Large-Scale Image Retrieval with Compressed Fisher Vectors

Florent Perronnin, Yan Liu, Jorge Sánchez and Hervé Poirier
Xerox Research Centre Europe (XRCE)
Firstname.Lastname@xrce.xerox.com

Abstract

The problem of large-scale image search has been traditionally addressed with the bag-of-visual-words (BOV). In this article, we propose to use as an alternative the Fisher kernel framework. We first show why the Fisher representation is well-suited to the retrieval problem: it describes an image by what makes it different from other images. One drawback of the Fisher vector is that it is high-dimensional and, as opposed to the BOV, it is dense. The resulting memory and computational costs do not make Fisher vectors directly amenable to large-scale retrieval. Therefore, we compress Fisher vectors to reduce their memory footprint and speed-up the retrieval. We compare three binarization approaches: a simple approach devised for this representation and two standard compression techniques. We show on two publicly available datasets that compressed Fisher vectors perform very well using as little as a few hundreds of bits per image, and significantly better than a very recent compressed BOV approach.

1. Introduction

We are interested in the problem of retrieving images of a given object or scene within a large collection of images using the query-by-example paradigm. The de-facto standard to address this problem is the bag-of-visual-words (BOV) [24].

BOV overview: In a nutshell, the BOV works as follows. Interest points are detected in the image and local invariant descriptors are extracted. Each descriptor is assigned to its closest visual word in a "visual vocabulary": a codebook obtained offline by clustering a large set of descriptors with k-means. This results in a typically high-dimensional sparse histogram representation. Similarly to text retrieval, an inverted list structure is employed for efficient indexing and tf-idf scoring is used to discount the influence of visual-words which occur in many images.

This approach has then been refined in a number of ways. Efficient quantization techniques, such as hierarchical k-means (HKM) [18] or approximate k-means (AKM) [21], were proposed for very large vocabularies. As information might be lost in the quantization process, descriptors can be assigned to multiple visual words [22, 11]. [11] proposes the Hamming Embedding (HE) technique which consists in complementing the visual word index for a given descriptor with a binary vector which indicates the approximate location of the descriptor in its Voronoi cell. Discounting the influence of descriptors which happen in burst can lead to additional improvements [12]. Finally, post-processing techniques such as spatial re-ranking [21, 11] and query expansion [4] can boost the accuracy.

Despite the use of inverted list structures, storage can be an issue for BOV approaches. [21] reports a memory usage of 4.3GB for approximately 1.1M images. A state-of-the-art system which makes use of HE is reported to take 35kB of memory per image [13], or equivalently more than 32GB for 1M images. As soon as data does not fit into RAM, the response time of the system collapses because of slow disk accesses [21]. Therefore, there has been much research in compressing BOV histograms.

Small codes: The easiest way to compress the BOV is to binarize the histogram entries as proposed in [24]. It is shown in [13] that this strategy does not incur any significant loss for visual vocabularies with more than a few thousand words. [3, 5] propose to apply min-hash to binary vectors. This technique was shown to be very efficient for the purpose of near-duplicate retrieval. However, when applied to complex retrieval problems, *e.g.* retrieving shots of objects taken from widely varying viewpoints, the memory usage is similar to the BOV.

Another approach is to use lossless compression techniques from text-retrieval to compress inverted files. With such techniques, [9] reports a compression factor of approximately 2 while [13] reports a factor of 4.

Recently, both [13] and [27] noted that standard compression techniques such as Local Sensitive Hashing (LSH) [6] could not be applied directly to the BOV because of its sparse nature and that it needed to be densified before compression. [13] proposes to use random aggregations of visual words. [27] decomposes an image into three layers: background information (which is discarded), information related to a set of pre-learned topics (which is dense and can be compressed with LSH) and residual information (which can be encoded in a sparse manner).

A common issue in all methods which compress BOV histograms is that they can perform as well as the BOV at best [1].

There has been a lot of work recently on computing compact binary codes with application mainly to GIST descriptors [19], including [25, 26, 16, 23]. While GIST descriptors can be very effective to retrieve similar objects and scenes which are well aligned, they cannot cope with wide variations in rotation, scaling or viewpoint [7].

Contributions: In this paper, we go beyond the BOV and the GIST representations. We apply the Fisher kernel to the problem of same object / scene image retrieval. The Fisher kernel is a generic framework introduced in [10] for classification purposes to combine the best of the generative and discriminative worlds. In the field of computer vision, this framework has been successfully applied to a number of problems including categorization [**?**, 20], dimensionality reduction [1] and saliency detection [17]. However, to the best of our knowledge, there has not been any thorough evaluation of Fisher kernels for retrieval. One of the reasons might be that this framework leads to very high-dimensional (hundreds of thousands of dimensions) dense representations. Because of the induced computational and storage costs, these vectors are not directly amenable to large-scale retrieval.

Our contributions are as follows:

1. We motivate the use of the Fisher representation for retrieval: it discounts the influence of background descriptors and therefore describes an image by what makes it different from other images. We also relate the similarity between two Fisher vectors with the similarity between two BOV histograms.

2. We propose a simple normalization procedure which can improve the retrieval accuracy significantly, especially for large visual vocabularies.

3. Since the Fisher vector is dense, we can apply standard binary encoding techniques. We compare a simple binarization strategy specially devised for the Fisher kernel with LSH [2] and Spectral Hashing [26].

This paper is organized as follows. In section 2, we review the Fisher kernel framework and how to apply it to obtain image signatures. In section 3, we provide an interpretation of the Fisher vector and discuss the problem of computing the similarity between such signatures. In section 4, we discuss three alternatives to binarizing the Fisher vector. Finally, we present experimental results on two standard benchmarks, the Holiday dataset [11] and the University of Kentucky benchmark [18], showing the excellent performance of our approach.

[1][13] reports better results than the BOV with compressed vectors but acknowledges that this is certainly due to a suboptimal choice of similarity for the BOV.

2. The Fisher Vector Representation

We first provide an introduction to the Fisher kernel framework as proposed in [10]. We then review the application of this framework to the problem of computing image signatures as proposed in [20].

2.1. Fisher kernel basics

Let X be a sample whose generation process can be modeled by a probability density function p with parameters λ. In our case, X corresponds to a single image. X can be described by the gradient vector:

$$G_\lambda^X = \nabla_\lambda \log p(X|\lambda). \tag{1}$$

The gradient of the log-likelihood describes the contribution of the parameters to the generation process. The dimensionality of this vector depends only on the number of parameters in λ. A natural kernel on these gradients is:

$$K(X,Y) = G_\lambda^{X\prime} F_\lambda^{-1} G_\lambda^Y \tag{2}$$

where F_λ is the Fisher information matrix of p:

$$F_\lambda = E_{x\sim p}\left[\nabla_\lambda \log p(x|\lambda) \nabla_\lambda \log p(x|\lambda)'\right]. \tag{3}$$

As F_λ is symmetric and positive definite, it has a Cholesky decomposition $F_\lambda = L_\lambda' L_\lambda$ and $K(X,Y)$ can be rewritten as a dot-product between normalized vectors $\mathcal{G}_\lambda$ with:

$$\mathcal{G}_\lambda^X \quad = \quad L_\lambda G_\lambda^X. \tag{4}$$

We will refer to $\mathcal{G}_\lambda^X$ as the *Fisher vector* of X.

2.2. Representing images with Fisher vectors

In the case where we want to describe a given image, $X = \{x_t, t = 1 \dots T\}$ is the set of descriptors extracted from the image (typically T is equal to a few hundreds or thousands). We assume that the x_t's are generated independently by p. A natural choice for p is a Gaussian mixture model (GMM): $p(x) = \sum_{i=1}^N w_i p_i(x)$. Each Gaussian p_i can be viewed as a visual word and N is the vocabulary size. In the remainder of the article, we therefore use the words "Gaussian" and "visual word" interchangeably. We denote $\lambda = \{w_i, \mu_i, \Sigma_i, i = 1 \dots N\}$ where w_i, μ_i and Σ_i are respectively the mixture weight, mean vector and covariance matrix of Gaussian i. We assume that the covariance matrices are diagonal and we denote by σ_i^2 the variance vector. The GMM p is trained on a large number of images using Maximum Likelihood Estimation (MLE).

In this paper, we focus on the partial derivatives with respect to the mean parameters. We make use of the diagonal closed-form approximation of [20], in which case the normalization of the gradient by $L_\lambda = F_\lambda^{-1/2}$ is simply a whitening of the dimensions.

Let $\gamma_t(i) = p(i|x_t)$ be the soft assignment of descriptor x_t to Gaussian i:

$$\gamma_t(i) = \frac{w_i p_i(x_t)}{\sum_{j=1}^N w_j p_j(x_t)}. \tag{5}$$

Let D denote the dimensionality of the descriptors. Let $\mathcal{G}_i^X$ be the D-dimensional gradient with respect to the mean μ_i of Gaussian i. Simple mathematical derivations lead to:

$$\mathcal{G}_i^X = \frac{1}{\sqrt{w_i}} \sum_{t=1}^T \gamma_t(i) \left(\frac{x_t - \mu_i}{\sigma_i} \right) \tag{6}$$

where the division between vectors is as a term-by-term operation. The final gradient vector $\mathcal{G}_\lambda^X$ is the concatenation of the $\mathcal{G}_i^X$ vectors for $i = 1 \ldots N$ and is therefore ND-dimensional. In section 5 we will experiment with values of N ranging from 1 to $4,096$.

3. Measuring the Similarity of Fisher Vectors

We now show that the Fisher vector is a good representation for retrieval as it automatically discounts the influence of background descriptors. Leveraging this interpretation, we then explain why the dot product between Fisher vectors is a good similarity. We finally propose a simple normalization step for the Fisher vector.

3.1. Fisher vector and TF-IDF

The tf-idf as proposed in text (and as applied to the BOV) discounts the influence of frequent *discrete events*, where an event is the occurrence of a word in a document (a visual word in an image). In the field of text processing, [8] connected the Fisher kernel and tf-idf scoring in the case of a specific model: the Dirichlet compound multinomial. We now show that the Fisher vector as applied to a GMM extends the tf-idf scoring to *continuous events*, where an event is the occurrence of a descriptor in an image.

Let x_t be a descriptor generated by Gaussian i. Given that our descriptors are fairly high-dimensional ($D = 64$), γ_t is typically a peaky distribution, *i.e.* $\gamma_t(i) \approx 1$ and $\gamma_t(j) \approx 0$ for $j \neq i$. x_t is likely to occur in any image – and is therefore likely to be a background descriptor – if it has a high likelihood $p(x_t)$ to have been generated by p. Given the assumption $\gamma_t(i) \approx 1$, we have $p(x_t) \approx w_i p_i(x_t)$. Hence, $p(x_t)$ is high if:

1. w_i is high, *i.e.* Gaussian i corresponds to a frequent visual word, and
2. $p_i(x_t)$ is large, *i.e.* if $||\frac{x_t - \mu_i}{\sigma_i}||^2$ is small. Loosely speaking this means that x_t is close to μ_i where proximity is measured with respect to the variance σ_i^2.

Going back to equation (6), we can see that if x_t has a high likelihood $p(x_t)$, its contribution to the gradient vector will be doubly discounted by the $\sqrt{w_i}$ division and the μ_i subtraction. Therefore, the influence of frequent (*i.e.* background) descriptors is automatically discounted in the Fisher vector in a way similar to tf-idf. This is a compelling argument to use the Fisher vector for retrieval.

3.2. Dot-product on Fisher vectors

Let us now see how the previous analysis translates in the case where we use the dot-product as a similarity. We introduce the following notations:

$$w_i^X = \frac{1}{T} \sum_{t=1}^T \gamma_t(i), \tag{7}$$

$$\mu_i^X = \frac{\sum_{t=1}^T \gamma_t(i) x_t}{\sum_{t=1}^T \gamma_t(i)}. \tag{8}$$

w_i^X is the proportion of descriptors of X soft-assigned to visual word i, *i.e.* this is the soft-BOV representation. μ_i^X is the average of the descriptors of X weighted by their probability of being assigned to Gaussian i (*i.e.*, loosely speaking, the average of the descriptors of X assigned to Gaussian i). Rewriting (6) using (7), (8) and the notation $\delta_i^X = \frac{\mu_i^X - \mu_i}{\sigma_i}$, we obtain:

$$\frac{1}{T} \mathcal{G}_i^X = \frac{w_i^X}{\sqrt{w_i}} \delta_i^X. \tag{9}$$

Hence, $\mathcal{G}_\lambda^{X\prime} \mathcal{G}_\lambda^Y$ is proportional to:

$$\sum_{i=1}^N \frac{w_i^X w_i^Y}{w_i} \delta_i^{X\prime} \delta_i^Y. \tag{10}$$

Each term in the sum can be decomposed into two parts:

1. $\frac{w_i^X w_i^Y}{w_i}$ is the product of the frequencies of visual word i in X and Y divided by the average frequency of occurrence of i in any image. This is similar to standard tf-idf BOV scoring.
2. The second term $\delta_i^{X\prime} \delta_i^Y$ is large if the δ_i^X and δ_i^Y vectors have *a similar direction and a large norm.* δ_i^X and δ_i^Y have a similar direction if the descriptors of X and Y assigned to Gaussian i have the same average. δ_i^X and δ_i^Y have a large norm if μ_i^X and μ_i^Y are significantly different from μ_i. Based on the analysis of section 3.1, a large norm indicates that on average the descriptors of X assigned to Gaussian i are unlikely to be background descriptors.

Hence, the dot-product appears as a good measure for the Fisher vector. We note that we do not use directly the dot-product on Fisher vectors for retrieval but the cosine similarity (i.e. the dot product between the L2 normalized vectors). It guarantees that if we query a database with one of its images, the first result will be the image itself – a desirable property. We underline that L2 normalization automatically normalizes the Fisher vector by the number of descriptors contained in the image. The previous analysis remains valid in the case of the cosine similarity.

3.3. Fisher vector normalization

Although the dot-product (or the cosine) is a natural measure of similarity on Fisher vectors, it does not necessarily lead to the optimal accuracy. Especially, we found-out that discounting the influence of large values before the L2 normalization could be beneficial. A simple way to achieve this goal is to raise each dimension of the Fisher vector to the power of a value $\alpha \in [0, 1]$ (using the same value α for all dimensions). If we go back to equation (9) the effects of this operation are easy to understand. The power on the BOV part downplays the influence of those descriptors which happen frequently within a given image (bursty visual features) in a manner similar to [12]. The power on the δ_i^X part discounts the influence of outlier descriptors.

We underline that the proposed α normalization does not modify our interpretation of the dot-product between Fisher vectors.

4. Binarizing Fisher Vectors

The direct application of the Fisher kernel framework to large-scale retrieval faces an important challenge: the gradient representations can be high-dimensional and dense. For instance, if we use descriptors of dimension $D = 64$ (as is the case in our experiments) and a visual vocabulary of size $N = 100$ (which is tiny by BOV standards), then the Fisher vector already contains $DN = 6,400$ dimensions. If we use floating points, this results in a 25kB signature.

We now describe three approaches to compressing the Fisher vector. The first one is a simple approach devised specially for the Fisher vector. The second and third ones are standard compression techniques: LSH and Spectral Hashing (SH).

4.1. $\alpha = 0$ binarization

As α goes to zero, z^α converges to -1 if $z < 0$, 1 if $z > 1$ and 0 if $z = 0$. Hence, as α goes to zero, the α-normalized Fisher vector (c.f. section 3.3) converges to a ternary representation. We will show experimentally that, despite the supposed information loss, this representation yields good results, especially as the number of Gaussians in the visual vocabulary increases. We now explain how to turn this ternary encoding into an equivalent binary encoding which is more efficient both in terms of storage and computation. We take advantage of the fact that the Fisher vector of X can be decomposed into the product of a BOV part w_i^X and a δ_i^X part (c.f. equation (9)):

1. We encode the BOV part w_i^X in a binary fashion as proposed in [24]. We define $b_i^X = 1$ if $w_i^X > 0$ and $b_i^X = 0$ otherwise [2].

2. We encode each dimension of δ_i^X on a single bit based on its sign. Let u_i^X be the binarized version of δ_i^X. u_i^X is reminiscent of the Hamming Embedding (HE) of [11] as it encodes in a binary fashion the approximate location of the descriptors which are assigned to Gaussian i. A major difference is the fact that [11] encodes each descriptor individually while we encode all the descriptors assigned to the same Gaussian jointly which is much more efficient.

Hence, the binarized Fisher vector is encoded on $N(D+1)$ bits. The dot-product between the ternary representations can be computed equivalently as a similarity between the binary representations:

$$\sum_{i=1}^{N} b_i^X b_i^Y \left(D - 2Ha(u_i^X, u_i^Y)\right) \tag{11}$$

where Ha is the Hamming distance between u_i^X and u_i^Y. Again, we do not use the dot-product directly but the cosine and therefore the above similarity should be divided by $\sqrt{D \sum_{i=1}^{N} b_i^X} \sqrt{D \sum_{i=1}^{N} b_i^Y}$.

4.2. Local Sensitive Hashing (LSH)

As we use the cosine to measure the similarity between Fisher vectors, we can make use of LSH to binarize Fisher vectors [2]. Let x and y be two d-dimensional vectors. Let r be a d-dimensional vector such that each of its coordinates is drawn from a 1-D Gaussian distribution with mean 0 and standard deviation 1. Let $h_r(x) = Q(x'r)$ where $Q(z) = +1$ if $z \geq 0$ and 0 otherwise. We have:

$$P\left(h_r(x) = h_r(y)\right) = 1 - \frac{\theta(x,y)}{\pi} \tag{12}$$

where $\theta(x, y)$ is the angle between x and y. To binarize a vector x, one simply draws a set of random vectors $\{r_b, b = 1 \ldots B\}$ and computes the sign of $r_b'x$.

[2]Because we make use of a probabilistic framework, w_i^X cannot be equal to a mathematical zero and therefore, this encoding should always yield $b_i^X = 1$. However, to speed-up the Fisher vector computation we have set the value of the soft-assignment $\gamma_t(i)$ to zero if its value was below 1e-4 and therefore we have zero w_i^X values in practice.

4.3. Spectral Hashing (SH)

The SH algorithm [26] finds a binary encoding such that points which are far apart in the original Euclidean space are also fart apart in the Hamming space and vice-versa. Since the cosine similarity is equivalent to the Euclidean distance on L2-normalized vectors, we can apply the SH algorithm to our L2-normalized Fisher vector representations.

In a nutshell, the idea of SH is to minimize the sum of the Hamming distances between pairs of binary codes weighted by the Gaussian kernel between the corresponding vectors. The solution is based on the eigenfunctions of continuous Laplacian operators. If one assumes that the data is uncorrelated (which is the case if the feature vectors have been projected on the principal components) and that it is uniformly distributed in each dimension, one can then make use of a simple and efficient encoding rule on a per-dimension basis.

In our experiments, we used the Matlab code provided by [26].

5. Experiments

5.1. Datasets and experimental set-up

We used publicly available datasets as well as publicly available feature detectors and descriptors to ensure the comparability of our results.

Datasets: We report results on two standard benchmarks: the Holiday dataset [11] and the University of Kentucky benchmark [18]. The Holiday dataset contains 1,491 images of 500 scenes and objects and the first image of each scene is used as a query. The retrieval accuracy is measured as the Average Precision (AP) averaged over the 500 queries (the query image is discarded for the AP computation). The Kentucky benchmark contains 10,200 images of 2,550 objects (4 images per object) and each image is used in turn as a query. The accuracy is measured in terms of the number of relevant images in the top 4 retrieved images, *i.e.* $4\times$ recall@4. For learning purposes, we used an additional set of 60K images (later referred to as Flickr60k) made available by the authors of [11]. Finally, for large-scale experiments, we also used an additional set of 1M distractor Flickr images (later referred to as Flickr1M) made available by the same authors.

Experimental set-up: We used the same feature detection and description procedure as in [11] (Hessian-Affine extractor and the SIFT descriptor) since the code and the features are available online. We reduced the dimensionalities of these features from 128 to $D = 64$ through Principal Component Analysis (PCA) as we observed in preliminary experiments that it had a positive impact on the accuracy. We used the Flickr60k dataset to learn the PCA and the GMM vocabularies as well as the parameters of the SH binary encoding. Except where it is mentioned explicitly, the Fisher vectors used in our experiments were computed by taking the gradient with respect to the mean parameters only.

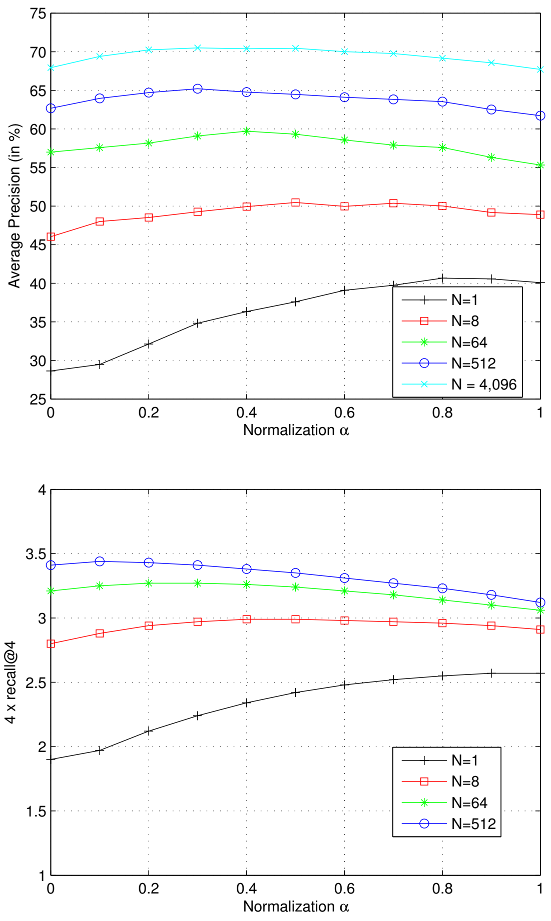


Figure 1. Influence of the vocabulary size N and the normalization factor α (c.f. section 3.3). Top: Holiday dataset. Bottom: Kentucky benchmark.

5.2. Experiments on non-binarized Fisher vectors

We first observe in Figure 1 the influence of the vocabulary size N and the normalization factor α. As the vocabulary size increases, the optimal value of α gets closer to zero. For instance, on the Holiday dataset, for $N = 1$ $\alpha = 0.8$ yields the best results while for $N = 4,096$, $\alpha = 0.3$ is optimal. The best results are 70.5% AP with $N = 4,096$ on the Holiday dataset and a score of 3.44 with $N = 512$ on the Kentucky benchmark. If we compute Fisher vectors by taking the gradient with respect to the mean and standard deviation, we can obtain a small increase of the accuracy to 73.5% AP on the Holiday dataset and 3.50 on the Kentucky benchmark at the expense of doubling the size of Fisher vectors. If we use only the BOV part of the similarity and disregard the δ part (c.f. equation

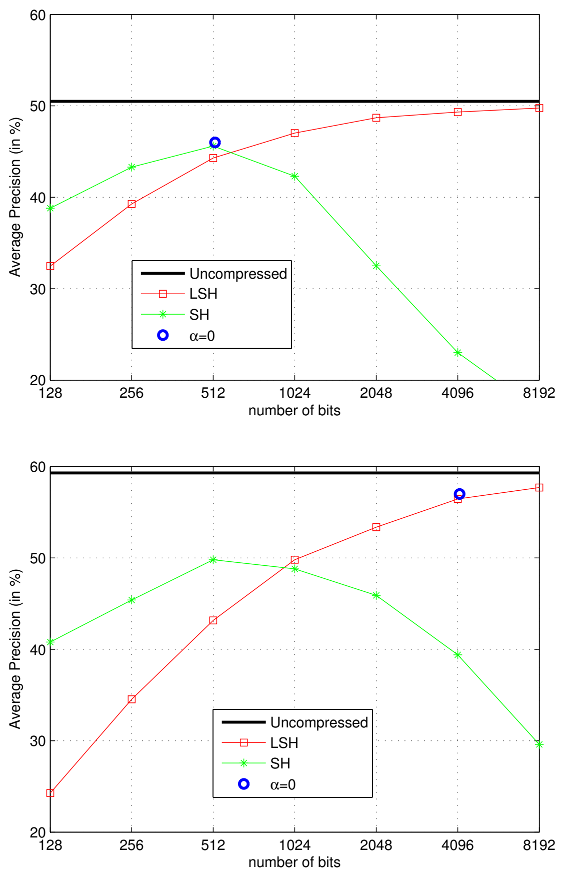


Figure 2. Comparison of the three proposed binarization schemes on the Holiday dataset for a vocabulary size of $N = 8$ Gaussians (top) and $N = 64$ Gaussians (bottom). Uncompressed = non-binarized accuracy (with the best α).

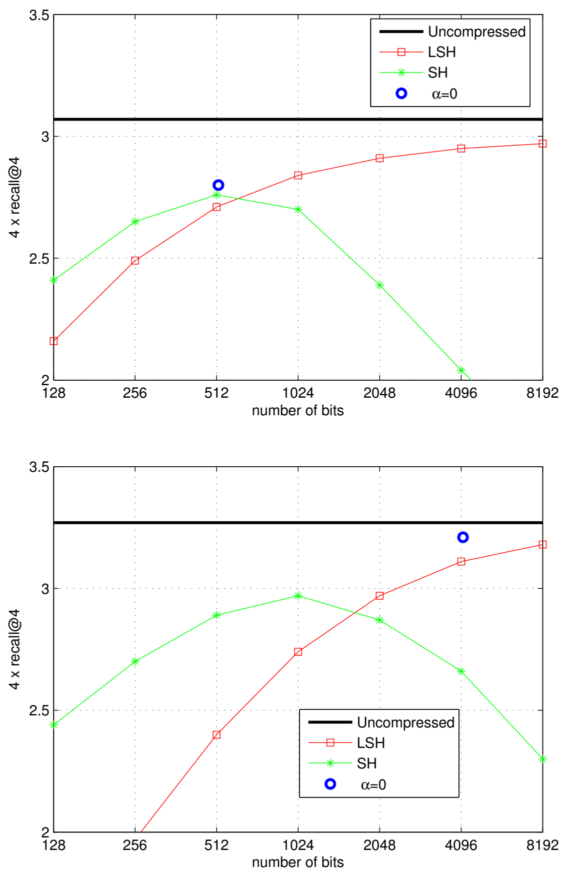


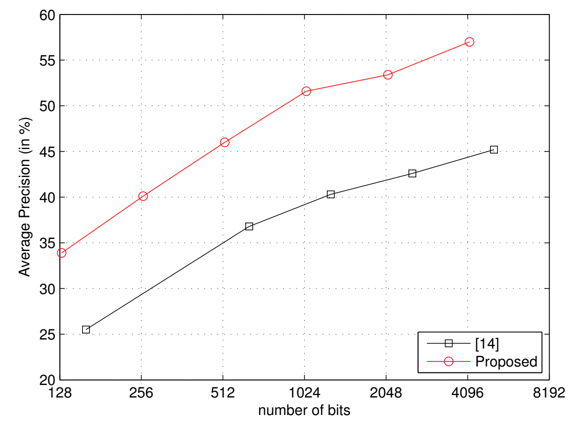
Figure 3. Comparison of the three proposed binarization schemes on the Kentucky benchmark for a vocabulary size of $N = 8$ Gaussians (top) and $N = 64$ Gaussians (bottom). Uncompressed = non-binarized accuracy (with the best α).

(10)), we obtain 41.9% AP and a score of 2.70 respectively. We underline that the Fisher vector yields reasonable results even for tiny vocabularies. For instance, with *a single visual word* ($N = 1$), we report an AP of 40.7% on the Holiday dataset, which is comparable to a BOV with 4,000 visual words.

We can compare these results to the state-of-the-art. Using the same features and a BOV approach, [13] reports on the Holiday dataset an AP of 54.9% (200K visual words) and on the Kentucky benchmark a score of 3.02 (20k visual words). [11] reports 72.7% AP on the Holiday dataset using Hamming Embedding (20k visual words). On the Kentucky benchmark, [15] reports a score of 3.60 using a fairly sophisticated contextual similarity. Hence, our approach far exceeds the accuracy of standard BOV approaches and is comparable to the state-of-the-art approaches (when no geometrical information is used).

5.3. Experiments on binarized Fisher vectors

We experiment with three approaches to binarizing Fisher vectors: the proposed approach which is equivalent to setting $\alpha = 0$, LSH and SH (c.f. section 4). Results are given for Holiday and Kentucky in Figures 2 and 3 respectively. For LSH and SH, we used the best α parameter for each vocabulary size and each dataset (which gives an advantage with respect to the simple $\alpha = 0$ binarization). Also, for LSH and SH we can vary the number of bits for a given vocabulary size N. For the $\alpha = 0$ binarization, given N, the number of bits is fixed and equal to $N(D+1) = 65N$ bits. We observe that for a small number of bits SH outperforms LSH but for a larger number of bits, LSH outperforms SH. This is in line with [23] which reports that SH can be outsmarted by binarization algorithms based on random projections for a large number of bits. Somewhat surprisingly, the simple $\alpha = 0$ binarization scheme works extremely well.



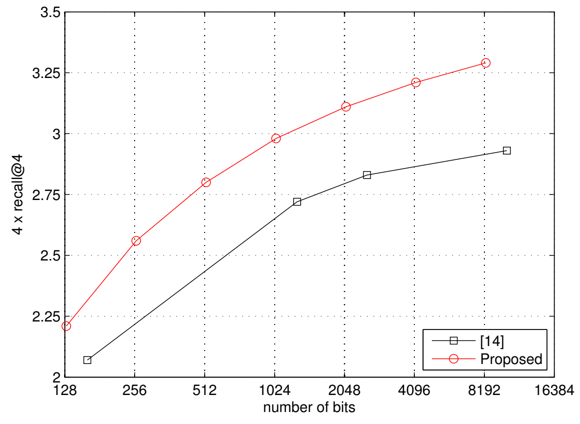


Figure 4. Comparison of the proposed binarized Fisher vectors and the results of [13] (compressed BOV). Top: Holiday dataset. Bottom: Kentucky benchmark. The results of [13] are directly taken from their Tables 1 and 2.

We now compare in Figure 4 our results with the recent results of [13] based on compressed BOV vectors. For Fisher vectors, we use the $\alpha = 0$ binarization scheme. We observe a very significant increase of the retrieval accuracy for a similar number of bits. For instance, on the Holiday dataset, we can do as well as their best reported results with approximately 10 times fewer bits. Yet, in all fairness, we should mention that [13] requires to scan only a fraction of the database while we perform an exhaustive comparison of the query with all database signatures.

Finally, we report in Figure 5 the results of our large-scale experiments using the combined Holiday+Flickr1M datasets. The same 500 Holiday images are queried and the 1M Flickr images are used as distractors. We report the recall@K (as is the case of [11, 13]) for various vocabulary sizes N. Again, if we compare our results with the recent results of [13], we observe a very significant improvement of the recall@K for a comparable number of bits. For instance, they report a recall@1000 around 45% for 5,120 bits per image (32 minibofs) while we report close to 65% with 4,160 bits.

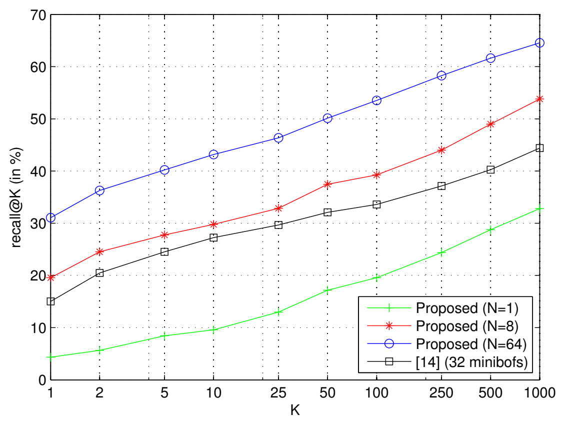


Figure 5. Large scale experiments on Holiday+Flickr1M. Comparison of the proposed approach (for various values of N) with the best results reported in [13] (compressed BOV).

For $N = 8$, an image signature takes 520 bits (1M+ images fit in 60MB) and the response time is approximately 100 ms per query on a 3GHz Opteron machine *using a single CPU*. For $N = 64$, one image signature takes 4,160 bits (1M+ images fit in 500MB) and the response time is approximately 440 ms per query. We show in Figure 6 sample results on the Holiday+Flickr1M dataset for 520 bits signatures. In some cases, even with such compact signatures, we can go well beyond duplicate detection.

Finally, in the same proceedings, Jégou *et al.* [14] report impressive results for very compact codes (128 bits) using a signature inspired by the Fisher vector and product quantizers. We believe that our Fisher vector signatures could benefit from similar quantizers.

6. Conclusion

In this article, we applied the Fisher kernel framework to image retrieval. We explained that the Fisher vector is particularly well-suited to retrieval because it discounts the influence of background descriptors. The Fisher vector, being high-dimensional and dense, cannot be directly applied to large-scale retrieval. We therefore explored three different techniques to compressing the Fisher vector: a simple approach devised for the Fisher kernel, LSH and SH. We showed that the simple binarization strategy performs extremely well and we showed that compressed Fisher vectors significantly outperform a recent retrieval technique using compressed BOV technique [13].

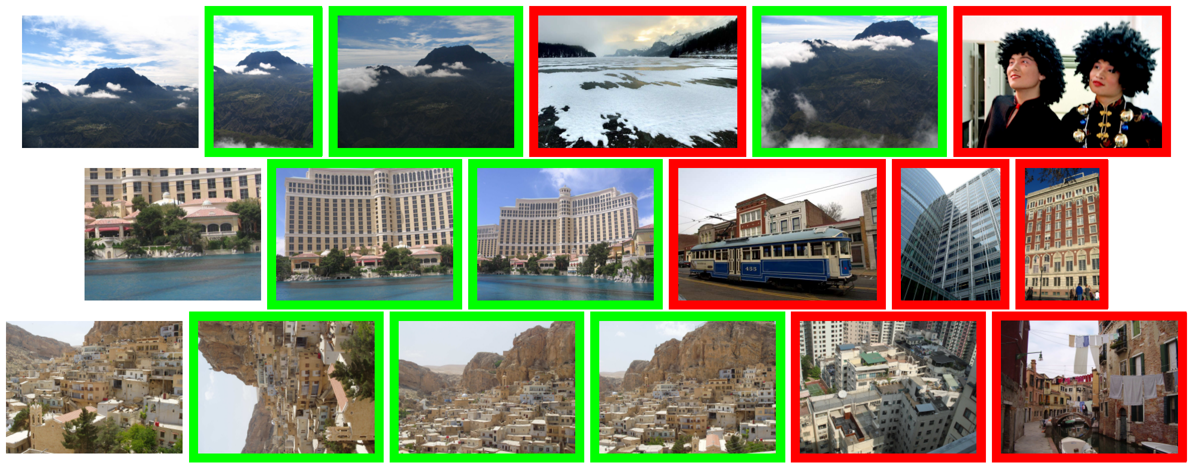


Figure 6. Sample results on the combined Holiday+Flickr1M dataset using a vocabulary of N=8 visual words, *i.e.* 520 bits signatures. The images on the left correspond to queries and the following images correspond to the first 5 retrieved images (discarding the query). Green/red boxes indicate correct/incorrect results respectively.

Acknowledgments

This work was partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^o 216529.

References

[1] M. Bressan, C. Cifarelli, and F. Perronnin. An analysis of the relationship between painters based on their work. In *ICIP*, 2008.

[2] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[3] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detetction. In *CIVR*, 2007.

[4] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, 2007.

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

[6] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SOCG*, 2004.

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

[8] C. Elkan. Deriving tf-idf as a fisher kernel. In *SPIRE*, 2005.

[9] S. Gammeter, L. Bossard, T. Quack, and L. V. Gool. I know what you did last summer: object-level auto-annotation of holiday snaps. In *ICCV*, 2009.

[10] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.

[11] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008. http://lear.inrialpes.fr/˜jegou/data.php.

[12] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *CVPR*, 2009.

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

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

[15] H. Jégou, H. Harzallah, and C. Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *CVPR*, 2007.

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

[17] L. Marchesotti, C. Cifarelli, and G. Csurka. A framework for visual saliency detetction with application to image thumnbnailing. In *ICCV*, 2009.

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

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

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

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

[22] 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*, 2008.

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

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

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

[26] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008. http://www.cs.huji.ac.il/˜yweiss/SpectralHashing/.

[27] X. Zhang, Z. Li, L. Zhang, W.-Y. Ma, and H.-Y. Shum. Efficient indexing for large-scale visual search. In *ICCV*, 2009.