Aggregating local descriptors into a compact image representation

Hervé Jégou, INRIA Rennes
Matthijs Douze, INRIA Grenoble
Cordelia Schmid, INRIA Grenoble
Patrick Pérez, Technicolor

Abstract

We address the problem of image search on a very large scale, where three constraints have to be considered jointly: the accuracy of the search, its efficiency, and the memory usage of the representation. We first propose a simple yet efficient way of aggregating local image descriptors into a vector of limited dimension, which can be viewed as a simplification of the Fisher kernel representation. We then show how to jointly optimize the dimension reduction and the indexing algorithm, so that it best preserves the quality of vector comparison. The evaluation shows that our approach significantly outperforms the state of the art: the search accuracy is comparable to the bag-of-features approach for an image representation that fits in 20 bytes. Searching a 10 million image dataset takes about 50ms.

1. Introduction

There are two reasons why the bag-of-features image representation (BOF) is popular for indexation and categorization applications. First, this representation benefits from powerful local descriptors, such as the SIFT descriptor [12] and more recent ones [13, 28, 29]. Second, these vector representations can be compared with standard distances, and subsequently be used by robust classification methods such as support vector machines.

When used for large scale image search, the BOF vector representing the image is advantageously chosen to be highly dimensional [16, 20, 10], up to a million dimensions. In this case, the search efficiency results from the use of inverted lists [23], which speeds up the computation of distances between sparse vectors. However, two factors limit the number of images that can be indexed in practice: the efficiency of the search itself, which becomes prohibitive when considering more than 10 million images, and the memory required to represent an image.

In this paper, we address the problem of searching the most similar images in a very large image database (ten million images or more). We put an emphasis on the joint optimization of three constraints: the search accuracy, its efficiency and the memory usage. The last two are related [24], as search efficiency can be approximated by the amount of memory to be visited. Most similar to our work is the approach of [9], which proposes an approximate nearest neighbor search for BOF vectors. However, this method is limited to small vocabulary sizes, yielding lower search accuracy compared to large ones. Moreover, an image still requires more than a hundred bytes to reproduce the relatively low search accuracy of a low-dimensional BOF vector.

The efficiency problem was partially addressed by the min-Hash approach [2, 3] and the method of Torresani et al. [25]. However, these techniques still require a significant amount of memory per image, and are more relevant in the context of near-duplicate image detection, as their search accuracy is significantly lower than BOF. Some authors address the efficiency and memory constraints by using GIST descriptors [17], and by converting them to compact binary vectors [5, 11, 24, 27]. These approaches are limited by the low degree of invariance of the global GIST descriptor, and none of them jointly fulfills the three aforementioned constraints: [24, 27] still requires an exhaustive search while [11] is memory consuming due to the redundancy of the LSH algorithm. In contrast, our approach obtains a significantly higher accuracy with, typically, a 20-byte representation. This is obtained by optimizing:

1. the representation, i.e., how to aggregate local image descriptors into a vector representation;
2. the dimensionality reduction of these vectors;
3. the indexing algorithm.

These steps are closely related: representing an image by a high-dimensional vector usually provides better exhaustive search results than with a low-dimensional one. However, high dimensional vectors are more difficult to index efficiently. In contrast, a low dimensional vector is more easily indexed, but its discriminative power is lower and might not be sufficient for recognizing objects or scenes.

Our first contribution consists in proposing a representation that provides excellent search accuracy with a reasonable vector dimensionality, as we know that the vector will be indexed subsequently. We propose a descriptor, derived from both BOF and Fisher kernel [18], that aggregates SIFT descriptors and produces a compact representation. It

is termed VLAD (vector of locally aggregated descriptors). Experimental results demonstrate that VLAD significantly outperforms BOF for the same size. It is cheaper to compute and its dimensionality can be reduced to a few hundreds components by principal component analysis (PCA) without noticeably impacting its accuracy.

As a second contribution, we show the advantage of jointly optimizing the trade-off between the dimensionality reduction and the indexation algorithm. We consider in particular the recent indexing method of [7], as we can directly compare the error induced by PCA with the error resulting from the indexation, due to the approximate reconstruction of the vector from its encoded index.

After presenting two image vector representations that inspired ours, BOF and the Fisher kernel [18], we introduce our descriptor aggregation method in Section 2. The joint optimization of dimensionality reduction and indexing is presented in Section 3. Experimental results demonstrate the performance of our approach in section 4: we show that the performance of BOF is attained with an image representation of about 20 bytes. This is a significant improvement over the state-of-the-art [9], both in terms of memory usage, search accuracy and efficiency.

2. Image vector representation

In this section, we briefly review two popular approaches that produce a vector representation of an image from a set of local descriptors. We then propose our method to aggregate local descriptors.

2.1. Bag of features

The BOF representation groups local descriptors. It requires the definition of a codebook of k "visual words" usually obtained by k-means clustering. Each local descriptor of dimension d from an image is assigned to the closest centroid. The BOF representation is obtained as the histogram of the assignment of all image descriptors to visual words. Therefore, it produces a k-dimensional vector, which is subsequently normalized. There are several variations on how to normalize the histogram. When seen as an empirical distribution, the BOF vector is normalized using the Manhattan distance. Another common choice consists in using Euclidean normalization. The vector components are then weighted by *idf* (inverse document frequency) terms. Several weighting schemes have been proposed [16, 23]. In the following, we perform L_2 normalization of histograms and use the *idf* calculation of [23].

Several variations have been proposed to improve the quality of this representation. One of the most popular [21, 26] consists in using soft quantization techniques instead of a k-means.

2.2. Fisher kernel

The Fisher kernel [6] is a powerful tool to transform an incoming variable-size set of independent samples into a fixed size vector representation, assuming that the samples follow a parametric generative model estimated on a training set. This description vector is the gradient of the sample's likelihood with respect to the parameters of this distribution, scaled by the inverse square root of the Fisher information matrix. It gives the direction in parameter space into which the learnt distribution should be modified to better fit the observed data. It has been shown that discriminative classifiers can be learned in this new representation space.

Perronnin et al. [18] applied Fisher kernel in the context of image classification. They model the visual words with a Gaussian mixture model (GMM), restricted to diagonal variance matrices for each of the k components of the mixture. Deriving a diagonal approximation of the Fisher matrix of a GMM, they obtain a $(2d+1) \times k - 1$ dimensional vector representation of an image feature set, or $d \times k$-dimensional when considering only the components associated with either the means or the variances of the GMM. In comparison with the BOF representation, fewer visual words are required by this more sophisticated representation.

2.3. VLAD: vector of locally aggregated descriptors

We propose a vector representation of an image which aggregates descriptors based on a locality criterion in feature space. It can be seen as a simplification of the Fisher kernel. As for BOF, we first learn a codebook $\mathcal{C} = \{c_1, ...c_k\}$ of k visual words with k-means. Each local descriptor x is associated to its nearest visual word $c_i = \text{NN}(x)$. The idea of the VLAD descriptor is to accumulate, for each visual word c_i, the differences $x - c_i$ of the vectors x assigned to c_i. This characterizes the distribution of the vectors with respect to the center.

Assuming the local descriptor to be d-dimensional, the dimension D of our representation is $D = k \times d$. In the following, we represent the descriptor by $v_{i,j}$, where the indices $i = 1 \dots k$ and $j = 1 \dots d$ respectively index the visual word and the local descriptor component. Hence, a component of v is obtained as a sum over all the image descriptors:

$$v_{i,j} = \sum_{x \text{ such that } \text{NN}(x)=c_i} x_j - c_{i,j} \qquad (1)$$

where x_j and $c_{i,j}$ respectively denote the j^{th} component of the descriptor x considered and of its corresponding visual word c_i. The vector v is subsequently L_2-normalized by $v := v/||v||_2$.

Experimental results show that excellent results can be obtained even with a relatively small number of visual

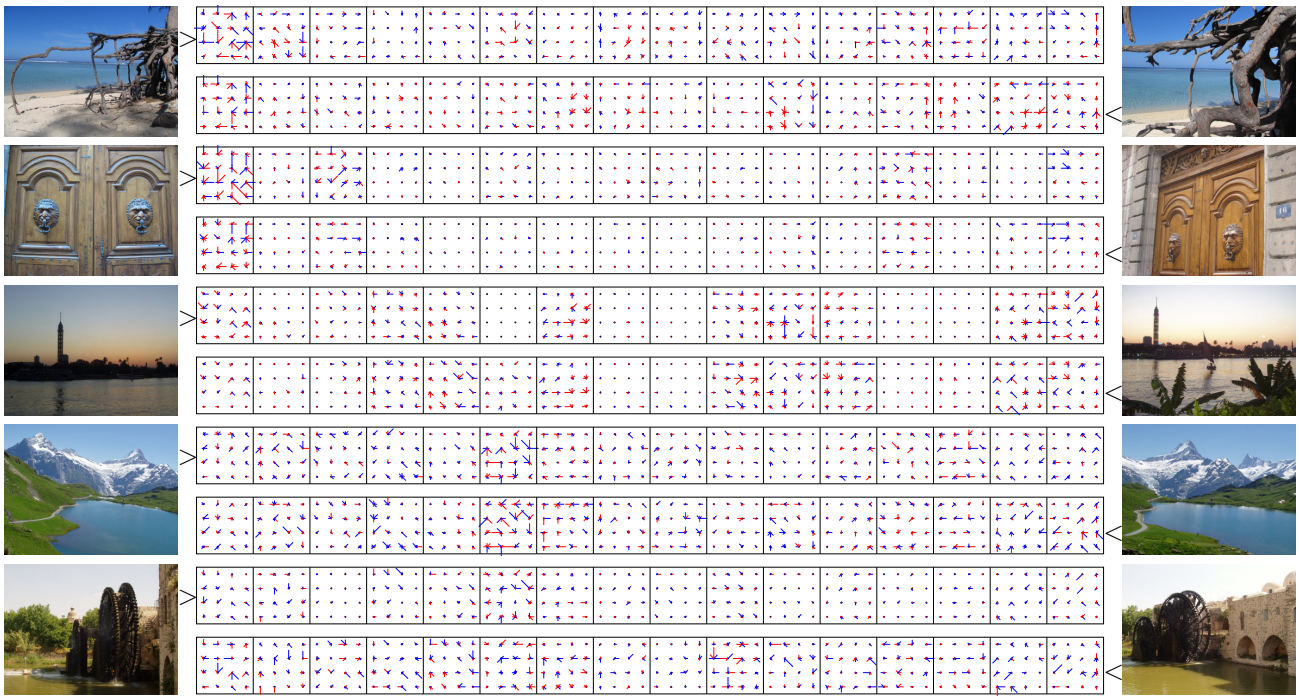

Figure 1. Images and corresponding VLAD descriptors, for k=16 centroids (D=16×128). The components of the descriptor are represented like SIFT, with negative components (see Equation 1) in red.

words k: we consider values ranging from k=16 to k=256.

Figure 1 depicts the VLAD representations associated with a few images, when aggregating 128-dimensional SIFT descriptors. The components of our descriptor map to components of SIFT descriptors. Therefore we adopt the usual 4×4 spatial grid representation of oriented gradients for each $v_{i=1..k}$. We have accumulated the descriptors in 16 of them, one per visual word. In contrast to SIFT descriptors, a component may be positive or negative, due to the difference in Equation 1.

One can observe that the descriptors are relatively sparse (few values have a significant energy) and very structured: most high descriptor values are located in the same cluster, and the geometrical structure of SIFT descriptors is observable. Intuitively and as shown later, a principal component analysis is likely to capture this structure. For sufficiently similar images, the closeness of the descriptors is obvious.

3. From vectors to codes

This section addresses the problem of coding an image vector. Given a D-dimensional input vector, we want to produce a code of B bits encoding the image representation, such that the nearest neighbors of a (non-encoded) query vector can be efficiently searched in a set of n encoded database vectors.

We handle this problem in two steps, that must be optimized jointly: 1) a projection that reduces the dimensionality of the vector and 2) a quantization used to index the resulting vectors. For this purpose, we consider the recent approximate nearest neighbor search method of [7], which is briefly described in the next section. We will show the importance of the joint optimization by measuring the mean squared Euclidean error generated by each step.

3.1. Approximate nearest neighbor

Approximate nearest neighbors search methods [4, 11, 15, 24, 27] are required to handle large databases in computer vision applications [22]. One of the most popular techniques is Euclidean Locality-Sensitive Hashing [4], which has been extended in [11] to arbitrary metrics. However, these approaches and the one of [15] are memory consuming, as several hash tables or trees are required. The method of [27], which embeds the vector into a binary space, better satisfies the memory constraint. It is, however, significantly outperformed in terms of the trade-off between memory and accuracy by the product quantization-based approximate search method of [7]. In the following, we use this method, as it offers better accuracy and because the search algorithm provides an explicit approximation of the indexed vectors. This allows us to compare the vector approximations introduced by the dimensionality reduction and the quantization. We use the asymmetric distance computation (ADC) variant of this approach, which only encodes the vectors of the database, but not the query vector. This method is summarized in the following.

ADC approach. Let $x \in \Re^D$ be a query vector and $\mathcal{Y} = \{y_1, \ldots, y_n\}$ a set of vectors in which we want to find the nearest neighbor $\mathrm{NN}(x)$ of x. The ADC approach consists in encoding each vector y_i by a quantized version $c_i = q(y_i) \in \Re^D$. For a quantizer $q(.)$ with k centroids, the vector is encoded by $\log_2(k)$ bits, k being a power of 2. Finding the a nearest neighbors $\mathrm{NN}_a(x)$ of x simply consists in computing

$$\mathrm{NN}_a(x) = a\text{-}\arg\min_i ||x - q(y_i)||^2. \quad (2)$$

Note that, in contrast with the embedding method of [27], the query x is not converted to a code: there is no approximation error on the query side.

To get a good vector approximation, k should be large ($k = 2^{64}$ for a 64 bit code). For such large values of k, learning a k-means codebook as well as assignment to the centroids is not tractable. Our solution is to use a product quantization method which defines the quantizer without explicitly enumerating its centroids. A vector x is first split into m subvectors x^1, ... x^m of equal length D/m. A product quantizer is then defined as function

$$q(x) = \big(q_1(x^1), ..., q_m(x^m)\big), \quad (3)$$

which maps the input vector x to a tuple of indices by separately quantizing the subvectors. Each individual quantizer $q_j(.)$ has k_s reproduction values learned by k-means. To limit the assignment complexity $\mathcal{O}(m \times k_\mathrm{s})$, k_s is a small value (e.g., k_s=256). However, the set k of centroids induced by the product quantizer $q(.)$ is large: $k = (k_\mathrm{s})^m$.

The squared distances in Equation 2 are computed using the decomposition

$$||x - q(y_i)||^2 = \sum_{j=1,\ldots,m} ||x^j - q_j(y_i^j)||^2, \quad (4)$$

where y_i^j is the j^{th} subvector of y_i. The square distances in this summation are read from look-up tables computed, prior to the search, between each subvector x^j and the k_s centroids associated with the corresponding quantizer q_j. The generation of the tables is of complexity $\mathcal{O}(D \times k_\mathrm{s})$. When $k_\mathrm{s} \ll n$, this complexity is negligible compared with the summation cost of $\mathcal{O}(D \times n)$ in Equation 2.

This quantization method was chosen because of its excellent performance, but also because it represents the indexation as a vector approximation: a database vector y_i can be decomposed as

$$y_i = q(y_i) + \varepsilon_\mathrm{q}(y_i), \quad (5)$$

where $q(y_i)$ is the centroid associated with y_i and $\varepsilon_\mathrm{q}(y_i)$ the error vector generated by the quantizer.

Notation: ADC $m \times b_\mathrm{s}$ refers to the method when using m subvectors and b_s bits to encode each subvector ($b_\mathrm{s} = \log_2 k_\mathrm{s}$). The total number of bits B used to encode a vector is then given by $B = m\, b_\mathrm{s}$.

3.2. Indexation-aware dimensionality reduction

Dimensionality reduction is an important step in approximate nearest neighbor search, as it impacts the subsequent indexation method. In this section, for the ADC approach[1], we express the compromise between this operation and the indexing scheme using a single quality measure: the approximation error. For the sake of presentation, we assume that the mean of the vectors is the null vector. This is approximately the case for VLAD vectors.

Principal component analysis (PCA) is a standard tool [1] for dimensionality reduction: the eigenvectors associated with the D' most energetic eigenvalues of the empirical vector covariance matrix are used to define a matrix M mapping a vector $x \in \Re^D$ to a vector $x' = Mx \in \Re^{D'}$. Matrix M is the $D' \times D$ upper part of an orthogonal matrix. This dimensionality reduction can also be interpreted in the initial space as a projection. In that case, x is approximated by

$$x_\mathrm{p} = x - \varepsilon_\mathrm{p}(x) \quad (6)$$

where the error vector $\varepsilon_\mathrm{p}(x)$ lies in the null space of M. The vector x_p is related to x' by the pseudo-inverse of M, which is the transpose of M in this case. Therefore, the projection is $x_\mathrm{p} = M^\top M x$. For the purpose of indexing, the vector x' is subsequently encoded as $q(x')$ using the ADC approach, which can also be interpreted in the original D-dimensional space as the approximation[2]

$$q(x_\mathrm{p}) = x - \varepsilon_\mathrm{p}(x) - \varepsilon_\mathrm{q}(x_\mathrm{p}) \quad (7)$$

where $\varepsilon_\mathrm{p}(x) \in \mathrm{Null}(M)$ and $\varepsilon_\mathrm{q}(x_\mathrm{p}) \in \mathrm{Null}(M)^\perp$ (because the ADC quantizer is learned in the principal subspace) are orthogonal. At this point, we make two observations:

1. Due to the PCA, the variance of the different components of x' is not balanced. Therefore the ADC structure, which regularly subdivides the space, quantizes the first principal components more coarsely in comparison with the last components that are selected. This allocation introduces a bottleneck on the first components with respect to the quantization error.
2. There is a trade-off on the number of dimension D' to be retained by the PCA. If D' is large, the projection error vector $\varepsilon_\mathrm{p}(x)$ is of limited magnitude, but a large quantization error $\varepsilon_\mathrm{q}(x_\mathrm{p})$ is introduced. On the opposite, keeping a small number of components leads to a high projection error and a low quantization error.

Balancing the components' variance. In [27], this issue was addressed by allocating different numbers of bits to the different components. The ADC method does not have this

[1] Note that [7] did not propose any dimensionality reduction.

[2] For the sake of conciseness, the quantities $M^T q(x')$ and $M^T \varepsilon_\mathrm{q}(x')$ are simplified to $q(x_\mathrm{p})$ and $\varepsilon_\mathrm{q}(x_\mathrm{p})$ respectively.

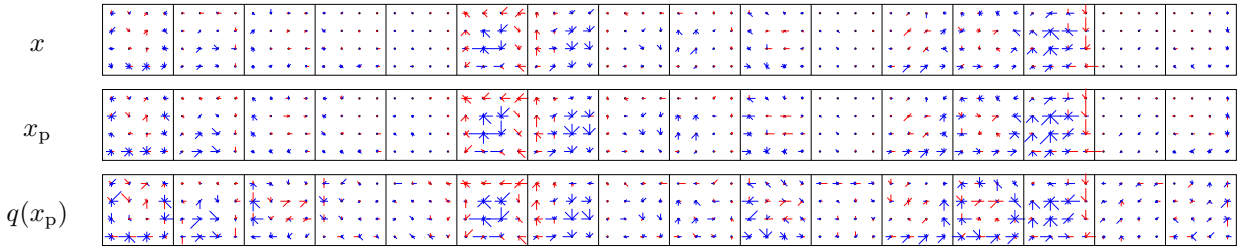

Figure 2. Effect of the encoding steps on the descriptor. *Top:* VLAD vector x for k=16 (D=2048). *Middle:* vector x_{p} altered by the projection onto the PCA subspace (D'=128). *Bottom:* vector $q(x_{\mathrm{p}})$ after indexing by ADC 16×8 (16-bytes code).

flexibility. Therefore, we address the problem by performing an orthogonal transformation after the PCA. In other terms, given the dimension D' and X a vector to index, we want to find an orthogonal matrix Q such that the components of the transformed vector $X'' = QX' = QMX$ have equal variances. It would be optimal to find the matrix $\tilde{Q}$ minimizing the expected quantization error introduced by the ADC method:

$$\tilde{Q} = \arg\min_{Q} \mathbb{E}_X \left[||\varepsilon_{\mathrm{q}}(QMX)||^2 \right]. \tag{8}$$

However, this optimization problem is not tractable, as the objective function requires to learn the product quantizer $q(.)$ of the ADC structure at each iteration. Finding a matrix Q satisfying the simplified balancing objective is done by choosing it in the form of a Householder matrix

$$Q = I - 2vv^\top, \tag{9}$$

with the optimization performed on the D' components of v. A simple alternative is to avoid this optimization by using, for Q, a $D' \times D'$ random orthogonal matrix. For high dimensional vectors, this choice is an acceptable solution with respect to our balancing criterion. This will be confirmed by our experiments in Section 4, which show that, for VLAD descriptors, both choices are equivalent.

Joint optimization of reduction/indexing. Let us now consider the second problem, i.e., optimizing the dimension D', having fixed a constraint on the number of bits B used to represent the D-dimensional VLAD vector x, for instance B=128 (16 bytes). The square Euclidean distance between the reproduction value and x is the sum of the errors $||\varepsilon_{\mathrm{p}}(x)||^2$ and $||\varepsilon_{\mathrm{q}}(x_{\mathrm{p}})||^2$, which both depend on the selected D'. The mean square error $e(D')$ is empirically measured on a learning vector set $\mathcal{L}$ as

$$e(D') = e_p(D') + e_q(D') \tag{10}$$

$$= \frac{1}{\mathrm{card}(\mathcal{L})} \sum_{x \in \mathcal{L}} ||\varepsilon_{\mathrm{p}}(x)||^2 + ||\varepsilon_{\mathrm{q}}(x_{\mathrm{p}})||^2. \tag{11}$$

This gives us an objective criterion to optimize directly the dimensionality, which is obtained by finding on the learning set the value of D' minimizing this criterion. For VLAD vectors (k=16, D=2048) generated from SIFT descriptors, we obtain the following average projection, quantization and total mean square errors:

D'	$e_p(D')$	$e_q(D')$	$e(D')$
32	0.0632	0.0164	0.0796
48	0.0508	0.0248	0.0757
64	0.0434	0.0321	**0.0755**
80	0.0386	0.0458	0.0844

It is important to compute the measures several times (here averaged over 10 runs), as the ADC structure depends on the local optimum found by k-means, and leads to variable quantization errors. The optimization selects D'=64 for k=16 and B=128. We will confirm in the experimental section that this value represents an excellent trade-off in terms of search results.

Remarks.

- The choice of D' is constrained by the structure of ADC, which requires that D' is a multiple of m. For instance, by keeping D'=64 eigenvalues, the valid set of values for m is {1,2,4,8,16,32,64}.
- The optimization is solely based on the mean squared error quantization criterion. For this reason, it is not clear how our framework could be extended to another indexation method, such as LSH [4], which does not provide an explicit approximation vector.
- We apply the projection $Q \times M$ before the L_2 normalization of the aggregated representation (see Subsection 2.3). This brings a marginal improvement in terms of image search accuracy.

The impact of dimensionality reduction and indexation based on ADC is illustrated by the VLAD pictorial representation introduced in Section 2. We can present the projected and quantized VLAD in this form, as both PCA projection and ADC provide a way of reconstructing the projected/quantized vector. Figure 2 illustrates how each of these operations impacts our representation. One can see that the vector is only slightly altered, even for a compact representation of B=16 bytes.

Descriptor	k	D	Holidays (mAP)				UKB (score/4)			
			D	$\rightarrow D'$=128	$\rightarrow D'$=64	$\rightarrow D'$=32	D	$\rightarrow D'$=128	$\rightarrow D'$=64	$\rightarrow D'$=32
BOF	1 000	1 000	0.401	0.444	0.434	0.408	2.86	2.99	2.91	2.77
	20 000	20 000	0.404	0.452	0.445	0.416	2.87	2.95	2.90	2.78
Fisher (μ)	16	2 048	0.497	0.490	0.475	0.452	3.07	3.05	2.98	2.83
	64	8 192	0.495	0.492	0.464	0.424	3.09	3.09	2.98	2.75
VLAD	16	2 048	0.496	0.495	0.494	0.451	3.07	3.05	2.99	2.82
	64	8 192	0.526	0.510	0.477	0.421	3.17	3.15	3.03	2.79

Table 1. Performance comparison of BOF, Fisher and VLAD representations, before and after dimension reduction: the performance is given for the full D-dimensional descriptor, and after a dimensionality reduction to D'=128, 64 and 32 components. Note that for UKB, the best score reported by Nister and Stewénius is 3.19, for a 1M vocabulary tree [16] learned on an independent dataset.

4. Experiments

In this section, we first evaluate our VLAD descriptor and the joint dimensionality reduction/indexing approach. We then provide a comparison with the state of the art and measure the accuracy and efficiency of the search on 10 million images.

4.1. Evaluation datasets and local descriptors

To extract local features, we have used the experimental setup of [9] and the feature extraction software available online[3]. More precisely, the regions of interest are extracted using the Hessian affine-invariant region detector [14] and described by the SIFT descriptor [12]. We have used an independent image set for all the learning stages. The evaluation is performed on three datasets:

- The INRIA Holidays dataset [8]. This is a collection of 1491 holiday images, 500 of them being used as queries. The accuracy is measured by the mean Average Precision (mAP).
- The University of Kentucky Benchmark (UKB). This set comprises images of 2550 objects, each of which is represented by 4 images. The most commonly used evaluation metric for this dataset counts the average number of relevant images (including the query itself) that are ranked in the first four positions when searching the 10 200 images.
- To evaluate the behavior of our method on a large scale, we have downloaded 10M images from Flickr. The Holidays dataset is merged with this set, as in [9], to provide ground truth matches.

4.2. Image vector representations

Table 1 compares the different local aggregation methods described in Section 2: BOF, Fisher kernel, and our VLAD aggregation technique. These representations are all parametrized by a single parameter k. It corresponds to the number of centroids for BOF and VLAD, and to the number of mixture components in the Fisher kernel representation. For Fisher, we have only used the components associated with the mean vectors (μ), as we observed that, although the variance components improve the results, they provide comparable results after dimensionality reduction to the same D'.

The evaluation is performed without the indexing scheme at this stage. Here, we put an emphasis on the performance obtained after dimensionality reduction, as these vectors will be indexed afterwards. Despite its simplicity, the VLAD descriptor equals or outperforms the Fisher kernel on both Holidays and UKB, and significantly outperforms BOF. Note that surprisingly the dimension reduction improves the accuracy for BOF. The scores of BOF are slightly different from those reported in [9], because we use Euclidean distances to compare representations instead of the cosine measure. These choices are not strictly equivalent, because *idf* weights are applied after the L_2 normalization.

Higher dimensional representations, which usually provide better accuracy, suffer more from the dimensionality reduction. This is especially true for Fisher kernel and VLAD: for D'=32, using only 16 centroids/mixtures is significantly better than larger values of k. On average, VLAD outperforms the Fisher kernel representation. For only D'=64 dimensions VLAD attains an excellent accuracy of mAP=0.494 on Holidays.

Note that in a very recent work [19], Perronnin et al. further improved the Fisher kernel in several ways, obtaining mAP=0.593 on Holidays with a mixture of k=64 Gaussians. Hopefully, our VLAD representation would also benefit from the proposed techniques.

4.3. Reduction and indexation

Balancing the variances. Table 2 compares the search performance obtained by applying the ADC indexing after 1) PCA dimension reduction, 2) PCA followed by an orthogonal transformation optimizing the variance balancing criterion (see Subsection 3.2), and 3) PCA followed by a random orthogonal transformation. The need for a rotation is clear. However, using a random one provides results

[3] http://lear.inrialpes.fr/people/jegou/data.php

Method	mAP
No transformation	0.445
Balancing optimization	0.457
Random orthogonal transformation	0.457

Table 2. Comparison of different orthogonal transformation matrices, with VLAD, k=16, D'=64, ADC 16×8. These measures are averaged over 10 runs on the Holidays dataset.

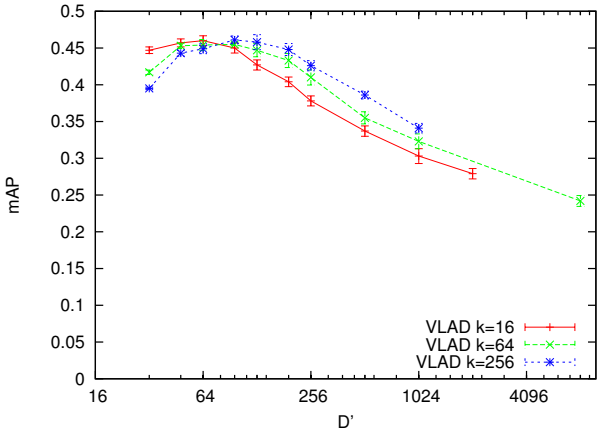

Figure 3. Search accuracy on Holidays with respect to reduction to different dimensions D' with ADC 16×8. Experiments are averaged over 5 learning runs. The error bars represent the standard deviations over those runs.

comparable to those obtained by optimizing the matrix. Our explanation is that the random rotation sufficiently balances the energy.

Choice of the projection subspace dimension. For a fixed image representation with a vector of length D and a fixed number B of bits to encode this vector, Figure 3 confirms the analysis of Section 3: there is an important trade-off on D'. The optimum limits the loss introduced by the projection and the quantization step. The best choice of D' corresponds to the one found by our optimization procedure. For instance, for VLAD with k=16, we obtain the same optimum (D'=64) as the one estimated in Section 3.2 based on our objective error criterion.

4.4. Comparison with the state of the art

Our objectives are comparable to those of [9] in terms of memory usage and desired degree of invariance (rotation/scale invariance). Table 3 and Figure 4 compare the accuracies obtained by [9] and our approach on the benchmarks Holidays and UKB. Our approach obtains a comparable search quality with at least an order of magnitude less memory. Equivalently, for the same memory usage, our method is significantly more precise.

Figure 4 also illustrates the trade-off between search quality and memory usage. An interesting observation is

Method	bytes	UKB	Holidays
BOF, k=20,000 (from [9])	10.364	2.92	0.446
miniBOF [9]	20	2.07	0.255
	80	2.72	0.403
	160	2.83	0.426
VLAD, k=16, ADC 16×8	16	2.88	0.460
VLAD, k=64, ADC 32×10	40	3.10	0.495

Table 3. Comparison with the state of the art on UKB (score/4) and Holidays (mAP). D'=64 for k=16 and D'=96 for k=64.

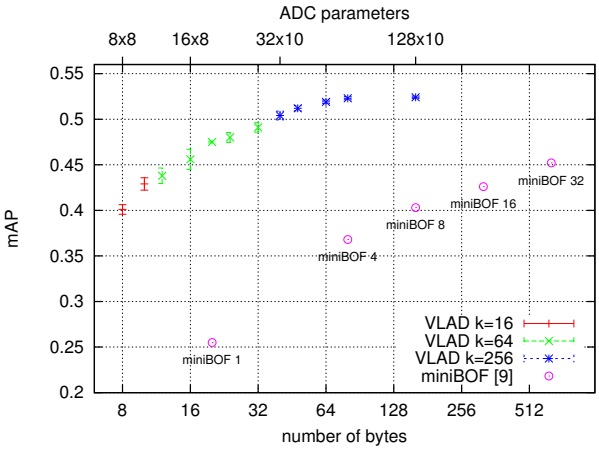

Figure 4. mAP for search on Holidays. For a given number of bytes, the optimal choice of D' is computed and only the results of the best codebook size (k=16, 64 or 256) are reported. The error bars represent the standard deviation over 5 runs. miniBOF results of [9] are reported for reference.

that the choice of the number of centroids k depends on the number of bits B chosen to represent the image. It shows that we attain competitive accuracy, with respect to BOF, using only 16 bytes. Note that small (resp. large) values of k should be associated with small (resp. large) values of B: large ones are more impacted by dimensionality reduction.

4.5. Large scale experiments

Figure 5 evaluates our approach on a large scale (up to 10 million images). It gives the mAP performance as a function of the dataset size for the full VLAD vector (k=64, D=8192), after it is reduced by PCA to D'=64 dimensions, and after the reduced vector is indexed with ADC 16×8 in 16 bytes of memory.

For this experiment, we have also used IVFADC, the non exhaustive search variant of ADC proposed in [7]. IVFADC combines ADC with an inverted file to restrict the search to a subset of vectors. Consequently, it stores the image identifiers explicitly (4 bytes per image), and therefore requires 20 bytes of memory with the selected parameters. It gives comparable results, depending on the operating point (better and more efficient for very large sets).

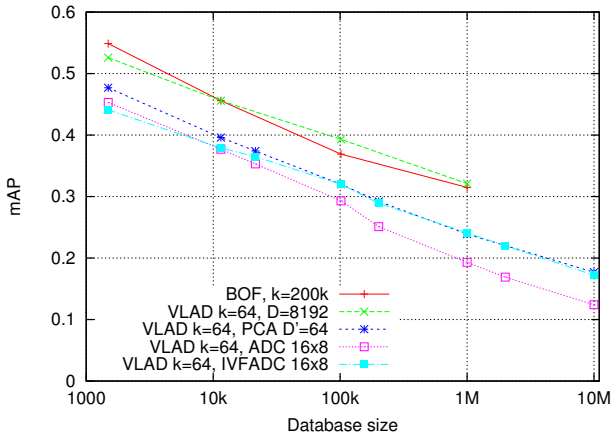

Figure 5. Search accuracy as a function of the database size.

Overall, our results are significantly better than those reported in [9], where a mAP of 0.066 is reported for 1 million images and a 20-bytes representation, versus mAP=0.193 or 0.241 in our case, depending on the ADC variant.

Timings: Timing experiments have been performed on a single processor core. Searching our 10 million dataset for VLAD vectors with k=64 reduced to D'=64 dimensions takes 7.2 s when Euclidean distances are exhaustively computed. Searching with ADC 16×8 takes 0.716 s, while using the IVFADC 16×8 variant takes 0.046 s. This timing is better than the one reported in [9], and this for a significantly better accuracy.

Acknowledgements

We would like to thank QUAERO and the ANR projects ICOS-HD and GAIA for their financial support. Many thanks to Florent Perronnin for insightful discussions.

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[2] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, June 2009.

[3] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, September 2008.

[4] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, June 2004.

[5] M. Douze, H. Jégou, H. Singh, L. Amsaleg, and C. Schmid. Evaluation of GIST descriptors for web-scale image search. In *CIVR*, July 2009.

[6] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, December 1998.

[7] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*. To appear.

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

[9] H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *ICCV*, September 2009.

[10] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 87(3), May 2010.

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

[12] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[13] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005.

[14] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005.

[15] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, February 2009.

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

[17] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

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

[19] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-scale image retrieval with compressed Fisher vectors. In *CVPR*, June 2010.

[20] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, June 2007.

[21] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, June 2008.

[22] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, chapter 3. MIT Press, March 2006.

[23] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, October 2003.

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

[25] L. Torresani, M. Szummer, and A. Fitzgibbon. Learning query-dependent prefilters for scalable image retrieval. In *CVPR*, June 2009.

[26] J. van Gemert, C. Veenman, A. Smeulders, and J. Geusebroek. Visual word ambiguity. *PAMI*. To appear.

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

[28] S. Winder and M. Brown. Learning local image descriptors. In *CVPR*, June 2007.

[29] S. Winder, G. Hua, and M. Brown. Picking the best Daisy. In *CVPR*, June 2009.