization (TQ) [6] performs decomposition into subspaces while constraining the non-orthogonality relations between different codebooks to form a tree graph. Under this constraint, efficient encoding is possible.

Overall, non-orthogonal quantizations achieve higher approximation accuracy than (O)PQ, especially for non-histogram data. At the same time, they are slower and require more complex learning procedures.

IVFADC. One of the first systems capable of dealing with billion-scale datasets efficiently was IVFADC introduced in [13]. This system combines the inverted index [19] as indexing structure and Product Quantization for database compression. IVFADC first splits the space into K cells via standard K-means and then encodes displacements of each point from the centroid of a cell it belongs to. The encoding is performed via Product Quantization that uses global codebooks shared by all cells.

The Inverted Multi-Index and Multi-D-ADC. The Inverted Multi-Index (IMI) [2] is an indexing algorithm for high-dimensional spaces and very large datasets. The IMI generalizes the inverted index by using product codebooks for cell centroids construction (typically, as few as two components in the product are considered). Thus the Inverted Multi-Index has two $\frac{D}{2}$-dimensional product codebooks for different halves of the vector, each with K sub-codewords, thus effectively producing K^2 cells, which would typically be orders of magnitude bigger than the number of cells within the IVFADC system or other systems using inverted indices. Large number of cells provides very dense partitioning of the space, which means that a small fraction of dataset has to be traversed to achieve high recall (w.r.t. the true nearest neighbor). For dataset compression, [2] followed the IVFADC system and used Product Quantization with global codebooks shared across all cells in order to encode the displacements of the vectors from centroids (this system is referred to as *Multi-D-ADC*).

Further improvements. [9] improved the performance of Multi-D-ADC by replacing Product Quantization with Optimized Product Quantization for both indexing and compression (hence the name of their system *OMulti-D-OADC*). OMulti-D-OADC gives the state-of-the-art performance in terms of the search accuracy on the SIFT1B dataset with global codebooks for database compression.

Another system, called LOPQ [14], is a modification of IVFADC which uses separate *local* OPQ codebooks for database points compression in each cell. As local codebooks represent points distribution in the particular cell more precisely, the accuracy of reranking in LOPQ is much higher than in IVFADC. Also, [14] introduces the Multi-LOPQ system, which uses local codebooks with the Inverted Multi-Index structure. Currently, Multi-LOPQ system provides the highest recall on the SIFT1B dataset.

3. The Non-Orthogonal Inverted Multi-Index

In this section we introduce a new indexing structure, the Non-Orthogonal Inverted Multi-Index (NO-IMI). We first describe the design of the NO-IMI and the intuition behind it. We then provide a procedure of the NO-IMI codebooks learning. Finally, we explain how candidate lists are extracted and are reranked using the NO-IMI.

3.1. The NO-IMI structure

Let us assume that a database $P = \{p_1, \dots, p_N\}$ of D-dimensional points is given. As well as the existing indexing structures the NO-IMI partitions the dataspace into a large number of cells and distributes database points over these cells. The cells in the NO-IMI are constructed based on two codebooks $S = \{S_1, \dots, S_K\}$ and $T = \{T_1, \dots, T_K\}$, each containing K codewords. The codewords in the codebooks are D-dimensional, $S, T \subset \mathbf{R}^D$.

The NO-IMI splits the data space into K^2 cells with centroids c_i^j defined by the formula:

$$c_i^j = S_i + T_j, \qquad i, j = 1, \dots, K \tag{2}$$

Thus, the NO-IMI cells are the regions of the data space defined by the following:

$$C_i^j = \{x \in \mathbf{R}^D | i, j = \arg\min_{k,l} \|x - (S_k + T_l)\|^2\} \tag{3}$$

This construction is similar to how cells within the Inverted Multi-Index are constructed. We however aim to avoid the space decomposition, underlying the Inverted Multi-Index, which is based on product quantization and which ignores correlations between subspaces of the deep descriptors data. Based on this consideration, it is reasonable to replace PQ by some non-orthogonal quantization.

Among non-orthogonal approaches, we chose Residual Vector Quantization (RVQ) as it allows to perform efficient short-lists extraction, as will be discussed below. With the RVQ method in mind, the NO-IMI codewords $S_1, \dots, S_K$

can be interpreted as cluster centroids in the original dataspace and we refer to them as *first-order centroids*. Similarly, the codewords $T_1, \ldots, T_K$ can be interpreted as centroids in the space of displacements of data points from the first-order centroids. We refer to them as *second-order centroids*. As a result, the indexing structure has K^2 centroids in the form $(S_i + T_j)$, $i, j = 1, \ldots, K$ and the same number of cells, each assigned with the *first-order index* i and the *second-order index* j.

The construction (2) is also directly related to the well-known Hierarchical K-Means (HKM) method [15]. This method also clusters the original data space, producing K first-order centroids. It then clusters the displacements in each cluster separately, producing K second-order centroids in each cluster. For large K, the usage of HKM is infeasible as it would require to keep K^2 D-dimensional vectors in the main memory. Such enormous memory consumption can be avoided by using K second-order centroids that are shared by all clusters. Sharing the second-order centroids makes Hierarchical K-Means equivalent to the RVQ scheme, described in the previous paragraph.

3.2. The generalized NO-IMI

The NO-IMI assumes that the displacements distributions within first-order clusters are similar, as it uses shared second-order centroids. However, this assumption may be too strong, as e.g. the spatial extent and the shape of the corresponding Voronoi cells can vary, especially when the data distribution has dense and sparse regions. To alleviate this assumption we augment the NO-IMI with an additional *α-matrix*, which incorporates cluster-wise information. In particular, an element $\alpha[i,j]$ is a factor for the j-th second-order centroid T_j in the i-th first-order cluster. After the incorporation of these factors, the expression for the corresponding centroid becomes:

$$c_i^j = S_i + \alpha[i,j] T_j, \qquad i, j = 1, \ldots, K \tag{4}$$

The addition of α-matrix allows to represent data distribution adequately even with shared second-order centroids. Below we refer to the version of the NO-IMI with the α-matrix as *Generalized Non-Orthogonal Inverted Multi-Index* (GNO-IMI). We demonstrate an example of centroids produced by the original IMI, the NO-IMI, and the GNO-IMI on a toy 2D dataset in Figure 1.

3.3. Indexing

The parameters of (G)NO-IMI, namely S, T, α, can all be learned from data as will be discussed below. Let us now assume that S, T, α are given and describe the process that distributes the dataset points $p_1, \ldots, p_N$ assigning them to different (G)NO-IMI cells. More formally, for each point p we have to find the closest centroid indices:

$$i, j = \arg\min_{k,l} \|p - (S_k + \alpha[k,l] T_l)\|^2 \tag{5}$$

Exhaustive minimization of this function would require the evaluation of K^2 possible solutions, which is impractical. Instead, we use a simple heuristic to cut out most of the suboptimal solutions. Let us rewrite the expression above:

$$\begin{aligned} \|p - (S_k + \alpha[k,l] T_l)\|^2 &= \|p - S_k\|^2 + \alpha[k,l]^2 \|T_l\|^2 \\ &- 2\alpha[k,l]\langle p, T_l\rangle + 2\alpha[k,l]\langle S_k, T_l\rangle \end{aligned} \tag{6}$$

Note, that the first term $\|p - S_k\|^2$ is a distance from p to the first-order centroid S_k. The RVQ heuristic is to inspect only r values $k_1, \ldots, k_r$ corresponding to the indices of centroids $S_{k_1}, \ldots, S_{k_r}$, which are the closest to p. With this pruning, only rK solutions have to be evaluated. Another speed optimization is to precompute the terms $\|p - S_k\|^2$ and $\langle p, T_l\rangle$, which can be done in $O(KD)$ operations. With these terms precomputed, each solution can be evaluated in $O(1)$ operations. After all rK possibilities are evaluated, p is assigned to the cell, corresponding to the optimal index pair.

The complexity of point assignment is therefore a sum of complexities of precomputation and pairs evaluation steps, which are $O(DK)$ and $O(rK)$ respectively. Hence, the overall complexity is $O(DK + rK)$, while the complexity in the Inverted Multi-Index equals $O(DK)$. In our experiments r is several times smaller than D and the actual timings of indexing for both the IMI and the (G)NO-IMI are rather similar.

3.4. Codebooks learning

We now focus on the task of obtaining codebooks S, T and the α-matrix that fit a given data distribution well. We suppose that a training dataset $P = \{p_1, \ldots, p_N\}$ is provided. The learning is performed by the minimization of the reconstruction error for all training points:

$$\sum_{i=1}^{N} \left\| p_i - (S_{k^i} + \alpha[k^i, l^i] T_{l^i}) \right\|^2 \rightarrow \min_{\substack{S_k \in \mathbf{R}^D, |S| = K, \\ T_k \in \mathbf{R}^D, |T| = K, \\ \alpha \in \mathbf{R}^{D \times D} \\ k^i, l^i \in \{1, \ldots, K\}}} \tag{7}$$

Overall, the optimization is performed over four groups of variables, where S, T and α variables are continuous, while the assignment variables $\{k^i, l^i\}$ are discrete. We minimize the function using block-coordinate descent by optimizing over one variable group at a time with the other three groups fixed. Below we describe the optimization over each variable group separately.

Optimization over assignments. At this step the optimization is performed over $\{k^i, l^i\}$ variables given codebooks S, T and α matrix. Thus, this problem is equivalent to the indexing task and we discuss an efficient method for that above in Section 3.3.

Optimization over α. We now fix the assignments $\{k^i, l^i\}$ and the codebooks S, T and minimize over the elements of the α matrix. Let us show that it is possible to minimize over each matrix element $\alpha[k,l]$ independently. For that we decompose the target function as follows:

$$\sum_{i=1}^{N} \left\|p_i - (S_{k^i} + \alpha[k^i,l^i]T_{l^i})\right\|^2 =$$

$$= \sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i:k_i=k,l_i=l} \|p_i - S_k - \alpha[k,l]T_l\|^2 \tag{8}$$

Note, that each term $\sum\limits_{i:k_i=k,l_i=l} \|p_i - S_k - \alpha[k,l]T_l\|^2$ depends on the only element $\alpha[k,l]$. Hence, the minimization of the whole function is equivalent to the independent minimizations of each term:

$$\sum_{i:k_i=k,l_i=l} \|p_i - S_k - \alpha[k,l]T_l\|^2 \to \min_{\alpha[k,l]} \tag{9}$$

This subproblem can be solved analytically and its closed-form solution is:

$$\alpha[k,l] = \frac{\sum\limits_{i:k_i=k,l_i=l} \langle p_i - S_k, T_l\rangle}{\sum\limits_{i:k_i=k,l_i=l} \|T_l\|^2} \tag{10}$$

Optimization over second-order codebook T. Similarly, we optimize over the second-order codewords T_l by decomposing the target function again:

$$\sum_{i=1}^{N} \left\|p_i - (S_{k^i} + \alpha[k^i,l^i]T_{l^i})\right\|^2 =$$

$$\sum_{l=1}^{K}\sum_{k=1}^{K}\sum_{i:k_i=k,l_i=l} \|p_i - S_k - \alpha[k,l]T_l\|^2 \tag{11}$$

Then each term $\sum\limits_{k=1}^{K}\sum\limits_{i:k_i=k,l_i=l} \|p_i - S_k - \alpha[k,l]T_l\|^2$ is minimized over T_l independently. This minimization also can be solved analytically and the solution has the form:

$$T_l = \frac{\sum\limits_{k=1}^{K} \alpha[k,l] \sum\limits_{i:k_i=k,l_i=l} (p_i - S_k)}{\sum\limits_{k=1}^{K}\sum\limits_{i:k_i=k,l_i=l} \alpha[k,l]^2} \tag{12}$$

Optimization over first-order codebook S. This problem is also decomposable and the minimization is performed completely analogously to the previous step. The solution to this step has the form

$$S_k = \frac{\sum\limits_{l=1}^{K}\sum\limits_{i:k_i=k,l_i=l} (p_i - \alpha[k,l]T_l)}{\sum\limits_{l=1}^{K}\sum\limits_{i:k_i=k,l_i=l} 1} \tag{13}$$

Initialization. We initialize the iterations of the GNO-IMI learning in the following way. The α-matrix is initialized by all ones and S is initialized by centroids obtained via K-means clustering of the dataset P. Second-order codebook T is initialized by centroids obtained via K-means clustering of the displacements from the database points to the closest first-order codewords.

Comments. In practice we observed that the method requires quite a few iterations to converge and we used ten iterations in our experiments. Parameter r was set to eight. Also note, that for some index pairs k,l there could be no points and then both the numerator and the denominator in the expression (10) vanish. In this case we set $\alpha[k,l] = 1$.

3.5. Short-list extraction

Here we describe the procedure that gets a query q and produces an ordered sequence of cells that correspond to the centroids that are the closest to q. The sequence can be of any desired length L and the points assigned to these cells form the short-list that is returned by the GNO-IMI in response to the query. The procedure consists of three steps:

1. First, we compute the distances $\|q - S_k\|^2$ between the query and the first-order centroids. Let us denote by $k_1,\dots,k_r$ indices of r first-order centroids that appeared to be the closest to q.

$$distances[k] = \|q - S_k\|^2, \quad k = 1,\dots,K$$
$$k_1,\dots,k_r = distances.argsort()[1:r]$$

We then discard all the cells with first-order indices not belonging to the set $\{k_1,\dots,k_r\}$. The subsequent search is performed among the remaining rK cells only.

$$cells = \{(k_i;l)\}$$
$$k_i \in \{k_1,\dots,k_r\}, \quad l = 1,\dots,K$$

The complexity of this step is $O(KD + K\log K)$ with the two terms corresponding to distances calculation and sorting.

2. Secondly, the procedure creates an array of the length rK and fills it with the distances to the centroids of the promising cells selected in step one:

```
for each $cell = (k_i;l)$ in $cells$:
```

$$cellDistances[(k_i;l)] = \|q - S_{k_i}\|^2 + \alpha[k_i,l]^2\|T_l\|^2$$
$$-2\alpha[k_i,l]\langle q,T_l\rangle + 2\alpha[k_i,l]\langle S_{k_i};T_l\rangle$$

Note, that the terms $\langle q,T_l\rangle$ can be precomputed once and reused for all k_i and l. The terms $\langle S_{k_i},T_l\rangle$ and the norms $\|T_l\|^2$ are precomputed before querying and are kept in lookup-tables. Thus, the complexity of this step is $O(rK)$.

3. Finally, we apply partial sorting of distances array with linear average complexity[1]. For any input array this partial sorting guarantees that the L smallest elements will be placed in the first L position of the output array (possibly unordered), where L is the desired length of

[1]std::nth_element from the standard STL library

cells sequence. Then only L cells corresponding to the smallest distances are sorted and the sorted sequence is yielded as a procedure result.

$PartialSort(cellDistances[], L)$

$Output = Sort(cellDistances[1:L])$

The complexity of this step is $O(rK + L \log L)$.

The total complexity of the procedure is a sum of complexities for the three steps and equals $O(KD + K \log K + rK + L \log L)$. In this sum the terms $O(KD)$ and $O(K \log K)$ are common for existing indexing structures as they also perform query quantization and sort distances to centroids. The term $O(rK + L \log L)$ is a computational overhead in the (G)NO-IMI but we show in the next section that it is insignificant and has no influence on the scheme applicability.

After forming the sorted sequence of cells as described above, the method starts to traverse the cells and collects points from these cells into a candidate list.

3.6. Reranking

After short-list extraction the (G)NO-IMI performs reranking of candidate points in the same manner as it was proposed in [2, 14]. During the indexing stage we calculate the displacement of each point from its cell centroid and compress these displacements via Optimized Product Quantization (OPQ). In the experiments, we used K sets of local PQ codebooks, and each set was shared by the cells with the same first-order index. We used one global rotation matrix for all database points.

When querying, each candidate point is reconstructed from its OPQ code and the distance between this reconstruction and the given query is evaluated. Finally, candidates are sorted based on the calculated distances.

3.7. Connection to IVFRVQ

The proposed NO-IMI system is similar to the IVFRVQ scheme proposed in [8]. Here we highlight the differences between the NO-IMI and IVFRVQ:

1. IVFRVQ is based on RVQ only, while NO-IMI uses OPQ to compress the data at the fine level. Not relying on RVQ at all levels is important for high perfromance, since RVQ degrades when it is applied with large number of codebooks [8].

2. The NO-IMI does not need to keep the norms of dataset points. IVFRVQ has to keep them, which needs extra memory and also extra operations during search.

3. IVFRVQ evaluates distances to all cell centroids and then explores only several inverted lists corresponding to the closest cells. The NO-IMI does not evaluate distances to all centroids and performs more efficient non-exhaustive evaluation.

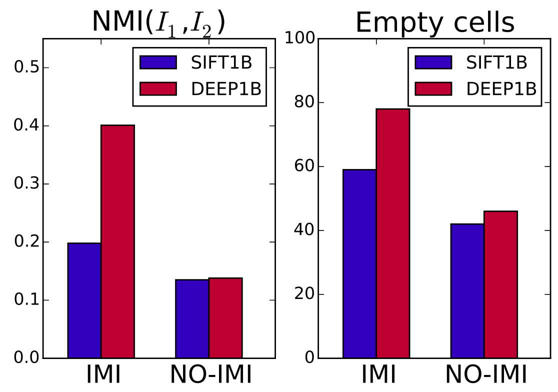

Figure 3. Normalized mutual information between the indices of the closest codewords in two codebooks (left) and the percent of empty index cells (right) for the IMI and NO-IMI systems. The high value of NMI in the case of DEEP1B and the IMI means that the implicit assumption of independent subspaces in the IMI does not hold for DEEP1B. The large percent of empty cells also indicates that cell centroids in the IMI represent DEEP1B data distribution poorly.

4. Experiments

In this section we provide results of experiments that compare the Non-Orthogonal Inverted Multi-Indices (NO-IMI and GNO-IMI) with the standard Inverted Multi-Index. The experiments were performed on two datasets:

1. **SIFT1B** dataset[13] that contains one billion of 128-dimensional SIFT descriptors along with precomputed groundtruth for $10,000$ queries. A hold-out learning set of 100 million descriptors is also provided.

2. **DEEP1B** dataset. This is a new dataset that we introduce to the community[2]. Descriptors for DEEP1B were produced in a way similar to [7]. Specifically, we took the outputs of the last fully-connected layer of a DNN for a billion images on the Web. Our DNN had the GoogLeNet[20] architecture and was trained on the ImageNet dataset[1]. The outputs were then compressed by PCA to 96 dimensions and l_2-normalized. We also prepared hold-out sets containing 350 millions of descriptors for learning and $10,000$ for querying (with known ground truth for nearest neighbors in the main set).

For both datasets we compare three indexing structures:

1. the inverted multi-index (IMI) with global rotation before dataspace decomposition [9]. The top-performing systems [14, 5] use this indexing structure, so we use it as a baseline. We used the authors' implementation[5];
2. the Non-Orthogonal Inverted Multi-Index (NO-IMI);
3. the Generalized Non-Orthogonal Inverted Multi-Index (GNO-IMI).

[2]The dataset and the project code are available on http://sites.skoltech.ru/compvision/noimi/

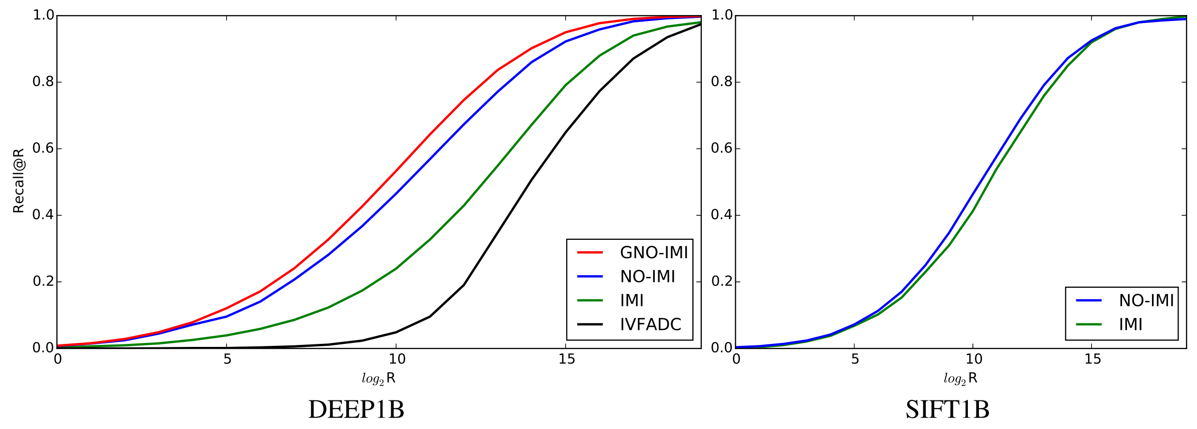

Figure 2. Recall as a function of the candidate list length. On DEEP1B (left plot) we compare four systems: IVFADC with 2^{17} codewords, the IMI with $K = 2^{14}$ and preliminary orthogonal transformation, the NO-IMI and the GNO-IMI with $K = 2^{14}$. For all recall levels the (G)NO-IMI provides much shorter candidate lists. For SIFT1B dataset (right plot) the advantage of the NO-IMI is negligible.

Dataset	IMI	NO-IMI	GNO-IMI
SIFT1B	35923	35207	34981
DEEP1B	0.321	0.272	0.255

Table 1. Average distances from data points to the closest centroid. Smaller distances usually results in more precise short-lists.

For all the structures we used the same value of $K = 2^{14}$, hence all the indices have the same number of cells. All the structures and the codebooks for database compression were optimized on the hold-out learning sets.

Representation accuracy. We first check our intuition that the subspaces corresponding to different halves of deep data are more correlated for deep data compared to SIFT descriptors. As described in section 2, the IMI centroids are constructed using codewords from the two codebooks C_1 and C_2, corresponding to orthogonal data subspaces. Let us introduce two discrete random variables I_1, I_2 corresponding to the indices of the closest codewords from C_1 and C_2 respectively for a certain point. We can measure the normalized mutual information between I_1 and I_2 using all the points from the particular dataset. High values of mutual information would then reveal the correlation between the halves.

Figure 3 (left) shows values of the normalized mutual information (NMI) for both datasets. For DEEP1B dataset $NMI(I_1, I_2)$ is two times higher than for SIFT1B. This verifies the intuition that the decomposition for deep data can lead to low performance, even when decorrelating global rotation (as in OPQ) is fitted during preprocessing. Figure 3 also shows NMI values for the NO-IMI scheme. In this case we measure the mutual information between the indices of the closest codewords in codebooks S and T. For both SIFT1B and DEEP1B the NMI is markedly smaller in the case of the NO-IMI index, suggesting better adaptivity of the RVQ model over PQ. We additionally calculate the percent of empty cells for both indexing structures, with the results shown in Figure 3(right). This further reveals that the NO-IMI significantly reduces a number of empty index cells compared to IMI, especially for DEEP1B data.

We further demonstrate the advantage of the (G)NO-IMI for deep data by measuring the average distance from the data points to the centroids of cells they belong to. This is a reliable indicator of the indexing quality as smaller distances mean that cells represent the actual data distribution better. Furthermore, the small distances mean that the displacements from the points to the closest centroids have small norms, hence they can be encoded more accurately, which results in better reranking. The average distances for two datasets and three systems are shown in Table 1.

For the deep data the NO-IMI reduces average distance by 15% compared to the IMI scheme with the same number of cells. This means that the NO-IMI cell centroids represent data distribution better and we show below that it results in much more precise short-lists. The more powerful Generalized NO-IMI scheme provides even greater improvement by 21%. For SIFT1B, the usage of the NO-IMI allows to reduce average distance by just three percent, which means that in this metric our approach gives very small improvement for this dataset.

Short-list quality. We now compare the retrieval performance of different schemes. In most of experiments we use the commonly used measure $Recall@R$, which is calculated as a rate of queries for which true nearest neighbor is present in a short-list of a length R.

We plot the values of $Recall@R$ for different values of R in Figure 2. For DEEP1B dataset (left), the NO-IMI and the GNO-IMI provide a great improvement in short-list quality over the original IMI. For instance, for the recall level 0.5 the GNO-IMI provides short-lists that are eight time more compact. For SIFT1B dataset (left), the benefit of the NO-IMI is negligible as short-lists are improved only by a very small margin. As the (G)NO-IMI has additional computational overhead comparing to the IMI, such a small improvement does not justify the usage of non-orthogonal

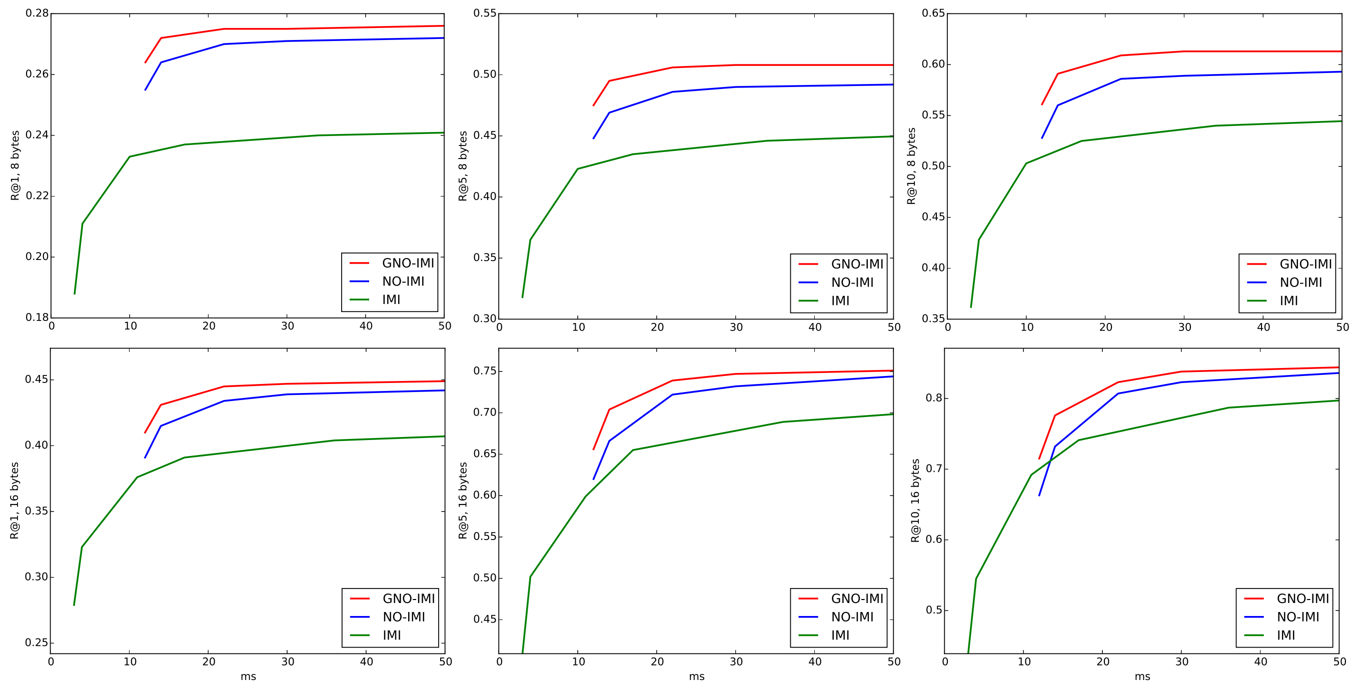

Figure 4. Comparison of the original IMI, NO-IMI and GNO-IMI in terms of recall after reranking, and runtime on the DEEP1B dataset. For all systems we used OPQ with local codebooks to compress database points. With few exceptions, for any given time budget above 11 ms the (G)NO-IMI provides considerably higher recall compared to the IMI-based scheme.

multi-indices with SIFT data.

In this experiment we also compare with IVFADC. This scheme was shown to be less accurate than the IMI on SIFT1B, but it might potentially offer advantage on other data distributions, as it does not perform any space decomposition. For IVFADC we use a considerably larger codebook of size $K = 2^{17}$, but it still produces noticeably less accurate short-lists compared to other schemes.

Reranking and runtime. Finally, we evaluate the (G)NO-IMI in combination with subsequent reranking of candidate lists. Our baseline here is the state-of-the-art Multi-LOPQ system (IMI+reranking), which provides the state-of-the-art on the SIFT1B dataset.

We perform experiments with $M = 8$ and 16 reranking codebooks, which corresponds to 8 and 16 bytes per point respectively. The parameter r in the (G)NO-IMI was set to 32 in all experiments. For both values of M we plot $Recall@1$, $Recall@5$ and $Recall@10$ for different lengths of candidate lists as functions of the corresponding search runtime. The results are summarized in Figure 4. We highlight several observations based on it:

- The timings for the (G)NO-IMI start from 11 milliseconds as this is the cost of the computational overhead. This time Multi-LOPQ allows to extract and to rerank a short-list of approximately 25 thousand points. Figure 2 demonstrates that such a small list for the IMI would contain a true nearest neighbor only for 70% queries, hence such small lists are quite unreliable.
- For any given time more than eleven milliseconds the (G)NO-IMI produces higher recall values comparing to Multi-OPQ. The margin is up to six absolute percent that corresponds to 13% of relative improvement.

Memory consumption. We also provide the amount of total memory consumption for all systems. The NO-IMI requires one additional gigabyte to keep the terms $\langle S_i, T_j \rangle$, $i, j = 1, \ldots, K$. The GNO-IMI also requires one gigabyte to keep K^2 elements of α-matrix. Overall, the amount of additional memory is relatively small. For example, in the setting with 16 bytes, memory consumption equals 23 gigabytes for Multi-LOPQ and 25 gigabytes for the GNO-IMI.

5. Conclusion

We investigated indexing structures for deep descriptors. We have shown that the original Inverted Multi-Index is suboptimal for deep data and have introduced two systems for billion-scale indexing, the Non-Orthogonal Inverted Multi-Index and the Generalized Non-Orthogonal Inverted Multi-Index, which provide more accurate indexing and more precise candidate lists. The advantages of the (G)NO-IMI come at a price of computational overhead, which is small for typical settings.

Acknowledgements

We thank Konstantin Lakhman for his help with data collection. Victor Lempitsky is supported by the Russian Ministry of Science and Education grant RFMEFI57914X0071.

References

[1] A Berg and J Deng and and L Fei-Fei. Large scale visual recognition challenge (ILSVRC). `http://www.image-net.org/challenges/LSVRC/2010/`, 2010. 6

[2] A. Babenko and V. Lempitsky. The inverted multi-index. In *CVPR*, 2012. 1, 3, 6

[3] A. Babenko and V. S. Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, 2014. 2, 3

[4] A. Babenko and V. S. Lempitsky. Aggregating deep convolutional features for image retrieval. In *International Conference on Computer Vision - ICCV*, 2015. 1

[5] A. Babenko and V. S. Lempitsky. The inverted multi-index. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(6):1247–1260, 2015. 1, 6

[6] A. Babenko and V. S. Lempitsky. Tree quantization for large-scale similarity search and classification. In *CVPR*, 2015. 2, 3

[7] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky. Neural codes for image retrieval. In *European Conference on Computer Vision - ECCV*, pages 584–599, 2014. 1, 6

[8] Y. Chen, T. Guan, and C. Wang. Approximate nearest neighbor search by residual vector quantization. In *Sensors*, 2010. 2, 3, 6

[9] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization. Technical report, 2013. 1, 3, 6

[10] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, 2013. 2

[11] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *13th European Conference on Computer Vision (ECCV)*, pages 392–407, 2014. 1

[12] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1), 2011. 1, 2

[13] H. Jegou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: Re-rank with source coding. In *ICASSP*, 2011. 1, 3, 6

[14] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *in Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR 2014)*. IEEE, 2014. 1, 3, 6

[15] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006. 4

[16] M. Norouzi and D. J. Fleet. Cartesian k-means. In *CVPR*, 2013. 2

[17] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops*, pages 512–519, 2014. 1

[18] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Visual instance retrieval with deep convolutional networks. *CoRR*, abs/1412.6574, 2014. 1

[19] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 1, 3

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 6

[21] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, 2014. 2, 3