Polysemous codes

Matthijs Douze, Hervé Jégou and Florent Perronnin

Facebook AI Research

Abstract. This paper considers the problem of approximate nearest neighbor search in the compressed domain. We introduce polysemous codes, which offer both the distance estimation quality of product quantization and the efficient comparison of binary codes with Hamming distance. Their design is inspired by algorithms introduced in the 90's to construct channel-optimized vector quantizers. At search time, this dual interpretation accelerates the search. Most of the indexed vectors are filtered out with Hamming distance, letting only a fraction of the vectors to be ranked with an asymmetric distance estimator.

The method is complementary with a coarse partitioning of the feature space such as the inverted multi-index. This is shown by our experiments performed on several public benchmarks such as the BIGANN dataset comprising one billion vectors, for which we report state-of-the-art results for query times below 0.3 millisecond per core. Last but not least, our approach allows the approximate computation of the k-NN graph associated with the Yahoo Flickr Creative Commons 100M, described by CNN image descriptors, in less than 8 hours on a single machine.

1 Introduction

Nearest neighbor search, or more generally similarity search, has received a sustained attention from different research communities in the last decades. The computer vision community has been especially active on this subject, which is of utmost importance when dealing with very large visual collections.

While early approximate nearest neighbor (ANN) methods were mainly optimising the trade-off between speed and accuracy, many recent works [38,46,28,4] put memory requirements as a central criterion for several reasons. For instance, due to the memory hierarchy, using less memory means using faster memory: disks are slower than the main memory, the main memory is slower than CPU caches, etc. Accessing the memory may be the bottleneck of the search. Therefore algorithms using compact codes are likely to offer a better efficiency than those relying on full vectors. For these reasons, we focus on ANN search with compact codes, which are able to make search in vector sets comprising as much as one billion vectors on a single machine.

We distinguish two separate lines of research in ANN with compact codes. The first class of methods proposes to map the original vectors to the Hamming hypercube [38,52,34,50]. The resulting bit-vectors are efficiently compared with the Hamming distance thanks to optimized low-level processor instructions

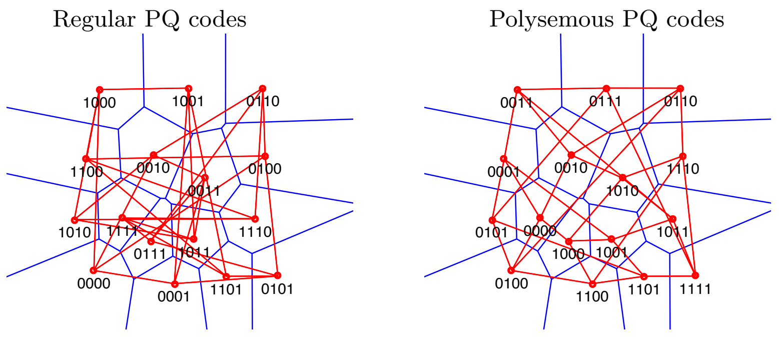

Fig. 1. Polysemous codes are compact representations of vectors that can be compared either with product quantization (222M distance evaluations per second per core for 8-byte codes) or as binary codes (1.19G distances per second). To obtain this property, we optimize the assignment of quantization indexes to bits such that closest centroids have a small Hamming distance. The figure shows k-means centroids (learned on points uniformly drawn in $[0,1] \times [0,1]$) and their corresponding binary representations. Observe how the codes differing by one bit (connected by red segments in the figure) generally correspond to close centroids after our optimization (*right*), which is not the case for standard PQ codes (*left*).

such as `xor` and `popcnt`, available both on CPUs and GPUs. Another increasingly popular approach [28,40,19,5,57,58] is to adopt a quantization point of view to achieve a better distance estimation for a given code size. While these two classes of approaches are often seen as contenders, they both have their advantages and drawbacks. Binary codes offer a faster elementary distance computation and do not need external meta-data once the codes are produced. In contrast, quantization-based approaches achieve better memory/accuracy operating points.

The polysemous codes introduced in this paper offer the best of both worlds. They can be compared either with binary codes, which is especially useful in a filtering step, or with the asymmetric distance estimator of product quantization approaches. The key aspect to attain this dual interpretation is the learning procedure. Our approach is inspired by works on channel-optimized vector quantization [17]. We start by training a product quantizer [28]. We then optimize the so-called *index assignment* of the centroids to binary codes. In other terms, we re-order the numeration of the centroids such that distances between similar centroids are small in the Hamming space, as illustrated in Figure 1.

As a result, our method is almost on par both with quantization-based methods in terms of accuracy and binary methods with respect to search efficiency. When combining this approach with a complementary approach such as the inverted multi-index [4], we outperform the state of the art by a large margin, as shown by our experiments carried out on several large public benchmarks. Interestingly, the high efficiency of our approach offers a scalable solution to the all-neighbor problem, *i.e.*, to compute the k-NN graph, for the large image collection Flickr100M described by 4,096 dimensional vectors.

This paper is organized as follows. After briefly reviewing related works on ANN search with compact codes in Section 2, Section 3 describes the design of polysemous codes. The experiments analyzing our approach and comparing to the state-of-the-art are detailed in Section 4. Finally Section 5 illustrates our method on the task of constructing an image graph on a large scale.

2 Related work: ANN with compact codes

The literature on efficient search with compact codes is vast and we refer the reader to two recent surveys [49,51] for extensive references on this subject. In this section, we present only a few popular approaches.

Compact binary codes. Locality-Sensitive hashing [25,20,12] is a pioneering binary encoding technique. Charikar [12] shows under some assumptions that the Hamming distance is statistically related to the cosine similarity (equivalently the Euclidean distance for normalized vectors). Brute-force comparison of binary hashes has been seen as a viable option for efficient image search with memory constraints [38], which was popularized by subsequent works evidencing the scalability of this approach to million-sized image collections [45]. Additionally, Norouzi and Fleet have proposed an algorithm to speed-up the search in this Hamming space [41]. Many variants have been subsequently proposed, such as spectral hashing [52] or ITQ [21] – see also [43,55,50] for representative works. Related to our work, the k-means hashing method [24] first produces a vector quantizer where the produced codes are compared with the Hamming distance.

Quantization-based codes. Several works have primarily focused on optimizing the trade-off between memory and distance estimation. In particular, it is shown that vector quantizers [28] satisfying the Lloyd conditions [23] offer statistical guarantees on the square Euclidean distance estimator, which is bounded in expectation by the quantizer squared loss. These quantization-based methods include product quantization (PQ) [28] and its optimized versions "optimized product quantization" [19] and "Cartesian k-means" [40].

These approaches are effective for approximate search within large collections of visual descriptors. Subsequent works [5,57,58] have pushed possible memory/efficiency trade-off by adopting a more general point of view, such as "Additive quantization" [5], which provides an excellent approximation and search performance, yet obtained with a much higher computational encoding cost [7]. In between PQ and this general formulation, good trade-offs are achieved by residual quantizers [11,31], which are routinely used in the non-exhaustive PQ variant [28] to reduce the quantization loss by encoding the residual error vector instead of the original vector, but also as a coding strategy on its own [13,39,1].

Non-exhaustive search. The aforementioned methods for ANN search limit the memory usage per indexed vector and provide a distance estimator that is faster to compute than the exact distance. However, the search is still exhaustive in the sense that the query is compared to all database elements. For billion-sized collections, reading the codes in memory is a severe constraint leading to search times in the order of a second, typically. The limitation imposed by this memory bottleneck has led to two-stage approaches [27,28,2,32], in which the feature space is first partitioned through hashing or clustering. Practically, an inverted list storing identifiers and corresponding compact codes is stored for each region. At query time, the distance is estimated only for the codes associated with a subset of regions [37,28,4]. It is also possible to use multiple partitions as in early LSH papers, as done in joint inverted indexing [54]. These solutions however require several indexing structures and are therefore not competitive in terms of memory usage. Various partitioning methods have been proposed for the coarse level [42,4]. In particular, the inverted multi-index uses product quantization both to define the coarse level and for coding residual vectors. This strategy offers state-of-the-art performance when further combined with a re-ranking strategy based on codes [30].

Motivation: Binary codes versus quantization-based approaches. The Hamming distance is significantly faster to evaluate than the distance estimator based on table look-ups involved in quantization methods[1]. From our measurements, the acceleration factor is typically between $4\times$ and $7\times$, depending on the code length. However, binary methods suffer limitations imposed by the Hamming space. First, the number of possible distances is at most $d{+}1$, where d is the binary vector length. This problem is partially solved by asymmetric variants of LSH [16,26,22], whose estimations use compact codes for database vectors but not on the query side. Yet such asymmetric measures require look-ups, like the methods derived from product quantization, and are therefore more expensive to evaluate than the Hamming distance. On the other hand, quantization-based methods offer a better memory/accuracy compromise, which is expected since binarization is a particular case of quantization.

Binary and quantization-based codes have their own advantages and drawbacks. While the literature usually presents binary and quantized-based codes as concurrent methods, the next section introduces a method that benefits from the advantages of both classes of methods.

3 Polysemous codes

We, now, introduce our strategy to take advantage of the fast computation of Hamming distances while offering the estimation accuracy of quantization-based methods. The main idea is to learn a regular product quantizer [28], and then to optimize the assignment of centroid indexes to binary codes such that the Hamming distance approximates the inter-centroid distance. In this section, we

[1] A recent method [3] reduces the scanning by employing a lower-bounding look-up table stored in SIMD registers, however this method is efficient only on long inverted lists, which makes it sub-optimal in the usual setting.

first describe the objective functions optimized to achieve this property, and then describe the optimization algorithm.

Note that for a product quantizer, one typically optimizes separately each of the constituent sub-quantizers. Therefore, in what follows, we have one objective function (and optimization process) per sub-quantizer.

3.1 Objective functions

We consider two objective functions: one that minimizes a loss based on the distance estimator and one that minimizes a ranking loss.

Notation. A quantizer is usually described by its set of centroids. Let $\mathcal{I}$ be the set of centroid indexes: $\mathcal{I} = \{0, 1, \ldots, 2^d - 1\}$ and $d = 8$ if each (sub-)quantizer encodes the original vectors on one byte as is standard practice. Let c_i be the reproduction value associated with centroid i. Let $d : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}^+$ be a distance between centroids, for instance the Euclidean distance. Let $\pi : \mathcal{I} \rightarrow \{0,1\}^d$ denote a bijective function that maps each centroid index to a different vertex of the unit hypercube. Finally let $h : \{0,1\}^d \times \{0,1\}^d \rightarrow \mathbb{R}^+$ be the Hamming distance between two d-dimensional binary representations.

Distance estimator loss. One possible objective if to find the bijective map π such that the distance $d(c_i, c_j)$ between two centroids is approximated by the Hamming distance $h(\pi(i), \pi(j))$ between the two corresponding binary codes:

$$\pi^* = \arg\min_\pi \sum_{i \in \mathcal{I}, j \in \mathcal{I}} \left[h(\pi(i), \pi(j)) - f(d(c_i, c_j))\right]^2 \tag{1}$$

where $f : \mathbb{R} \rightarrow \mathbb{R}$ is a monotonously increasing function that maps the distance $d(c_i, c_j)$ between codewords into a range comparable to Hamming distances. In practice, we choose for f a simple linear mapping. This choice is motivated by the following observations. The Hamming distance between two binary vectors randomly drawn from $\{0,1\}^d$ follows a binomial distribution with mean $d/2$ and variance $d/4$. Assuming that the distribution of distances $d(c_i, c_j)$ can be approximated by a Gaussian distribution – which is a good approximation of the binomial – with mean μ and standard deviation sigma σ, we can force the two distributions to have the same mean and variance. This yields:

$$f(x) = \frac{\sqrt{d}}{2\sigma}(x - \mu) + \frac{d}{2} \tag{2}$$

where μ and σ are measured empirically.

As, in the context of k-NN, it is more important to approximate small distances than large ones, we found out in practice that it is beneficial to weight the distances in the objective function (1). This leads to a weighted objective:

$$\pi^* = \arg\min_\pi \sum_{i \in \mathcal{I}, j \in \mathcal{I}} w(f(d(c_i, c_j))) \left[h(\pi(i), \pi(j)) - f(d(c_i, c_j))\right]^2 . \tag{3}$$

We choose a function $w : \mathbb{R} \to \mathbb{R}$ of the form $w(u) = \alpha^u$ with $\alpha < 1$. In our experiments, we set $\alpha = 1/2$ but we found out that values of α in the range $[0.2, 0.6]$ yielded similar results.

Ranking loss. In the context of k-NN search, we are interested in finding a bijective map π that preserves the ranking of codewords. For this purpose, we adopt an Information Retrieval perspective. Let (i, j) be a pair of codewords such that i is assumed to be a "query" and j is assumed to be "relevant" to i. We will later discuss the choice of (query, relevant) pairs. We take as negatives for query i the codewords k such that $d(c_i, c_j) < d(c_i, c_k)$. The loss for pair (i, j) may be defined as:

$$r_\pi(i,j) = \sum_{k \in \mathcal{I}} \mathbb{1}\left[d(c_i, c_j) < d(c_i, c_k)\right] \mathbb{1}\left[h(\pi(i), \pi(j)) > h(\pi(i), \pi(k))\right] \qquad (4)$$

where $\mathbb{1}[u] = 1$ if u is true, 0 otherwise. It measures how many codewords k are closer to i than j according to the Hamming distance while, i is closer to j than k according to the distance between centroids. We note that the previous loss measures the number of correctly ranked pairs which is closely related to Kendall's tau coefficient.

An issue with the loss $r_\pi(i, j)$ is that it gives the same weight to the top of the list as to the bottom. However, in ranking problems it is desirable to give more weight to errors occurring in the top ranks. Therefore, we do not use directly the loss $r_\pi(i, j)$ for the pair (i, j), but adopt instead a loss that increases sublinearly with $r_\pi(i, j)$. More specifically, we follow [47] and introduce a monotonously decreasing sequence α_i as well as the sequence $\ell_j = \sum_{i=1}^{j} \alpha_i$, which increases sublinearly with j. We define the weighted loss for pair (i, j) as $\ell_{r_\pi(i,j)}$.

A subsequent question is how to choose pairs (i, j). One possibility would be to choose j among the k-NNs of i, in which case we would optimize

$$\pi^* = \arg\min_\pi \sum_{i \in \mathcal{I}} \sum_{j \in k-\mathrm{NN}(i)} \ell_{r_\pi(i,j)}. \qquad (5)$$

An issue with this approach is that it requires choosing an arbitrary length k for the NN list. An alternative is to consider all $j \neq i$ as being potentially "relevant" to i but to downweight the contribution of those j's which are further away from i. In such a case, we optimize

$$\pi^* = \arg\min_\pi \sum_{i \in I, j \in \mathcal{I}} \alpha_{r(i,j)} \ell_{r_\pi(i,j)}, \qquad (6)$$

where we recall that α_i is a decreasing sequence and $r(i, j)$ is the rank of j in the ordered list of neighbors to i:

$$r(i,j) = \sum_{k \in \mathcal{I}} \mathbb{1}\left[d(c_i, c_j) < d(c_i, c_k)\right] \qquad (7)$$

In all our ranking experiments, we use Equation (6) and choose $\alpha_i = 1/i$ (following [47]).

3.2 Optimization

The aforementioned objective functions aim at finding a bijective map π, or equivalently another numeration of the set of PQ centroids, that would assign similar binary codes to neighbouring centroids.

This problem is similar to that of channel optimized vector quantization [56,17,18], for which researchers have designed quantizers such that the corruption of a bit by the channel impacts the reconstruction as little as possible. This is a discrete optimisation problem that can not be relaxed, and for which we can only target a local minimum, as the set of possible bijective maps is huge. In the coding literature, such index assignment problems were first optimised in a greedy manner, for instance by using the binary switching algorithm [56]. Starting from an initial index assignment, at each iteration, this algorithm tests all possible bit swaps (*i.e.*, d), and keeps the one providing the best update of the objective function. As shown by Farvardin [17], this strategy rapidly gets trapped in a poor local minimum. To our knowledge, the best approach to index assignment problems is to employ simulated annealing to carry out the optimization. This choice was shown to be significantly better [17] than previous greedy approaches. The algorithm aims at optimizing a loss $L(\pi)$ that depends on the bijective mapping π defined as a table of size 2^d. It proceeds as follows

```
 1. Initialize
 2.      current solution $\pi := [0, ...., 2^d - 1]$
 3.      temperature $t := t_0$
 4. Iterate $N_{\text{iter}}$ times:
 5.      draw $i, j \in \mathcal{I}, i \neq j$ at random
 6.      $\pi' := \pi$, with entries $i$ and $j$ swapped
 7.      compute the cost update $\Delta C := L(\pi') - L(\pi)$
 8.      if $\Delta C < 0$ or at random with probability $t$:
 9.           accept the new solution: $\pi := \pi'$
10.      $t := t \times t_{\text{decay}}$
```

The algorithm depends on the number of iterations $N_{\text{iter}} = 500,000$, the initial "temperature" $t_0 = 0.7$ and $t_{\text{decay}} = 0.9^{1/500}$, ie. decrease by a factor 0.9 every 500 iterations. Evaluating the distance estimation loss (resp ranking loss) has a complexity in $\mathcal{O}(2^{2d})$ (resp. $\mathcal{O}(2^{3d})$). However, computing the cost update incurred by a swap can be implemented in $\mathcal{O}(2^d)$ (resp. $\mathcal{O}(2^{2d})$).

Figure 2 shows on a set of SIFT descriptors that our optimization is effective: the comparison of the codes used as binary vectors is much more correlated with the true distance after than before the optimization.

3.3 Discussion

Although the optimization algorithm is similar to those previously employed in channel-optimized vector quantization, our objective functions are significantly different to reflect our application scenario. In communications, it is unlikely

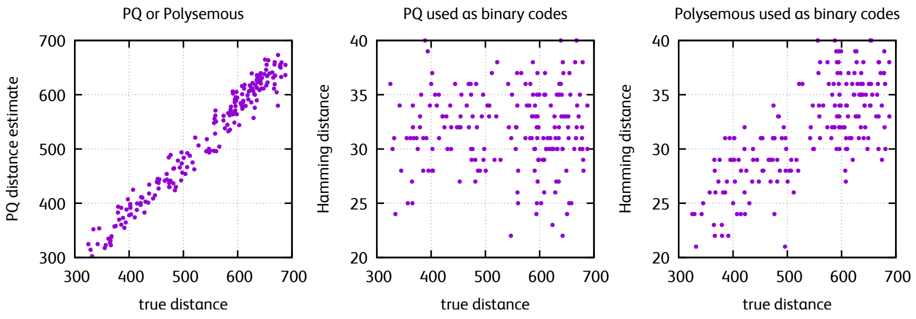

Fig. 2. *Left:* True distances *vs* distance estimates with PQ codes. *Middle:* True distances *vs* Hamming distances before polysemous optimization. *Right:* True distances *vs* Hamming distances after polysemous optimization. The binary comparison with Polysemous is much more discriminative, while offering the same estimation when being interpreted as PQ codes.

that many bit errors occur simultaneously, in particular not on a memoryless channel. Therefore the objective functions employed in communication focus on small Hamming distances. In contrast, for ANN the typical Hamming distances of the neighbors are relatively large.

We point out that, while the proposed binarized PQ codes offer a competitive performance, their accuracy is significantly lower than that of PQ. This suggests a two-step strategy for large-scale search. Given a query, we first filter out the majority of the database items using the fast Hamming distance on the binarized PQ codes. We then evaluate the more costly asymmetric distances for the items whose Hamming distance was below a given threshold τ.

Other strategies could be contemplated for the filtering stage. One such strategy is to measure how many quantization indexes differ for the product quantizer[2]. In other terms, one can filter out vectors if more than a given number of sub-quantizers produce indexes not identical to those of the queries. As shown in the experimental section 4, this method is not as efficient nor precise as the strategy proposed in this section.

Another such strategy would be to use for the filtering stage a binary encoding technique unrelated to PQ, *e.g.*, ITQ. The issue is that it would increase the memory requirements of the method as it would involve storing ITQ codes and PQ codes. In constrat, we only store one polysemous code per database item in the proposed approach – a must if the emphasis is on storage requirements.

4 Experiments

This section gives an analysis and evaluates our polysemous codes. After introducing the evaluation protocol, we analyze our core approach in different aspects.

[2] Formally, this quantity is also called Hamming distance, but measured between vector of indexes and not binary vectors.

Then we show that our approach is compatible with the inverted multi-index (IMI) and give a comparison against the state of the art.

4.1 Evaluation protocol

We analyze and evaluate our approach with standard benchmarks for ANN as well as a new benchmark that we introduce to evaluate the search quality.

SIFT1M is a benchmark [30] of 128-dimensional SIFT descriptors [36]. There are one million vectors in the database, plus 100,000 vectors for training and 10,000 query vectors. This is a relatively small set that we mainly use for parameter analysis.

BIGANN is a large-scale benchmark [30] widely used for ANN search, also made of SIFT descriptors. It comprises one billion database vectors, 100 million training vectors and 10,000 queries.

FYCNN1M and FYCNN90M are introduced to evaluate the quality of the search with more challenging features. We leverage the Yahoo Flickr Creative Commons 100M[3] image collection [44] as follows. In FYCNN90M, we split the dataset into three sets: 90M vectors are to be indexed, 10k vectors serve as queries, 5M vectors are used for training. FYCNN1M uses the same training set and queries, but the indexed set is restricted to the first million images for the purpose of analyzing our method. We extract convolutional neural networks features [35] following the guidelines of [9]: we compute the activations of the 7th layer of AlexNet [33]. This yields 4096-dimensional image descriptors. Prior to indexing we reduce these descriptors to 256D with PCA and subsequently apply a random rotation [9,29].

For all datasets, the accuracy is evaluated by recall@R. This metric measures the fraction of the queries for which the true nearest neighbor is returned within the top R results. All reported times are on a single core of a 2.8GHz machine.

4.2 Analysis of Polysemous codes performance

We first analyze the performance of polysemous codes. Let us introduce notations. We first consider three ways of constructing a product quantizer:

PQ is the baseline: we directly use the code produced by the product quantizer, without any optimization of the index assignment;

PolyD refers to a product quantizer whose index assignment is optimized by minimizing the distance estimator loss introduced in Section 3.1;

PolyR similarly refers to a PQ optimized with the proposed ranking loss.

Once the codebook and index assignment are learned, we consider the following methods to estimate distances based on polysemous codes:

ADC is the regular comparison based on an asymmetric distance estimator [28];

[3] Out of which only 95M are available for download today.

	SIFT1M		FYCNN1M		
	R@1	R@100	R@1	R@100	query (ms)
PQ/disidx	0.071	0.281	0.031	0.284	3.66
PQ/binary	0.036	0.129	0.015	0.124	1.42
PolyD/binary	0.107	0.503	0.027	0.281	1.45
PolyR/binary	0.105	0.467	0.022	0.222	1.45
PQ/dual ($\tau = 55$)	0.312	0.507	0.116	0.522	2.59
PolyD/dual ($\tau = 51$)	0.441	0.987	0.132	0.804	2.53
PolyR/dual ($\tau = 53$)	0.439	0.960	0.130	0.745	2.47
Baseline: LSH [12]	0.114	0.576	0.089	0.643	1.45
Baseline: ITQ [21]	0.135	0.688	0.088	0.654	1.45
Baseline: PQ [28]	0.442	0.997	0.133	0.838	9.01

Table 1. Analysis of polysemous codes (16 bytes/vector). The performance of disidx does not depend on the index assignment. We give the performance of codes when compared in binary, before (PQ/binary) and after (PolyD/binary and PolyR/binary) our optimization. Then we present the results for the proposed polysemous dual strategy, which are almost as accurate as PQ while approaching the speed of binary methods. The Hamming thresholds are adjusted on the training sets so that the Hamming comparison filters out at least 95% of the points. The results are averaged over 5 runs, the sources of randomness being the k-means of the PQ training and the simulated annealing (the standard deviation over runs is always < 0.005). The last 3 rows are baselines provided for reference: LSH, ITQ and PQ. LSH uses a random rotation instead of random projection for better performance [10].

binary refers to the bitwise comparison with the Hamming distance when the codes are regarded as bitvectors, like for binary codes (*e.g.*, ITQ);

disidx counts how many sub-quantizers give different codes (see Section 3.3);

dual refers to the strategy employing both interpretations of polysemous codes: the Hamming codes are used to filter-out the database vectors whose distance to the query is above a threshold τ. The indexed vectors satisfying this test are compared with the asymmetric distance estimator.

Note: Polysemous codes are primarily PQ codes. Therefore the performance of polysemous codes and regular PQ is identical when the comparison is independent from the index assignment, which is the case for ADC and disidx. For instance the combinations PolyD/ADC, PolyR/ADC and PQ/ADC are equivalent both in efficiency and accuracy.

Table 1 details the performance of the aforementioned PQ constructions. First, note that the accuracy of disidx is low, and that it is also relatively slow due to the lack of a dedicated machine instruction. Second, these results show that our index assignment optimization is very effective for improving the quality of the binary comparison. Without this optimization, the binary comparison is ineffective both to rank results (PQ/binary), and to filter (PQ/dual). The ranking loss PolyR is slightly inferior to PolyD, so we adopt the latter in the following.

Figure 3 confirms the relevance of PolyD/dual. It gives the performance achieved by this method for varying Hamming thresholds τ, which parametrizes

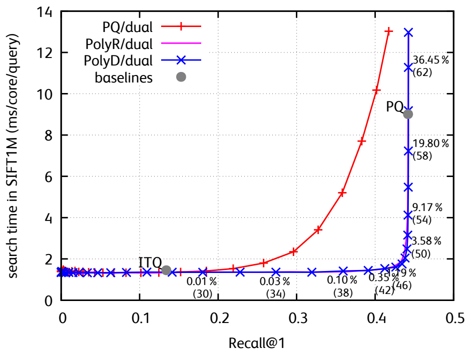

Fig. 3. Impact of the threshold on the dual strategy: Recall@1 vs search speed for the SIFT1M dataset, with 128 bits (16 subquantizers). The operating points for polysemous are parametrized by the Hamming threshold (in parenthesis), which influences the rate of points kept for PQ distance estimation. The trade-offs obtained without polysemous optimization (PQ/dual) and two baselines (ITQ and PQ) are given for reference.

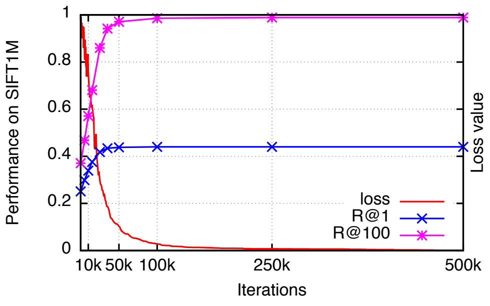

Fig. 4. Performance of polysemous codes (dual, $\tau = 52$, 128 bits) along the iterations for the distance-based objective function described in Section 3 (results with the ranking loss are similar). Note that the initial state (0 iteration) corresponds to a product quantizer not yet optimized with our method.

the trade-off between speed and accuracy. Polysemous codes allow us to make almost no compromise: attaining the quality of PQ/ADC only requires a minor sacrifice in search time compared to binary codes. With threshold $\tau = 54$, 90–95% of the points are filtered out; for $\tau = 42$ this raises to more than 99.5%.

Convergence: Figure 4 shows the performance of the binary filtering as a function of the number of iterations. The algorithm typically converges in a few hundred thousand iterations (1 iteration = 1 test of possible index swaps). For a set of PQ subquantizer with 256 centroids each, this means a few seconds for the distance reconstruction loss PolyR and up to one hour for the ranking loss PolyR.

4.3 Comparison with the state of the art

As mentioned in the related work Section 2, for large datasets the best trade-offs between accuracy, search time and memory are obtained by hybrid methods [28,4] that combine a preliminary space partitioning, typically implemented through clustering, with compact codes learned on residual vectors. That is why we combine our polysemous codes with IMI [4]. This method partitions the space with a product quantizer (the "coarse" partitioning level) and uses PQ to encode residual error vectors. The search proceeds by selecting a few inverted lists at the coarse level, and then use the residual PQ codes to estimate distances for the vectors associated with the selected lists. We further optimize the computation

	code	sizes →	8 bytes			16 bytes		
	K	probes/cap	R@1	R@100	time (ms)	R@1	R@100	time (ms)
IMI [4]	16384^2	–/10k	0.158	0.706	6	0.304	0.740	7
IMI [4]	16384^2	–/30k	0.164	0.813	13	0.328	0.885	13
IMI⋆	16384^2	1024/10k	0.159	0.719	1.57 2.58	0.313	0.753	1.92 2.89
IMI⋆	4096^2	1024/10k	0.125	0.550	0.99 1.23	0.255	0.576	1.16 1.44
IMI⋆	4096^2	16/10k	0.115	0.462	0.50 0.75	0.226	0.479	0.64 0.88
IMI⋆+PolyD+ADC	4096^2	16/10k	0.103	0.332	**0.27** 0.51	0.206	0.397	**0.33** 0.58
IMI⋆	16384^2	1024/30k	0.162	0.796	2.20 3.15	0.330	0.856	2.77 3.75
IMI⋆	4096^2	1024/30k	0.134	0.696	1.35 1.61	0.295	0.755	1.77 2.07
IMI⋆	4096^2	16/30k	0.117	0.505	0.59 0.81	0.238	0.532	0.75 1.01
IMI⋆+PolyD+ADC	4096^2	16/30k	0.106	0.370	0.33 0.56	0.217	0.447	0.38 0.64

Table 2. Comparison against the state of the art on BIGANN (1 billion vectors). We cap both the maximum number of visited lists and number of distance evaluations (column probes/cap). For the timings, using our improved implementation (⋆), the first number is for queries performed in batch mode, while the second corresponds to a single query at a time. Our polysemous method is set to filter out 80% of the codes.

of the lookup tables involved in PQ when multiple lists are probed [6], and use an optimized rotation before encoding with PQ [19].

Building upon this method, we learn polysemous codes for the residual PQ, which allows us to introduce an intermediate stage to filter out most of the list items, hence avoiding most of the distance estimation with PQ. Table 2 gives a comparison against state-of-the-art algorithms on the BIGANN dataset. We report both the timings reported for concurrent methods and our improved re-implementation of IMI. Note already that our system obtains very competitive results compared to the original IMI. Note, in the case where a single query vector is searched at a time, as opposed to batch mode, the coarse quantization becomes 50 to 60% more expensive. Therefore, in the following we use $K = 4096^2$ to target more aggressive operating points by reducing the fixed cost of the coarse quantizer. In this case, the results of PolyD/dual gives a clear improvement compared to IMI⋆ and the state of the art. In particular, with 16 bytes we are able to achieve a recall@1=0.217 in less than 1 ms on one core (0.38 ms in single query mode, 0.64 ms in batch mode). The binary filter divides by about 2× the search time, inducing only a small reduction of the Recall@1 score.

We now compare our method on the more challenging FYCNN90M benchmark, for which a single query amounts to searching an image in a collection containing 90 million images. Figure 5 reports the performance achieved by different methods. First observe that the non-exhaustive methods (*bottom*) are at least 2 orders of magnitude faster than the methods that compare the codes exhaustively (*top*), like ITQ. The former are able to find similar images in a few seconds. Again, our polysemous strategy IMI+PolyD/dual offers a competitive advantage over its competitor IMI. Our method is approximately 1.5× faster for a negligible loss in accuracy.

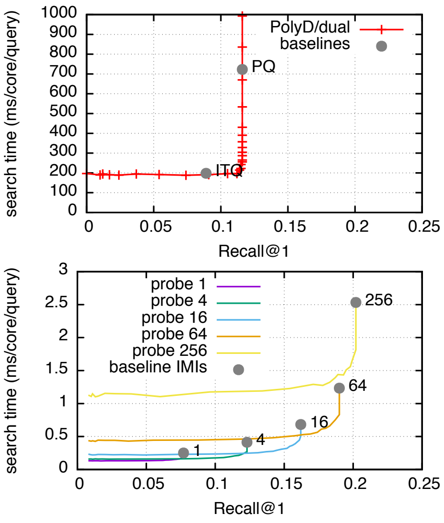

Fig. 5. Performance on the FYCNN90M benchmarks. We use 20 bytes per vector (128 bits for the code and 4 bytes per identifier), *i.e.*, per indexed image. *Above:* For reference we give results obtained by methods that exhaustively compare the query to all vectors indexes based on their codes. As to be expected, the non-exhaustive methods (*below*) achieve much better performance, especially when probing a large number of inverted lists (see "probe 256"). Our proposal IMI+PolyD/dual offers the best trade-off between memory, search time and accuracy by a fair margin.

5 Application: large-scale k-NN image graph

As an application to our fast indexing scheme, we now consider the problem of building the approximate k-NN graph of a very large image collection. For this experiment, we make use of the 95,063,295 images available in the Flickr 100M dataset. As was the case in Section 4, we use 4,096D AlexNet features reduced to 256D with PCA. For the graph construction, we simply compute the k-NN with k=100 for each image in turn. This takes 7h44 using 20 threads of a CPU server. Note that the collection that we consider is significantly larger than the ones considered in previous works [15,48] on kNN graph. Moreover, our approach may be complementary with the method proposed by Dong *et al.* [15].

For visualization purposes, we seek the modes following a random walk technique [14]: we first iteratively compute the stationary distribution of the walk, (*i.e.* the probability of each node to be visited during the random walk) and then consider as modes each local maximum of the stationary probability in the graph. We find on the order of 3,000 such maxima. Figure 6 depicts a sample of these maxima as well as their closest neighbors. We believe that these results are representative of the typical quality of the found neighbors, except that, for privacy reasons, we do not show the numerous modes corresponding to faces, of which we found many including specialized modes of "pairs of persons", "clusters of more than two persons" or "baby faces".

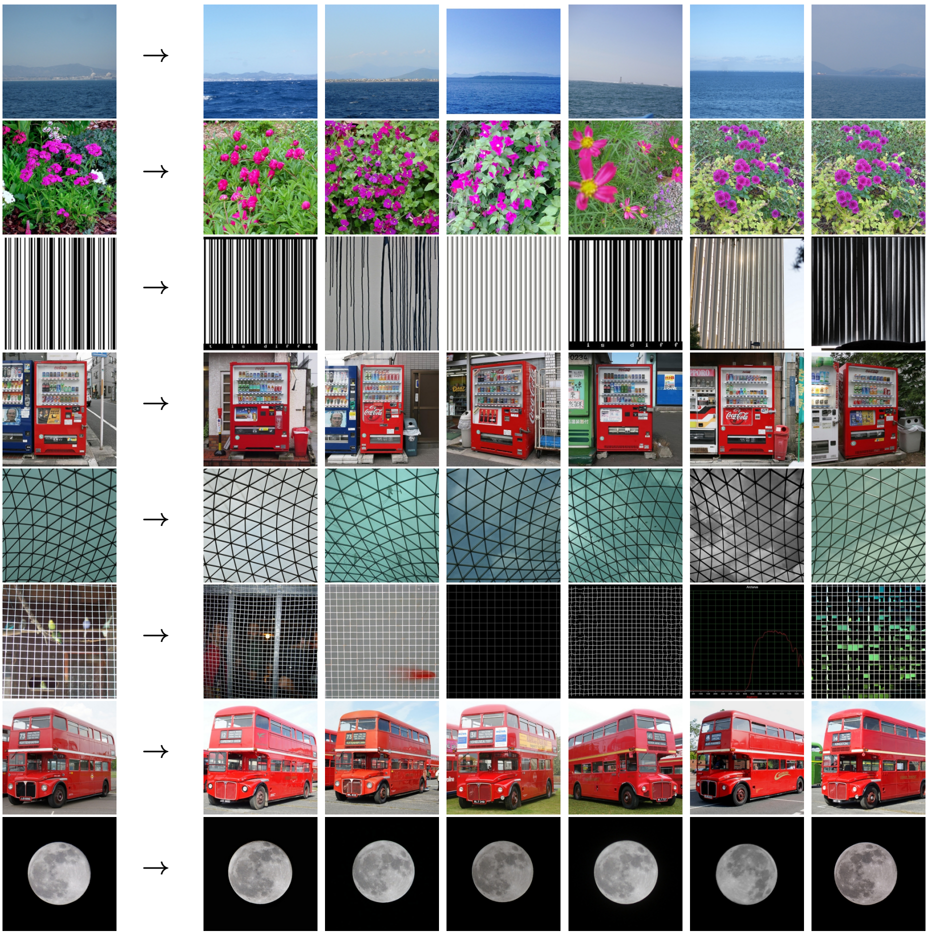

Fig. 6. Examples of image modes and their neighbors in the graph. For each reference image (left), we show the corresponding image neighbors in the kNN graph on its right.

6 Conclusion

In this work, we introduced polysemous codes, *i.e.*, codes that can be interpreted both as binary codes and as product quantization codes. These complementary views are exploited for very large-scale indexing in a simple two-stage process that first involves filtering-out the majority of irrelevant indexes using the fast Hamming distance on binary codes, and then re-ordering the short list of candidates using the more precise but also slower asymmetric distance on PQ codes. This yields a competitive indexing scheme that combines the best of both worlds: its speed is comparable to that of pure binary techniques and its accuracy matches that of PQ.

Acknowledgements. We are very grateful to Armand Joulin and Laurens van de Maaten for providing the Flicrk100M images and their CNN descriptors. Alexandre Sablayrolles had the idea of extending the OPQ method to reduce the number of dimensions.

Additional results

We present results obtained after the ECCV final version submission. We compare operating points with recent papers [53,8]. The only addition to our method is to reduce the dimension of the descriptors before encoding it, within the OPQ [19] transformation. The runtime parameters (multi-probe, Hamming threshold) are tuned automatically to optimize the tradeoff between speed and accuracty.

Table 3 compares with a recent hybrid CPU/GPU method [53], for a given accuracy. It shows that our implementation is slower with a single thread, but is much faster when all cores are used.

Deep1B is a 1-billion CNN vector dataset, introduced by Babenko et al [8]. The vectors are reduced to 96 dimensions, otherwise the dataset statistics are the same as for SIFT1B. We compare against the method that was introduced along with the dataset, allowing 20-byte codes to have a similar memory usage. Table 3 shows that our method is about 5 times faster for a comparable accuracy.

SIFT1B (8 bytes per code)			
method	hardware	1-R@10	time (ms)
Wieschollek [53]	Titan X	*0.35*	*0.15*
OPQ8_64,IMI2x13,PQ8	1 thread	0.349	0.485
	20 threads	0.349	0.035

Deep1B (20 bytes per code)			
method	hardware	1-R@1	time (ms)
Babenko [8]	1 thread	*0.45*	*20*
OPQ20_80,IMI2x14,PQ20	1 thread	0.456	**3.66**

Table 3. Additional results on the datasets SIFT1B and Deep1B [8]. Parameters of the methods: OPQ8_64 = OPQ [19] to 8 blocks, and reduction to 64 D, IMI2x13 means that we use an Inverted multi-index [4] with 2 sub-quantizers, each having 2^{13} centroids. PQ8 refers to a product quantizer with 8-byte codes.

References

1. Ai, L., Yu, J., Wu, Z., He, Y., Guan, T.: Optimized residual vector quantization for efficient approximate nearest neighbor search. Multimedia Systems pp. 1–13 (2015)
2. Andoni, A., Indyk, P., Nguyen, H.L., Razenshteyn, I.: Beyond locality-sensitive hashing. In: SODA. pp. 1018–1028 (2014)
3. André, F., Kermarrec, A.M., le Scouarnec, N.: Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan. In: Proceedings of the International Conference on Very Large DataBases (2015)
4. Babenko, A., Lempitsky, V.: The inverted multi-index. In: CVPR (June 2012)
5. Babenko, A., Lempitsky, V.: Additive quantization for extreme vector compression. In: CVPR (June 2014)
6. Babenko, A., Lempitsky, V.: Improving bilayer product quantization for billion-scale approximate nearest neighbors in high dimensions. arXiv preprint arXiv:1404.1831 (2014)
7. Babenko, A., Lempitsky, V.: Tree quantization for large-scale similarity search and classification. In: CVPR (June 2015)
8. Babenko, A., Lempitsky, V.: Efficient indexing of billion-scale datasets of deep descriptors. In: CVPR (2016)
9. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: ECCV (September 2014)
10. Balu, R., Furon, T., Jégou, H.: Beyond project and sign for distance estimation with binary codes. In: ICASSP (April 2014)
11. Barnes, C.F., Rizvi, S., Nasrabadi, N.: Advances in residual vector quantization: a review. Image Processing, IEEE Transactions on 5(2), 226–262 (1996)
12. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: STOC. pp. 380–388 (May 2002)
13. Chen, Y., Guan, T., Wang, C.: Approximate nearest neighbor search by residual vector quantization. Sensors 10(12), 11259–11273 (2010)
14. Cho, M., Lee, K.M.: Mode-seeking on graphs via random walks. In: CVPR (June 2012)
15. Dong, W., Charikar, M., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: WWW (March 2011)
16. Dong, W., Charikar, M., Li, K.: Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In: SIGIR. pp. 123–130 (July 2008)
17. Farvardin, N.: A study of vector quantization for noisy channels. IEEE Trans. Inform. Theory 36(5), 799–809 (July 1990)
18. Farvardin, N., Vaishampayan, V.: On the performance and complexity of channel-optimized vector quantizers. IEEE Trans. Inform. Theory 37(1), 155–160 (1991)
19. Ge, T., He, K., Ke, Q., Sun, J.: Optimized product quantization for approximate nearest neighbor search. In: CVPR (June 2013)
20. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimension via hashing. In: Proceedings of the International Conference on Very Large DataBases. pp. 518–529 (1999)
21. Gong, Y., Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes. In: CVPR (June 2011)
22. Gordo, A., Perronnin, F.: Asymmetric distances for binary embeddings. In: CVPR (2011)

23. Gray, R.M., Neuhoff, D.L.: Quantization. IEEE Trans. Inform. Theory 44, 2325–2384 (October 1998)
24. He, K., Wen, F., Sun, J.: K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In: CVPR (2013)
25. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: STOC. pp. 604–613 (1998)
26. Jain, M., Jégou, H., Gros, P.: Asymmetric hamming embedding. In: ACM Multimedia (October 2011)
27. Jégou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: ECCV (October 2008)
28. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. PAMI 33(1), 117–128 (January 2011)
29. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: CVPR (June 2010)
30. Jégou, H., Tavenard, R., Douze, M., Amsaleg, L.: Searching in one billion vectors: re-rank with source coding. In: ICASSP (May 2011)
31. Juang, B.H., Gray, A.J.: Multiple stage vector quantization for speech coding. In: ICASSP. vol. 7, pp. 597–600. IEEE (1982)
32. Kalantidis, Y., Avrithis, Y.: Locally optimized product quantization for approximate nearest neighbor search. In: CVPR (June 2014)
33. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (December 2012)
34. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: ICCV (October 2009)
35. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Handwritten digit recognition with a back-propagation network. In: Advances in neural information processing systems 2, NIPS (1989)
36. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV 60(2), 91–110 (2004)
37. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe LSH: Efficient indexing for high-dimensional similarity search. In: Proceedings of the International Conference on Very Large DataBases. pp. 950–961 (2007)
38. Lv, Q., Charikar, M., Li, K.: Image similarity search with compact data structures. In: CIKM. pp. 208–217 (November 2004)
39. Martinez, J., Hoos, H.H., Little, J.J.: Stacked quantizers for compositional vector compression. arXiv preprint arXiv:1411.2173 (2014)
40. Norouzi, M., Fleet, D.: Cartesian k-means. In: CVPR (June 2013)
41. Norouzi, M., Punjani, A., Fleet, D.J.: Fast search in hamming space with multi-index hashing. In: CVPR (2012)
42. Paulevé, L., Jégou, H., Amsaleg, L.: Locality sensitive hashing: a comparison of hash function types and querying mechanisms. Pattern Recognition Letters 31(11), 1348–1358 (August 2010)
43. Raginsky, M., Lazebnik, S.: Locality-sensitive binary codes from shift-invariant kernels. In: NIPS (2010)
44. Thomee, B., Shamma, D.A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., Li, L.J.: The new data and new challenges in multimedia research. arXiv preprint arXiv:1503.01817 (March 2015)
45. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: a large database for non-parametric object and scene recognition. IEEE Trans. PAMI 30(11), 1958–1970 (November 2008)

46. Torralba, A., Fergus, R., Weiss, Y.: Small codes and large databases for recognition. In: CVPR (June 2008)
47. Usunier, N., Buffoni, D., Gallinari, P.: Ranking with ordered weighted pairwise classification. In: ICML (June 2009)
48. Wang, J., Wang, J., Zeng, G., Tu, Z., Gan, R., Li, S.: Scalable k-NN graph construction for visual descriptors. In: CVPR. pp. 1106–1113 (June 2012)
49. Wang, J., Shen, H.T., Song, J., Ji, J.: Hashing for similarity search: A survey. arXiv preprint arXiv:1408.2927 (2014)
50. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for large scale search. IEEE Trans. PAMI 6(12), 1 (2012)
51. Wang, J., Liu, W., Kumar, S., Chang, S.: Learning to hash for indexing big data - A survey. CoRR abs/1509.05472 (2015), `http://arxiv.org/abs/1509.05472`
52. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS (December 2009)
53. Wieschollek, P., Wang, O., Sorkine-Hornung, A., Lensch, H.P.A.: Efficient large-scale approximate nearest neighbor search on the gpu. In: CVPR (June 2016)
54. Xia, Y., He, K., Wen, F., Sun, J.: Joint inverted indexing. In: ICCV (December 2013)
55. Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., Yu, N.: Complementary hashing for approximate nearest neighbor search. In: ICCV (November 2011)
56. Zeger, K., Gersho, A.: Pseudo-gray coding. IEEE Trans. Communications 38(12), 2147–2158 (1990)
57. Zhang, T., Du, C., Wang, J.: Composite quantization for approximate nearest neighbor search. In: ICML. pp. 838–846 (June 2014)
58. Zhang, T., Qi, G.J., Tang, J., Wang, J.: Sparse composite quantization. In: CVPR (June 2015)