International Journal of Computer Vision manuscript No.
(will be inserted by the editor)

Image Classification with the Fisher Vector: Theory and Practice

Jorge Sánchez · Florent Perronnin · Thomas Mensink · Jakob Verbeek

Received: date / Accepted: date

Abstract A standard approach to describe an image for classification and retrieval purposes is to extract a set of local patch descriptors, encode them into a high dimensional vector and pool them into an image-level signature. The most common patch encoding strategy consists in quantizing the local descriptors into a finite set of prototypical elements. This leads to the popular Bag-of-Visual words (BoV) representation. In this work, we propose to use the Fisher Kernel framework as an alternative patch encoding strategy: we describe patches by their deviation from an "universal" generative Gaussian mixture model. This representation, which we call Fisher Vector (FV) has many advantages: it is efficient to compute, it leads to excellent results even with efficient linear classifiers, and it can be compressed with a minimal loss of accuracy using product quantization. We report experimental results on five standard datasets – PASCAL VOC 2007, Caltech 256, SUN 397, ILSVRC 2010 and ImageNet10K – with up to 9M images and 10K classes, showing that the FV framework is a state-of-the-art patch encoding technique.

Jorge Sánchez
CIEM-CONICET, FaMAF, Universidad Nacional de Córdoba,
X5000HUA, Córdoba, Argentine
E-mail: jsanchez@famaf.unc.edu.ar

Florent Perronnin
Xerox Research Centre Europe,
6 chemin de Maupertuis, 38240 Meylan, France
E-mail: florent.perronnin@xrce.xerox.com

Thomas Mensink
Inteligent Systems Lab Amsterdam, University of Amsterdam,
Science Park 904, 1098 XH, Amsterdam, The Netherlands
E-mail: thomas.mensink@uva.nl

Jakob Verbeek
LEAR Team, INRIA Grenoble,
655 Avenue de l'Europe, 38330 Montbonnot, France
E-mail: jakob.verbeek@inria.fr

1 Introduction

This article considers the image classification problem: given an image, we wish to annotate it with one or multiple keywords corresponding to different semantic classes. We are especially interested in the large-scale setting where one has to deal with a large number of images and classes. Large-scale image classification is a problem which has received an increasing amount of attention over the past few years as larger labeled images datasets have become available to the research community. For instance, as of today, ImageNet[1] consists of more than 14M images of 22K concepts (Deng et al, 2009) and Flickr contains thousands of groups[2] – some of which with hundreds of thousands of pictures – which can be exploited to learn object classifiers (Perronnin et al, 2010c, Wang et al, 2009).

In this work, we describe an image representation which yields high classification accuracy and, yet, is sufficiently efficient for large-scale processing. Here, the term "efficient" includes the cost of computing the representations, the cost of learning the classifiers on these representations as well as the cost of classifying a new image.

By far, the most popular image representation for classification has been the Bag-of-Visual words (BoV) (Csurka et al, 2004). In a nutshell, the BoV consists in extracting a set of local descriptors, such as SIFT descriptors (Lowe, 2004), in an image and in assigning each descriptor to the closest entry in a "visual vocabulary": a codebook learned offline by clustering a large set of descriptors with k-means. Averaging the occurrence counts – an operation which is generally referred to as average pooling – leads to a histogram of "visual word" occurrences. There have been several extensions of this popular framework including the use of better coding

[1] http://www.image-net.org
[2] http://www.flickr.com/groups

techniques based on soft assignment (Farquhar et al, 2005, Perronnin et al, 2006, VanGemert et al, 2010, Winn et al, 2005) or sparse coding (Boureau et al, 2010, Wang et al, 2010, Yang et al, 2009b) and the use of spatial pyramids to take into account some aspects of the spatial layout of the image (Lazebnik et al, 2006).

The focus in the image classification community was initially on developing classification systems which would yield the best possible accuracy fairly independently of their cost as examplified in the PASCAL VOC competitions (Everingham et al, 2010). The winners of the 2007 and 2008 competitions used a similar paradigm: many types of low-level local features are extracted (referred to as "channels"), one BoV histogram is computed for each channel and non-linear kernel classifiers such as χ^2-kernel SVMs are used to perform classification (van de Sande et al, 2010, Zhang et al, 2007). The use of many channels and non-linear SVMs – whose training cost scales somewhere between quadratically and cubically in the number of training samples – was made possible by the modest size of the available databases.

In recent years only the computational cost has become a central issue in image classification and object detection. Maji et al (2008) showed that the runtime cost of an intersection kernel (IK) SVM could be made independent of the number of support vectors with a negligible performance degradation. Maji and Berg (2009) and Wang et al (2009) then proposed efficient algorithms to learn IKSVMs in a time linear in the number of training samples. Vedaldi and Zisserman (2010) and Perronnin et al (2010b) subsequently generalized this principle to any additive classifier. Attempts have been made also to go beyond additive classifiers (Perronnin et al, 2010b, Sreekanth et al, 2010). Another line of research consists in computing BoV representations which are directly amenable to costless linear classification. Boureau et al (2010), Wang et al (2010), Yang et al (2009b) showed that replacing the average pooling stage in the BoV computation by a max-pooling yielded excellent results.

We underline that all the previously mentioned methods are inherently limited by the shortcomings of the BoV. First, it is unclear why such a histogram representation should be optimal for our classification problem. Second, the descriptor quantization is a lossy process as underlined in the work of Boiman et al (2008).

In this work, we propose an alternative patch aggregation mechanism based on the Fisher Kernel (FK) principle of Jaakkola and Haussler (1998). The FK combines the benefits of generative and discriminative approaches to pattern classification by deriving a kernel from a generative model of the data. In a nutshell, it consists in characterizing a sample by its deviation from the generative model. The deviation is measured by computing the gradient of the sample log-likelihood with respect to the model parameters. This leads to a vectorial representation which we call Fisher Vector (FV). In the image classification case, the samples correspond to the local patch descriptors and we choose as generative model a Gaussian Mixture Model (GMM) which can be understood as a "probabilistic visual vocabulary".

The FV representation has many advantages with respect to the BoV. First, it provides a more general way to define a kernel from a generative process of the data: we show that the BoV is a particular case of the FV where the gradient computation is restricted to the mixture weight parameters of the GMM. We show experimentally that the additional gradients incorporated in the FV bring large improvements in terms of accuracy. A second advantage of the FV is that it can be computed from much smaller vocabularies and therefore at a lower computational cost. A third advantage of the FV is that it performs well even with simple linear classifiers. A significant benefit of linear classifiers is that they are very efficient to evaluate and efficient to learn (linear in the number of training samples) using techniques such as Stochastic Gradient Descent (SGD) (Bottou and Bousquet, 2007, Shalev-Shwartz et al, 2007).

However, the FV suffers from a significant disadvantage with respect to the BoV: while the latter is typically quite sparse, the FV is almost dense. This leads to storage as well as input/output issues which make it impractical for large-scale applications as is. We address this problem using Product Quantization (PQ) (Gray and Neuhoff, 1998) which has been popularized in the computer vision field by Jégou et al (2011) for large-scale nearest neighbor search. We show theoretically why such a compression scheme makes sense when learning linear classifiers. We also show experimentally that FVs can be compressed by a factor of at least 32 with only very limited impact on the classification accuracy.

The remainder of this article is organized as follows. In Section 2, we introduce the FK principle and describe its application to images. We also introduce a set of normalization steps which greatly improve the classification performance of the FV. Finally, we relate the FV to several recent patch encoding methods and kernels on sets. In Section 3, we provide a first set of experimental results on three small- and medium-scale datasets – PASCAL VOC 2007 (Everingham et al, 2007), Caltech 256 (Griffin et al, 2007) and SUN 397 (Xiao et al, 2010) – showing that the FV outperforms significantly the BoV. In Section 4, we present PQ compression, explain how it can be combined with large-scale SGD learning and provide a theoretical analysis of why such a compression algorithm makes sense when learning a linear classifier. In Section 5, we present results on two large datasets, namely ILSVRC 2010 (Berg et al, 2010) (1K classes and approx. 1.4M images) and ImageNet10K (Deng et al, 2010) (approx. 10K classes and 9M images). Finally, we present our conclusions in Section 6.

This paper extends our previous work (Perronnin and Dance, 2007, Perronnin et al, 2010c, Sánchez and Perronnin,

2011) with: (1) a more detailed description of the FK framework and especially of the computation of the Fisher information matrix, (2) a more detailed analysis of the recent related work, (3) a detailed experimental validation of the proposed normalizations of the FV, (4) more experiments on several small- medium-scale datasets with state-of-the-art results, (5) a theoretical analysis of PQ compression for linear classifier learning and (6) more detailed experiments on large-scale image classification with, especially, a comparison to k-NN classification.

2 The Fisher Vector

In this section we introduce the Fisher Vector (FV). We first describe the underlying principle of the Fisher Kernel (FK) followed by the adaption of the FK to image classification. We then relate the FV to several recent patch encoding techniques and kernels on sets.

2.1 The Fisher Kernel

Let $X = \{x_t, t = 1 \dots T\}$ be a sample of T observations $x_t \in \mathscr{X}$. Let u_λ be a probability density function which models the generative process of elements in $\mathscr{X}$ where $\lambda = [\lambda_1, \dots, \lambda_M]' \in \mathbb{R}^M$ denotes the vector of M parameters of u_λ. In statistics, the *score function* is given by the gradient of the log-likelihood of the data on the model:

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

This gradient describes the contribution of the individual parameters to the generative process. In other words, it describes how the parameters of the generative model u_λ should be modified to better fit the data X. We note that $G_\lambda^X \in \mathbb{R}^M$, and thus that the dimensionality of G_λ^X only depends on the number of parameters M in λ and not on the sample size T.

From the theory of information geometry (Amari and Nagaoka, 2000), a parametric family of distributions $\mathscr{U} = \{u_\lambda, \lambda \in \Lambda\}$ can be regarded as a Riemanninan manifold M_Λ with a local metric given by the Fisher Information Matrix (FIM) $F_\lambda \in \mathbb{R}^{M \times M}$:

$$F_\lambda = E_{x \sim u_\lambda}\left[G_\lambda^x G_\lambda^{x\prime}\right]. \tag{2}$$

Following this observation, Jaakkola and Haussler (1998) proposed to measure the similarity between two samples X and Y using the *Fisher Kernel* (FK) which is defined as:

$$K_{FK}(X,Y) = G_\lambda^{X\prime} F_\lambda^{-1} G_\lambda^Y. \tag{3}$$

Since F_λ is positive semi-definite, so is its inverse. Using the Cholesky decomposition $F_\lambda^{-1} = L_\lambda{}' L_\lambda$, the FK in (3) can be re-written explicitly as a dot-product:

$$K_{FK}(X,Y) = \mathscr{G}_\lambda^{X\prime} \mathscr{G}_\lambda^Y, \tag{4}$$

where

$$\mathscr{G}_\lambda^X = L_\lambda G_\lambda^X = L_\lambda \nabla_\lambda \log u_\lambda(X). \tag{5}$$

We call this normalized gradient vector the *Fisher Vector* (FV) of X. The dimensionality of the FV $\mathscr{G}_\lambda^X$ is equal to that of the gradient vector G_λ^X. A non-linear kernel machine using K_{FK} as a kernel is equivalent to a linear kernel machine using $\mathscr{G}_\lambda^X$ as feature vector. A clear benefit of the explicit formulation is that, as explained earlier, linear classifiers can be learned very efficiently.

2.2 Application to images

Model. Let $X = \{x_t, t = 1, \dots, T\}$ be the set of D-dimensional local descriptors extracted from an image, *e.g.* a set of SIFT descriptors (Lowe, 2004). Assuming that the samples are independent, we can rewrite Equation (5) as follows:

$$\mathscr{G}_\lambda^X = \sum_{t=1}^{T} L_\lambda \nabla_\lambda \log u_\lambda(x_t). \tag{6}$$

Therefore, under this independence assumption, the FV is a sum of normalized gradient statistics $L_\lambda \nabla_\lambda \log u_\lambda(x_t)$ computed for each descriptor. The operation:

$$x_t \rightarrow \varphi_{FK}(x_t) = L_\lambda \nabla_\lambda \log u_\lambda(x_t) \tag{7}$$

can be understood as an embedding of the local descriptors x_t in a higher-dimensional space which is more amenable to linear classification. We note that the independence assumption of patches in an image is generally incorrect, especially when patches overlap. We will return to this issue in Section 2.3 as well as in our small-scale experiments in Section 3.

In what follows, we choose u_λ to be a Gaussian mixture model (GMM) as one can approximate with arbitrary precision any continuous distribution with a GMM (Titterington et al, 1985). In the computer vision literature, a GMM which models the generation process of local descriptors in any image has been referred to as a *universal (probabilistic) visual vocabulary* (Perronnin et al, 2006, Winn et al, 2005). We denote the parameters of the K-component GMM by $\lambda = \{w_k, \mu_k, \Sigma_k, k = 1, \dots, K\}$, where w_k, μ_k and Σ_k are respectively the mixture weight, mean vector and covariance matrix of Gaussian k. We write:

$$u_\lambda(x) = \sum_{k=1}^{K} w_k u_k(x), \tag{8}$$

where u_k denotes Gaussian k:

$$u_k(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)'\Sigma_k^{-1}(x-\mu_k)\right\}, \tag{9}$$

and we require:

$$\forall_k : w_k \geq 0, \qquad \sum_{k=1}^{K} w_k = 1, \tag{10}$$

to ensure that $u_\lambda(x)$ is a valid distribution. In what follows, we assume diagonal covariance matrices which is a standard assumption and denote by σ_k^2 the variance vector, *i.e.* the diagonal of Σ_k. We estimate the GMM parameters on a large training set of local descriptors using the Expectation-Maximization (EM) algorithm to optimize a Maximum Likelihood (ML) criterion. For more details about the GMM implementation, the reader can refer to Appendix B.

Gradient formulas. For the weight parameters, we adopt the soft-max formalism of Krapac et al (2011) and define

$$w_k = \frac{\exp(\alpha_k)}{\sum_{j=1}^{K}\exp(\alpha_j)}. \tag{11}$$

The re-parametrization using the α_k avoids enforcing explicitly the constraints in Eq. (10). The gradients of a single descriptor x_t w.r.t. the parameters of the GMM model, $\lambda = \{\alpha_k, \mu_k, \Sigma_k, k = 1,\ldots,K\}$, are:

$$\nabla_{\alpha_k} \log u_\lambda(x_t) = \gamma_t(k) - w_k, \tag{12}$$

$$\nabla_{\mu_k} \log u_\lambda(x_t) = \gamma_t(k)\left(\frac{x_t - \mu_k}{\sigma_k^2}\right), \tag{13}$$

$$\nabla_{\sigma_k} \log u_\lambda(x_t) = \gamma_t(k)\left[\frac{(x_t-\mu_k)^2}{\sigma_k^3} - \frac{1}{\sigma_k}\right], \tag{14}$$

where $\gamma_t(k)$ is the soft assignment of x_t to Gaussian k, which is also known as the posterior probability or responsibility:

$$\gamma_t(k) = \frac{w_k u_k(x_t)}{\sum_{j=1}^{K} w_j u_j(x_t)}, \tag{15}$$

and where the division and exponentiation of vectors should be understood as term-by-term operations.

Having an expression for the gradients, the remaining question is how to compute L_λ, which is the square-root of the inverse of the FIM. In Appendix A we show that under the assumption that the soft assignment distribution $\gamma_t(i)$ is sharply peaked on a single value of i for any patch descriptor x_t (*i.e.* the assignment is almost hard), the FIM is diagonal. In section 3.2 we show a measure of the sharpness of γ_t on real data to validate this assumption. The diagonal FIM can be taken into account by a coordinate-wise normalization of the gradient vectors, which yields the following normalized gradients:

$$\mathcal{G}_{\alpha_k}^X = \frac{1}{\sqrt{w_k}}\sum_{t=1}^{T}\left(\gamma_t(k) - w_k\right), \tag{16}$$

$$\mathcal{G}_{\mu_k}^X = \frac{1}{\sqrt{w_k}}\sum_{t=1}^{T}\gamma_t(k)\left(\frac{x_t-\mu_k}{\sigma_k}\right), \tag{17}$$

$$\mathcal{G}_{\sigma_k}^X = \frac{1}{\sqrt{w_k}}\sum_{t=1}^{T}\gamma_t(k)\frac{1}{\sqrt{2}}\left[\frac{(x_t-\mu_k)^2}{\sigma_k^2} - 1\right]. \tag{18}$$

Note that $\mathcal{G}_{\alpha_k}^X$ is a scalar while $\mathcal{G}_{\mu_k}^X$ and $\mathcal{G}_{\sigma_k}^X$ are D-dimensional vectors. The final FV is the concatenation of the gradients $\mathcal{G}_{\alpha_k}^X$, $\mathcal{G}_{\mu_k}^X$ and $\mathcal{G}_{\sigma_k}^X$ for $k = 1,\ldots,K$ and is therefore of dimension $E = (2D+1)K$.

To avoid the dependence on the sample size (see for instance the sequence length normalization in Smith and Gales (2001)), we normalize the resulting FV by the sample size T, *i.e.* we perform the following operation:

$$\mathcal{G}_\lambda^X \leftarrow \frac{1}{T}\mathcal{G}_\lambda^X \tag{19}$$

In practice, T is almost constant in our experiments since we resize all images to approximately the same number of pixels (see the experimental setup in Section 3.1). Also note that Eq. (16)–(18) can be computed in terms of the following 0-order, 1st-order and 2nd-order statistics (see Algorithm 1):

$$S_k^0 = \sum_{t=1}^{T}\gamma_t(k) \tag{20}$$

$$S_k^1 = \sum_{t=1}^{T}\gamma_t(k)x_t \tag{21}$$

$$S_k^2 = \sum_{t=1}^{T}\gamma_t(k)x_t^2 \tag{22}$$

where $S_k^0 \in \mathbb{R}$, $S_k^1 \in \mathbb{R}^D$ and $S_k^2 \in \mathbb{R}^D$. As before, the square of a vector must be understood as a term-by-term operation.

Spatial pyramids. The Spatial Pyramid (SP) was introduced in Lazebnik et al (2006) to take into account the rough geometry of a scene. It was shown to be effective both for scene recognition (Lazebnik et al, 2006) and loosely structured object recognition as demonstrated during the PASCAL VOC evaluations (Everingham et al, 2007, 2008). The SP consists in subdividing an image into a set of regions and pooling descriptor-level statistics over these regions. Although the SP was introduced in the framework of the BoV, it can also be applied to the FV. In such a case, one computes one FV per image region and concatenates the resulting FVs. If R is the number of regions per image, then the FV representation becomes $E = (2D+1)KR$ dimensional. In this work, we use a very coarse SP and extract 4 FVs per image: one FV for the whole image and one FV in three horizontal stripes corresponding to the top, middle and bottom regions of the image.

We note that more sophisticated models have been proposed to take into account the scene geometry in the FV framework (Krapac et al, 2011, Sánchez et al, 2012) but we will not consider such extensions in this work.

2.3 FV normalization

We now describe two normalization steps which were introduced in Perronnin et al (2010c) and which were shown to

be necessary to obtain competitive results when the FV is combined with a linear classifier.

ℓ_2-normalization. Perronnin et al (2010c) proposed to ℓ_2-normalize FVs. We provide two complementary interpretations to explain why such a normalization can lead to improved results. The first interpretation is specific to the FV and was first proposed in Perronnin et al (2010c). The second interpretation is valid for any high-dimensional vector.

In Perronnin et al (2010c), the ℓ_2-normalization is justified as a way to cancel-out the fact that different images contain different amounts of background information. Assuming that the descriptors $X = \{x_t, t = 1, \ldots, T\}$ of a given image follow a distribution p and using the i.i.d. image model defined above, we can write according to the law of large numbers (convergence of the sample average to the expected value when T increases):

$$\frac{1}{T} G_\lambda^X \approx \nabla_\lambda E_{x\sim p} \log u_\lambda(x) = \nabla_\lambda \int_x p(x) \log u_\lambda(x) dx. \quad (23)$$

Now let us assume that we can decompose p into a mixture of two parts: a background image-independent part which follows u_λ and an image-specific part which follows an image-specific distribution q. Let $0 \leq \omega \leq 1$ be the proportion of image-specific information contained in the image:

$$p(x) = \omega q(x) + (1-\omega) u_\lambda(x). \quad (24)$$

We can rewrite:

$$\frac{1}{T} G_\lambda^X \approx \omega \nabla_\lambda \int_x q(x) \log u_\lambda(x) dx + (1-\omega) \nabla_\lambda \int_x u_\lambda(x) \log u_\lambda(x) dx. \quad (25)$$

If the values of the parameters λ were estimated with a ML process – i.e. to maximize at least locally and approximately $E_{x\sim u_\lambda} \log u_\lambda(x)$ – then we have:

$$\nabla_\lambda \int_x u_\lambda(x) \log u_\lambda(x) dx = \nabla_\lambda E_{x\sim u_\lambda} \log u_\lambda(x) \approx 0. \quad (26)$$

Consequently, we have:

$$\frac{1}{T} G_\lambda^X \approx \omega \nabla_\lambda \int_x q(x) \log u_\lambda(x) dx = \omega \nabla_\lambda E_{x\sim q} \log u_\lambda(x). \quad (27)$$

This shows that the image-independent information is approximately discarded from the FV, a desirable property. However, the FV still depends on the proportion of image-specific information ω. Consequently, two images containing the same object but different amounts of background information (*e.g.* the same object at different scales) will have different signatures. Especially, small objects with a small ω value will be difficult to detect. To remove the dependence on ω, we can ℓ_2-normalize[3] the vector G_λ^X or $\mathscr{G}_\lambda^X$.

[3] Normalizing by any ℓ_p-norm would cancel-out the effect of ω. Perronnin et al (2010c) chose the ℓ_2-norm because it is the natural norm associated with the dot-product. In Section 3.2 we experiment with different ℓ_p-norms.

We now propose a second interpretation which is valid for any high-dimensional vector (including the FV). Let $U_{p,E}$ denote the uniform distribution on the ℓ_p unit sphere in an E-dim space. If $u \sim U_{p,E}$, then a closed form solution for the marginals over the ℓ_p-normalized coordinates $u_i = u_i / \|u\|_p$, is given in Song and Gupta (1997):

$$g_{p,E}(u_i) = \frac{p\Gamma(E/p)}{2\Gamma(1/p)\Gamma((E-1)/p)} (1 - |u_i|^p)^{(E-1)/p-1} \quad \text{with} \quad u_i \in [-1,1] \quad (28)$$

For $p = 2$, as the dimensionality E grows, this distribution converges to a Gaussian (Spruill, 2007). Moreover, Burrascano (1991) suggested that the ℓ_p metric is a good measure between data points if they are distributed according to a generalized Gaussian:

$$f_p(x) = \frac{p^{(1-1/p)}}{2\Gamma(1/p)} \exp\left(-\frac{|x-x_0|^p}{p}\right). \quad (29)$$

To support this claim Burrascano showed that, for a given value of the dispersion as measured with the ℓ_p-norm, f_p is the distribution which maximizes the entropy and therefore the amount of information. Note that for $p = 2$, equation (29) corresponds to a Gaussian distribution. From the above and after noting that: *a)* FVs are high dimensional signatures, *b)* we rely on linear SVMs, where the similarity between samples is measured using simple dot-products, and that *c)* the dot-product between ℓ_2-normalized vectors relates to the ℓ_2-distance as $\|x-y\|_2^2 = 2(1-x'y)$, for $\|x\|_2 = \|y\|_2 = 1$, it follows that choosing $p = 2$ for the normalization of the FV is natural.

Power normalization. In Perronnin et al (2010c), it was proposed to perform a power normalization of the form:

$$z \leftarrow \text{sign}(z)|z|^\rho \text{ with } 0 < \rho \leq 1 \quad (30)$$

to each dimension of the FV. In all our experiments the power coefficient is set to $\rho = \frac{1}{2}$, which is why we also refer to this transformation as "signed square-rooting" or more simply "square-rooting". The square-rooting operation can be viewed as an explicit data representation of the Hellinger or Bhattacharyya kernel, which has also been found effective for BOV image representations, see *e.g.* Perronnin et al (2010b) or Vedaldi and Zisserman (2010).

Several explanations have been proposed to justify such a transform. Perronnin et al (2010c) argued that, as the number of Gaussian components of the GMM increases, the FV becomes sparser which negatively impacts the dot-product. In the case where FVs are extracted from sub-regions, the "peakiness" effect is even more prominent as fewer descriptor-level statistics are pooled at a region-level compared to the image-level. The power normalization "unsparsifies" the FV and therefore makes it more suitable for comparison with the dot-product. Another interpretation proposed in Perronnin et al (2010a) is that the power normalization downplays the

influence of descriptors which happen frequently within a given image (bursty visual features) in a manner similar to Jégou et al (2009). In other words, the square-rooting corrects for the incorrect independence assumption. A more formal justification was provided in Jégou et al (2012) as it was shown that FVs can be viewed as emissions of a compound distribution whose variance depends on the mean. However, when using metrics such as the dot-product or the Euclidean distance, the implicit assumption is that the variance is stabilized, *i.e.* that it does not depend on the mean. It was shown in Jégou et al (2012) that the square-rooting had such a stabilization effect.

All of the above papers acknowledge the incorrect patch independence assumption and try to correct *a posteriori* for the negative effects of this assumption. In contrast, Cinbis et al (2012) proposed to go beyond this independence assumption by introducing an exchangeable model which ties all local descriptors together by means of latent variables that represent the GMM parameters. It was shown that such a model leads to discounting transformations in the Fisher vector similar to the simpler square-root transform, and with a comparable positive impact on performance.

We finally note that the use of the square-root transform is not specific to the FV and is also beneficial to the BoV as shown for instance by Perronnin et al (2010b), Vedaldi and Zisserman (2010), Winn et al (2005).

2.4 Summary

To summarize the computation of the FV image representation, we provide an algorithmic description in Algorithm 1. In practice we use SIFT (Lowe, 2004) or Local Color Statistics (Clinchant et al, 2007) as descriptors computed on a dense multi-scale grid. To simplify the presentation in Algorithm 1, we have assumed that Spatial Pyramids (SPs) are not used. When using SPs, we follow the same algorithm for each region separately and then concatenate the FVs obtained for each cell in the SP.

2.5 Relationship with other patch-based approaches

The FV is related to a number of patch-based classification approaches as we describe below.

Relationship with the Bag-of-Visual words (BoV). First, the FV can be viewed as a generalization of the BoV framework (Csurka et al, 2004, Sivic and Zisserman, 2003). Indeed, in the soft-BoV (Farquhar et al, 2005, Perronnin et al, 2006, VanGemert et al, 2010, Winn et al, 2005), the average number of assignments to Gaussian k can be computed as:

$$\frac{1}{T}\sum_{t=1}^{T}\gamma_t(k) = \frac{S_k^0}{T}. \tag{34}$$

Algorithm 1 Compute Fisher vector from local descriptors

Input:

- Local image descriptors $X = \{x_t \in \mathbb{R}^D, t = 1,\ldots,T\}$,
- Gaussian mixture model parameters $\lambda = \{w_k, \mu_k, \sigma_k, k = 1,\ldots,K\}$

Output:

- normalized Fisher Vector representation $\mathcal{G}_\lambda^X \in \mathbb{R}^{K(2D+1)}$

1. **Compute statistics**
   - For $k = 1,\ldots,K$ initialize accumulators
     - $S_k^0 \leftarrow 0, \quad S_k^1 \leftarrow 0, \quad S_k^2 \leftarrow 0$
   - For $t = 1,\ldots T$
     - Compute $\gamma_t(k)$ using equation (15)
     - For $k = 1,\ldots,K$:
       - $S_k^0 \leftarrow S_k^0 + \gamma_t(k)$,
       - $S_k^1 \leftarrow S_k^1 + \gamma_t(k)x_t$,
       - $S_k^2 \leftarrow S_k^2 + \gamma_t(k)x_t^2$
2. **Compute the Fisher vector signature**
   - For $k = 1,\ldots,K$:

$$\mathcal{G}_{\alpha_k}^X = \left(S_k^0 - Tw_k\right)/\sqrt{w_k} \tag{31}$$

$$\mathcal{G}_{\mu_k}^X = \left(S_k^1 - \mu_k S_k^0\right)/\left(\sqrt{w_k}\sigma_k\right) \tag{32}$$

$$\mathcal{G}_{\sigma_k}^X = \left(S_k^2 - 2\mu_k S_k^1 + (\mu_k^2 - \sigma_k^2)S_k^0\right)/\left(\sqrt{2w_k}\sigma_k^2\right) \tag{33}$$

   - Concatenate all Fisher vector components into one vector

$$\mathcal{G}_\lambda^X = \left(\mathcal{G}_{\alpha_1}^X,\ldots,\mathcal{G}_{\alpha_K}^X,\mathcal{G}_{\mu_1}^{X\prime},\ldots,\mathcal{G}_{\mu_K}^{X\prime},\mathcal{G}_{\sigma_1}^{X\prime},\ldots,\mathcal{G}_{\sigma_K}^{X\prime}\right)'$$

3. **Apply normalizations**
   - For $i = 1,\ldots,K(2D+1)$ apply power normalization
     - $\left[\mathcal{G}_\lambda^X\right]_i \leftarrow \mathrm{sign}\left(\left[\mathcal{G}_\lambda^X\right]_i\right)\sqrt{\left|\left[\mathcal{G}_\lambda^X\right]_i\right|}$
   - Apply ℓ_2-normalization:

$$\mathcal{G}_\lambda^X = \mathcal{G}_\lambda^X / \sqrt{\mathcal{G}_\lambda^{X\prime}\mathcal{G}_\lambda^X}$$

This is closely related to the gradient with respect to the mixture weight $\mathcal{G}_{a_k}^X$ in th FV framework, see Equation (16). The difference is that $\mathcal{G}_{a_k}^X$ is mean-centered and normalized by the coefficient $\sqrt{w_k}$. Hence, for the same visual vocabulary size K, the FV contains significantly more information by including the gradients with respect to the means and standard deviations. Especially, the BoV is only K dimensional while the dimension of the FV is $(2D+1)K$. Conversely, we will show experimentally that, for a given feature dimensionality, the FV usually leads to results which are as good – and sometimes significantly better – than the BoV. However, in such a case the FV is much faster to compute than the BoV since it relies on significantly smaller visual vocabularies. An additional advantage is that the FV is a more principled approach than the BoV to combine the generative and discriminative worlds. For instance, it was shown in (Jaakkola and Haussler, 1998) (see Theorem 1) that if the classification label is included as a latent variable of the generative model u_λ, then the FK derived from this model is, asymptotically, never inferior to the MAP decision rule for this model.

Relationship with GMM-based representations. Several works proposed to model an image as a GMM adapted

from a universal (*i.e.* image-independent) distribution u_λ (Liu and Perronnin, 2008, Yan et al, 2008). Initializing the parameters of the GMM to λ and performing one EM iteration leads to the following estimates $\hat{\lambda}$ for the image GMM parameters:

$$\hat{w}_k = \frac{\sum_{t=1}^T \gamma_t(k) + \tau}{T + K\tau} \quad (35)$$

$$\hat{\mu}_k = \frac{\sum_{t=1}^T \gamma_t(k) x_t + \tau\mu_k}{\sum_{t=1}^T \gamma_t(k) + \tau} \quad (36)$$

$$\hat{\sigma}_k^2 = \frac{\sum_{t=1}^T \gamma_t(k) x_t^2 + \tau\left(\sigma_k^2 + \mu_k^2\right)}{\sum_{t=1}^T \gamma_t(k) + \tau} - \hat{\mu}_k^2 \quad (37)$$

where τ is a parameter which strikes a balance between the prior "universal" information contained in λ and the image-specific information contained in X. It is interesting to note that the FV and the adapted GMM encode essentially the same information since they both include statistics of order 0, 1 and 2: compare equations (35-37) with (31-33) in Algorithm 1, respectively. A major difference is that the FV provides a vectorial representation which is more amenable to large-scale processing than the GMM representation.

Relationship with the Vector of Locally Aggregated Descriptors (VLAD). The VLAD was proposed in Jégou et al (2010). Given a visual codebook learned with k-means, and a set of descriptors $X = \{x_t, t = 1, \ldots, T\}$ the VLAD consists in assigning each descriptor x_t to its closest codebook entry and in summing for each codebook entry the mean-centered descriptors. It was shown in Jégou et al (2012) that the VLAD is a simplified version of the FV under the following approximations: 1) the soft assignment is replaced by a hard assignment and 2) only the gradient with respect to the mean is considered. As mentioned in Jégou et al (2012), the same normalization steps which were introduced for the FV – the square-root and ℓ_2-normalization – can also be applied to the VLAD with significant improvements.

Relationship with the Super Vector (SV). The SV was proposed in Zhou et al (2010) and consists in concatenating in a weighted fashion a BoV and a VLAD (see equation (2) in their paper). To motivate the SV representation, Zhou *et al.* used an argument based on the Taylor expansion of non-linear functions which is similar to the one offered by Jaakkola and Haussler (1998) to justify the FK[4]. A major difference between the FV and the SV is that the latter one does not include any second-order statistics while the FV does in the gradient with respect to the variance. We will show in Section 3 that this additional term can bring substantial improvements.

Relationship with the Match Kernel (MK). The MK measures the similarity between two images as a sum of similarities between the individual descriptors (Haussler, 1999). If $X = \{x_t, t = 1, \ldots, T\}$ and $Y = \{y_u, u = 1, \ldots, U\}$ are two sets of descriptors and if $k(\cdot,\cdot)$ is a "base" kernel between local descriptors, then the MK between the sets X and Y is defined as:

$$K_{MK}(X,Y) = \frac{1}{TU}\sum_{t=1}^T\sum_{u=1}^U k(x_t, y_u). \quad (38)$$

The original FK without ℓ_2- or power-normalization is a MK if one chooses the following base kernel:

$$k_{FK}(x_t, y_u) = \varphi_{FK}(x_t)'\varphi_{FK}(y_u), \quad (39)$$

A disadvantage of the MK is that by summing the contributions of all pairs of descriptors, it tends to overcount multiple matches and therefore it cannot cope with the burstiness effect. We believe this is one of the reasons for the poor performance of the MK (see the third entry in Table 4 in the next section). To cope with this effect, alternatives have been proposed such as the "sum-max" MK of (Wallraven et al, 2003):

$$K_{SM}(X,Y) = \frac{1}{T}\sum_{t=1}^T \max_{u=1}^U k(x_t, y_u) + \frac{1}{U}\sum_{u=1}^U \max_{t=1}^T k(x_t, y_u). \quad (40)$$

or the "power" MK of (Lyu, 2005):

$$K_{POW}(X,Y) = \frac{1}{T}\frac{1}{U}\sum_{t=1}^T\sum_{u=1}^U k(x_t, y_u)^\rho. \quad (41)$$

In the FK case, we addressed the burstiness effect using the square-root normalization (see Section 2.3).

Another issue with the MK is its high computational cost since, in the general case, the comparison of two images requires comparing every pair of descriptors. While efficient approximation exists for the original (poorly performing) MK of equation (38) when there exists an explicit embedding of the kernel $k(\cdot,\cdot)$ (Bo and Sminchisescu, 2009), such approximations do not exist for kernels such as the one defined in (Lyu, 2005, Wallraven et al, 2003).

3 Small-scale experiments

The purpose of this section is to establish the FV as a state-of-the-art image representation before moving to larger scale scenarios. We first describe the experimental setup. We then provide detailed experiments on PASCAL VOC 2007. We also report results on Caltech256 and SUN397.

[4] See appendix A.2 in the extended version of Jaakkola and Haussler (1998) which is available at: http://people.csail.mit.edu/tommi/papers/gendisc.ps

3.1 Experimental setup

Images are resized to 100K pixels if larger. We extract approximately 10K descriptors per image from 24×24 patches on a regular grid every 4 pixels at 5 scales. We consider two types of patch descriptors in this work: the 128-dim SIFT descriptors of Lowe (2004) and the 96-dim Local Color Statistic (LCS) descriptors of Clinchant et al (2007). In both cases, unless specified otherwise, they are reduced down to 64-dim using PCA, so as to better fit the diagonal covariance matrix assumption. We will see that the PCA dimensionality reduction is key to make the FV work. We typically use in the order of 10^6 descriptors to learn the PCA projection.

To learn the parameters of the GMM, we optimize a Maximum Likelihood criterion with the EM algorithm, using in the order of 10^6 (PCA-reduced) descriptors. In Appendix B we provide some details concerning the implementation of the training GMM.

By default, for the FV computation, we compute the gradients with respect to the mean and standard deviation parameters only (but not the mixture weight parameters). In what follows, we will compare the FV with the soft-BoV histogram. For both experiments, we use the exact same GMM package which makes the comparison completely fair. For the soft-BoV, we perform a square-rooting of the BoV (which is identical to the power-normalization of the FV) as this leads to large improvements at negligible additional computational cost (Perronnin et al, 2010b, Vedaldi and Zisserman, 2010). For both the soft-BoV and the FV we use the same spatial pyramids with $R = 4$ regions (the entire images and three horizontal stripes) and we ℓ_2-normalized the per-region sub-vectors.

As for learning, we employ linear SVMs and train them using Stochastic Gradient Descent (SGD) (Bottou, 2011).

3.2 PASCAL VOC 2007

We first report a set of detailed experiments on PASCAL VOC 2007 (Everingham et al, 2007). Indeed, VOC 2007 is small enough (20 classes and approximately 10K images) to enable running a large number of experiments in a reasonable amount of time but challenging enough (as shown in (Torralba and Efros, 2011)) so that the conclusions we draw from our experiments extrapolate to other (equally challenging) datasets. We use the standard protocol which consists in training and validating on the "train" and "val" sets and testing on the "test" set. We measure accuracy using the standard measure on this dataset which is the interpolated Average Precision (AP). We report the average over 20 categories (mean AP or mAP) in %. In the following experiments, we use a GMM with 256 Gaussians, which results in 128K-dim FVs, unless otherwise specified.

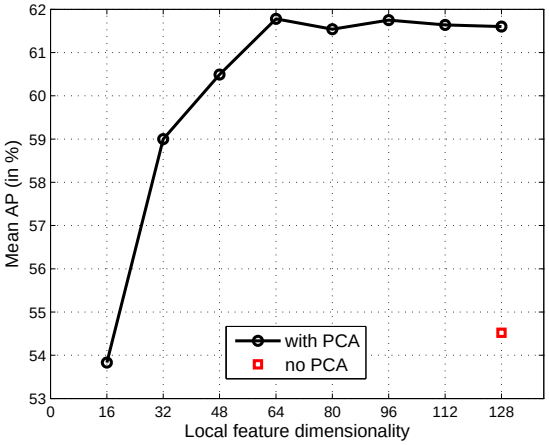

Fig. 1 Influence of the dimensionality reduction of the SIFT descriptors on the FV on PASCAL VOC 2007.

Table 1 Impact of the proposed modifications to the FK on PASCAL VOC 2007. "PN" = power normalization. "ℓ_2" = ℓ_2-normalization, "SP" = Spatial Pyramid. The first line (no modification applied) corresponds to the baseline FK of Perronnin and Dance (2007). Between parentheses: the absolute improvement with respect to the baseline FK. Accuracy is measured in terms of AP (in %).

PN	ℓ_2	SP	SIFT		LCS	
No	No	No	49.6		35.2	
Yes	No	No	57.9	(+8.3)	47.0	(+11.8)
No	Yes	No	54.2	(+4.6)	40.7	(+5.5)
No	No	Yes	51.5	(+1.9)	35.9	(+0.7)
Yes	Yes	No	59.6	(+10.0)	49.7	(+14.7)
Yes	No	Yes	59.8	(+10.2)	50.4	(+15.2)
No	Yes	Yes	57.3	(+7.7)	46.0	(+10.8)
Yes	Yes	Yes	61.8	(+12.2)	52.6	(+17.4)

Impact of PCA on local descriptors. We start by studying the influence of the PCA dimensionality reduction of the local descriptors. We report the results in Figure 1. We first note that PCA dimensionality reduction is key to obtain good results: without dimensionality reduction, the accuracy is 54.5% while it is above 60% for 48 PCA dimensions and more. Second, we note that the accuracy does not seem to be overly sensitive no the exact number of PCA components. Indeed, between 64 and 128 dimensions, the accuracy varies by less than 0.3% showing that the FV combined with a linear SVM is robust to noisy PCA dimensions. In all the following experiments, the PCA dimensionality is fixed to 64.

Impact of improvements. The goal of the next set of experiments is to evaluate the impact of the improvements over the original FK work of Perronnin and Dance (2007). This includes the use of the power-normalization, the ℓ_2-normalization, and SPs. We evaluate the impact of each of these three improvements considered separately, in pairs or all three together. Results are shown in Table 1 for SIFT

and LCS descriptors separately. The improved performance compared to the results in Perronnin et al (2010c), is probably due to denser sampling and a different layout of the spatial pyramids.

From the results we conclude the following. The single most important improvement is the power-normalization: +8.3 absolute for SIFT and +11.8 for LCS. On the other hand, the SP has little impact in itself: +1.9 on SIFT and +0.7 on LCS. Combinations of two improvements generally increase accuracy over a single one and combining all three improvements leads to an additional increment. Overall, the improvement is substantial: +12.2 for SIFT and +17.4 for LCS.

Approximate FIM *vs.* empirical FIM. We now compare the impact of using the proposed diagonal closed-form approximation of the Fisher Information Matrix (FIM) (see equations (16), (17) and (18) as well as Appendix A) as opposed to its empirical approximation as estimated on a training set. We first note that our approximation is based on the assumption that the distribution of posterior probabilities $\gamma_t(k)$ is sharply peaked. To verify this hypothesis, we computed on the "train" set of PASCAL VOC 2007 the value $\gamma_t^* = \max_k \gamma_t(k)$ for each observation x_t and plotted its cumulated distribution. We can deduce from Figure 2 that the distribution of the posterior probabilities is quite sharply peaked. For instance, more than 70% of the local descriptors have a $\gamma_t^* \geq 0.5$, *i.e.* the majority of the posterior is concentrated in a single Gaussian. However this is still far from the $\gamma_t^* = 1$ assumption we made for the approximated FIM. Nevertheless, in practice, this seems to have little impact on the accuracy: using the diagonal approximation of the FIM we get 61.8% accuracy while we get 60.6% with the empirical diagonal estimation. Note that we do not claim that this difference is significant nor that the closed-form approximation is superior to the empirical one in general. Finally, the FIM could be approximated by the identity matrix, as originally proposed in Jaakkola and Haussler (1998). Using the identity matrix, we observe a decrease of the performance to 59.8%.

Impact of patch density. In Section 2.3, it was hypothesized that the power-norm counterbalanced the effect of the incorrect patch independence assumption. The goal of the following experiment is to validate this claim by studying the influence of the patch density on the classification accuracy. Indeed, patches which are extracted more densely overlap more and are therefore more correlated. Conversely, if patches are extracted less densely, then the patch independence assumption is more correct. We vary the patch extraction step size from 4 pixels to 24 pixels. Since the size of our patches is 24×24, this means that we vary the overlap between two neighboring patches between more than 80% down to 0%. Results are shown in Table 2 for SIFT and LCS descriptors separately. As the step-size decreases, *i.e.* as the independence assumption gets more and more violated, the impact of the power-norm increases. We believe that this observation validates our hypothesis: the power-norm is a simple way to correct for the independence assumption

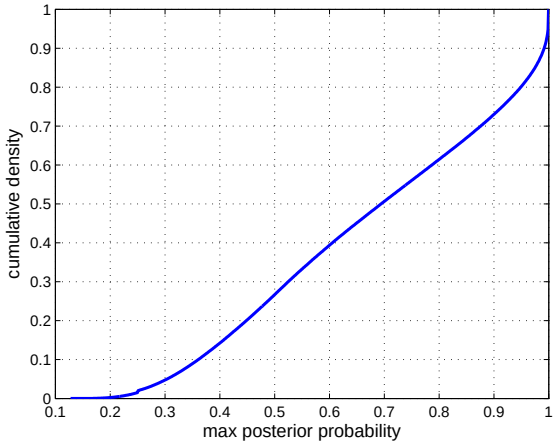

Fig. 2 Cumulative distribution of the max of the posterior probability $\gamma_t^* = \max_k \gamma_t(k)$ on PASCAL VOC 2007 for SIFT descriptors.

Table 2 Impact of the patch extraction step-size on PASCAL VOC 2007. The patch size is 24×24. Hence, when the step-size is 24, there is no overlap between patches. We also indicate the approximate number of patches per image for each step size. "PN" stands for Power Normalization. Δ abs. and Δ rel. are respectively the absolute and relative differences between using PN and not using PN. Accuracy is measured in terms of mAP (in %).

Step size	24	12	8	4
Patches per image	250	1,000	2,300	9,200
SIFT				
PN: No	51.1	55.8	57.0	57.3
PN: Yes	52.9	58.1	60.3	61.8
Δ abs.	1.8	2.3	3.3	4.5
Δ rel.	3.5	4.1	5.8	7.9
LCS				
PN: No	42.9	45.8	46.2	46.0
PN: Yes	46.7	50.4	51.2	52.6
Δ abs.	3.8	4.6	5.0	6.6
Δ rel.	8.9	10.0	10.8	14.3

Impact of cropping. In Section 2.3, we proposed to ℓ_2-normalize FVs and we provided two possible arguments. The first one hypothesized that the ℓ_2-normization is a way to counterbalance the influence of variable amounts of "informative patches" in an image where a patch is considered non-informative if it appears frequently (in any image). The second argument hypothesized that the ℓ_2 normalization of high-dimensional vectors is always beneficial when used in combination with linear classifiers.

The goal of the following experiment is to validate (or invalidate) the first hypothesis: we study the influence of the ℓ_2-norm when focusing on informative patches. One practical difficulty is the choice of informative patches. As shown

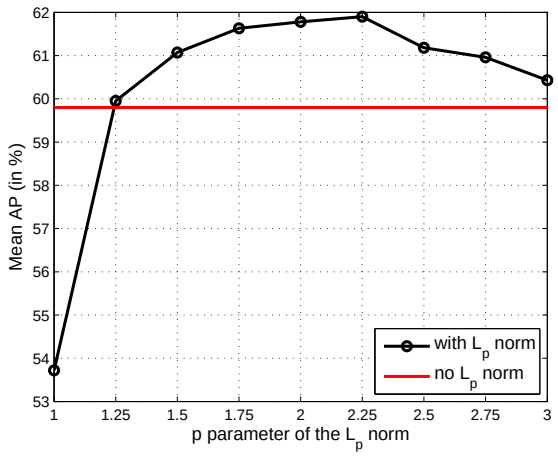

Fig. 3 Influence of the parameter p of the ℓ_p-norm on the FV on PASCAL VOC 2007.

∇	MAP (in %)
w	46.9
μ	57.9
σ	59.6
$\mu\sigma$	61.8
wμ	58.1
wσ	59.6
w$\mu\sigma$	61.8

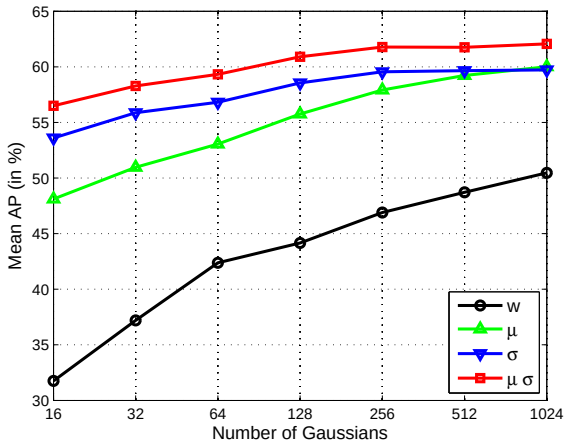

Fig. 4 Accuracy of the FV as a function of the gradient components on PASCAL VOC 2007 with SIFT descriptors only. w = gradient with respect to mixture weights, μ = gradient with respect to means and σ = gradient with respect to standard deviations. Top: accuracy for 256 Gaussians. Bottom: accuracy as a function of the number of Gaussians (we do not show $w\mu$, $w\sigma$ and $w\mu\sigma$ for clarity as there is little difference respectively with μ, σ and $\mu\sigma$).

in Uijlings et al (2009), foreground patches (*i.e.* object patches) are more informative than background object patches. Therefore, we carried-out experiments on cropped object images as a proxy to informative patches. We cropped the PASCAL VOC images to a single object (drawn randomly from the ground-truth bounding box annotations) to avoid the bias toward images which contain many objects. When using all improvements of the FV, we obtain an accuracy of 64.4% which is somewhat better than the 61.8% we report on full images. If we do not use the ℓ_2-normalization of the FVs, then we obtain an accuracy of 57.2%. This shows that the ℓ_2-normalization still has a significant impact on cropped objects which seems to go against our first argument and to favor the second one.

Impact of p in ℓ_p-norm. In Section 2.3, we proposed to use the ℓ_2-norm as opposed to any ℓ_p-norm because it was more consistent with our choice of a linear classifier. We now study the influence of this parameter p. Results are shown in Figure 3. We see that the ℓ_p-normalization improves over no normalization over a wide range of values of p and that the highest accuracy is achieved with a p close to 2. In all the following experiments, we set $p = 2$.

Impact of different Fisher vector components. We now evaluate the impact of the different components when computing the FV. We recall that the gradient with respect to the mixture weights, mean and standard deviation correspond respectively to 0-order, 1st-order and 2nd-order statistics and that the gradient with respect to the mixture weight corresponds to the soft-BoV. We see in Figure 4 that there is an increase in performance from 0-order (BoV) to the combination of 0-order and 1st-order statistics (similar to the statistics used in the SV(Zhou et al, 2010)), and even further when the 1st-order and 2nd-order statistics are combined. We also observe that the 0-order statistics add little discriminative information on top of the 1st-order and 2nd-order statistics. We also can see that the 2nd-order statistics seem to bring more information than the 1st-order statistics for a small number of Gaussians but that both seem to carry similar information for a larger number of Gaussians.

Comparison with the soft-BoV. We now compare the FV to the soft-BoV. We believe this comparison to be completely fair, since we use the same low-level SIFT features and the same GMM implementation for both encoding methods. We show the results in Figure 5 both as a function of the number of Gaussians of the GMM and as a function of the feature dimensionality (note that the SP increases the dimensionality for both FV and BoV by a factor 4). The conclusions are the following ones. For a given number of Gaussians, the FV always significantly outperforms the BoV. This is not surprising since, for a given number of Gaussians, the dimensionality of the FV is much higher than that of the BoV. The difference is particularly impressive for a small number of Gaussians. For instance for 16 Gaussians, the BoV obtains 31.8 while the FV gets 56.5. For a given number of dimensions, the BoV performs slightly better for a small number of dimensions (512) but the FV performs better for a large number of dimensions. Our best

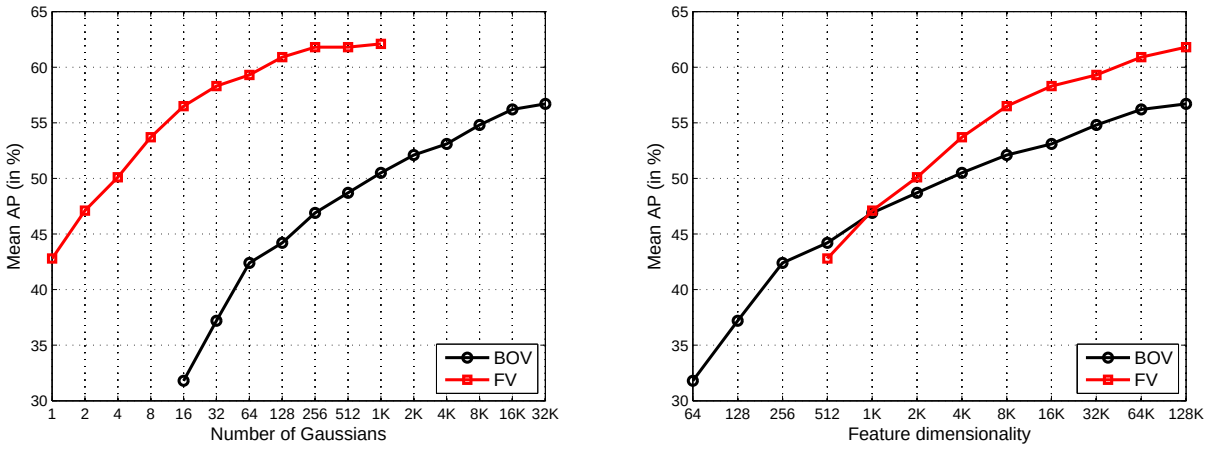

Fig. 5 Accuracy of the soft-BoV and the FV as a function of the number of Gaussians (left) and feature dimensionality (right) on PASCAL VOC 2007 with SIFT descriptors only.

Table 3 Comparison with the state-of-the-art on PASCAL VOC 2007.

Algorithm	MAP (in %)
challenge winners	59.4
(Uijlings et al, 2009)	59.4
(VanGemert et al, 2010)	60.5
(Yang et al, 2009a)	62.2
(Harzallah et al, 2009)	63.5
(Zhou et al, 2010)	64.0
(Guillaumin et al, 2010)	66.7
FV (SIFT)	61.8
FV (LCS)	52.6
FV (SIFT + LCS)	63.9

results with the BoV is 56.7% with 32K Gaussians while the FV gets 61.8% with 256 Gaussians. With these parameters, the FV is approximately 128 times faster to compute since, by far, the most computationally intensive step for both the BoV and the GMM is the cost of computing the assignments $\gamma_k(x)$. We note that our soft-BoV baseline is quite strong since it outperforms the soft-BoV results in the recent benchmark of Chatfield et al (2011), and performs on par with the best sparse coding results in this benchmark. Indeed, Chatfield *et al*. report 56.3% for soft-BoV and 57.6% for sparse coding with a slightly different setting.

Comparison with the state-of-the-art. We now compare our results to some of the best published results on PASCAL VOC 2007. The comparison is provided in Table 3. For the FV, we considered results with SIFT only and with a late fusion of SIFT+LCS. In the latter case, we trained two separate classifiers, one using SIFT FVs and one using LCS FVs. Given an image, we compute the two scores and average them in a weighted fashion. The weight was cross-validated on the validation data and the optimal combination was found to be $0.6 \times \text{SIFT} + 0.4 \times \text{LCS}$. The late fusion of the SIFT and LCS FVs yields a performance of 63.9%, using only the SIFT features obtains 61.8%. We now provide more details on the performance of the other published methods.

The challenge winners obtained 59.4% accuracy by combining many different channels corresponding to different feature detectors and descriptors. The idea of combining multiple channels on PASCAL VOC 2007 has been extensively used by others. For instance, VanGemert et al (2010) reports 60.5% with a soft-BoV representation and several color descriptors and Yang et al (2009a) reports 62.2% using a group sensitive form of Multiple Kernel Learning (MKL). Uijlings et al (2009) reports 59.4% using a BoV representation and a single channel but assuming that one has access to the ground-truth object bounding box annotations at both training and test time (which they use to crop the image to the rectangles that contain the objects, and thus suppress the background to a large extent). This is a restrictive setting that cannot be followed in most practical image classification problems. Harzallah et al (2009) reports 63.5% using a standard classification pipeline in combination with an image detector. We note that the cost of running one detector per category is quite high: from several seconds to several tens of seconds per image. Zhou et al (2010) reports 64.0% with SV representations. However, with our own re-implementation, we obtained only 58.1% (this corresponds to the line $w\mu$ in the table in Figure 4. The same issue was noted in Chatfield et al (2011). Finally, Guillaumin et al (2010) reports 66.7% but assuming that one has access to the image tags. Without access to such information, their BoV results dropped to 53.1%.

Computational cost. We now provide an analysis of the computational cost of our pipeline on PASCAL VOC 2007. We focus on our "default" system with SIFT descriptors

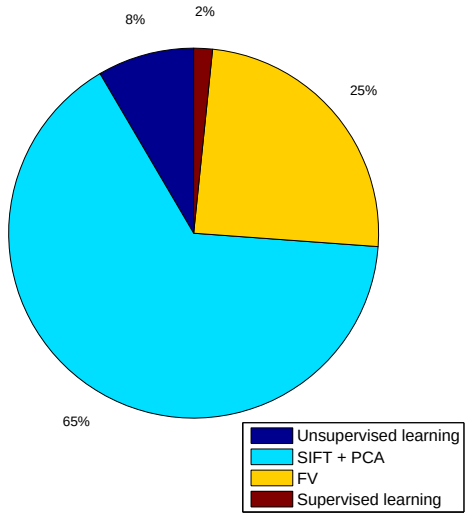

Fig. 6 Breakdown of the computational cost of our pipeline on PASCAL VOC 2007. The whole pipeline takes approx. 2h on a single processor, and is divided into: 1) Learning the PCA on the SIFT descriptors and the GMM with 256 Gaussians (Unsupervised learning). 2) Computing the dense SIFT descriptors for the 10K images and projecting them to 64 dimensions (SIFT + PCA). 3) Encoding and aggregating the low-level descriptors into FVs for the 10K images (FV). 4) Learning the 20 SVM classifiers using SGD (Supervised Learning). The testing time – *i.e.* the time to classify the 5K test FVs – is not shown as it represents only 0.1% of the total computational cost.

only and 256 Gaussians (128K-dim FVs). Training and testing the whole pipeline from scratch on a Linux server with an Intel Xeon E5-2470 Processor @2.30GHz and 128GBs of RAM takes approx. 2h using a single processor. The repartition of the cost is shown in Figure 6. From this breakdown we observe that $\frac{2}{3}$ of the time is spent on computing the low-level descriptors for the train, val and test sets. Encoding the low-level descriptors into image signatures costs about 25% of the time, while learning the PCA and the parameters of the GMM takes about 8%. Finally learning the 20 SVM classifiers using the SGD training takes about 2% of the time and classification of the test images is in the order of seconds (0.1% of the total computational cost).

3.3 Caltech 256

We now report results on Caltech 256 which contains approx. 30K images of 256 categories. As is standard practice, we run experiments with different numbers of training images per category: 5, 10, 15, ..., 60. The remainder of the images is used for testing. To cross-validate the parameters, we use half of the training data for training, the other half for validation and then we retrain with the optimal parameters on the full training data. We repeat the experiments 10 times. We measure top-1 accuracy for each class and report the average as well as the standard deviation. In Figure 7(a), we compare a soft-BoV baseline with the FV (using only SIFT descriptors) as a function of the number of training samples. For the soft-BoV, we use 32K Gaussians and for the FV 256 Gaussians. Hence both the BoV and FV representations are 128K-dimensional. We can see that the FV always outperforms the BoV.

We also report results in Table 4 and compare with the state-of-the-art. We consider both the case where we use only SIFT descriptors and the case where we use both SIFT and LCS descriptors (again with a simple weighted linear combination). We now provide more details about the different techniques. The baseline of Griffin et al (2007) is a reimplementation of the spatial pyramid BoV of Lazebnik et al (2006). Several systems are based on the combination of multiple channels corresponding to many different features including (Bergamo and Torresani, 2012, Boiman et al, 2008, Gehler and Nowozin, 2009, VanGemert et al, 2010). Other works, considered a single type of descriptors, typically SIFT descriptors (Lowe, 2004). Bo and Sminchisescu (2009) make use of the Efficient Match Kernel (EMK) framework which embeds patches in a higher-dimensional space in a non-linear fashion (see also Section 2.5). Wang et al (2010), Yang et al (2009b) considered different variants of sparse coding and Boureau et al (2011), Feng et al (2011) different spatial pooling strategies. Kulkarni and Li (2011) extracts on the order of a million patches per image by computing SIFT descriptors from several affine transforms of the original image and uses sparse coding in combination with Adaboost. Finally, the best results we are aware of are those of Bo et al (2012) which uses a deep architecture which stacks three layers, each one consisting of three steps: coding, pooling and contrast normalization. Note that the deep architecture of Bo et al (2012) makes use of color information. Our FV which combines the SIFT and LCS descriptors, outperform all other methods using any number of training samples. Also the SIFT only FV is among the best performing descriptors.

3.4 SUN 397

We now report results on the SUN 397 dataset (Xiao et al, 2010) which contains approx. 100K images of 397 categories. Following the protocol of Xiao et al (2010), we used 5, 10, 20 or 50 training samples per class and 50 samples per class for testing. To cross-validate the classifier parameters, we use half of the training data for training, the other half for validation and then we retrain with the optimal parameters on the full training data[5]. We repeat the experiments 10 times using the partitions provided at the website of the

[5] Xiao et al (2010) also report results with 1 training sample per class. However, a single sample does not provide any way to perform cross-validation which is the reason why we do not report results in this setting.

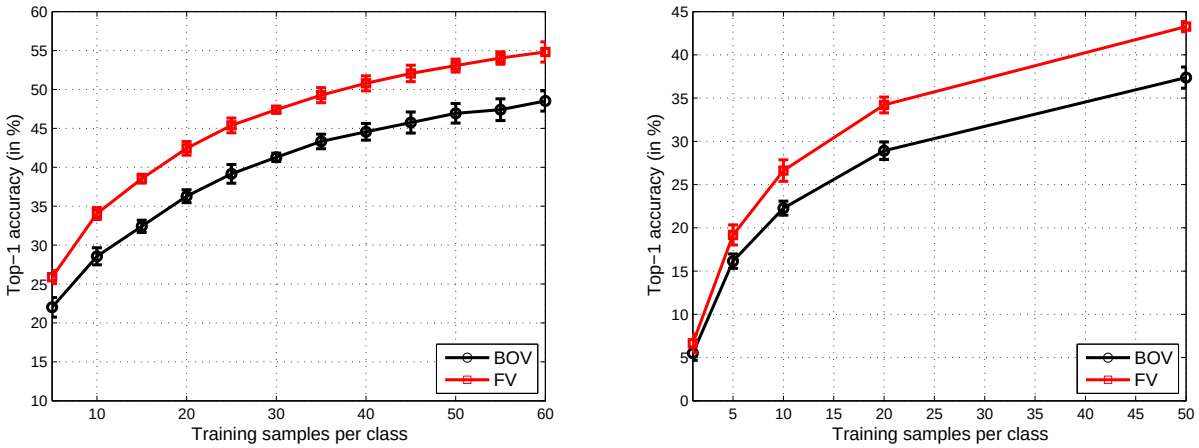

Fig. 7 Comparison of the soft-BoV and the FV on Caltech256 (left) and SUN 397 (right) as a function of the number of training samples. We only use SIFT descriptors and report the mean and 3 times the average deviation.

Table 4 Comparison of the FV with the state-of-the-art on Caltech 256.

Method	ntrain=15	ntrain=30	ntrain=45	ntrain=60
Griffin et al (2007)	-	34.1 (0.2)	-	-
Boiman et al (2008)	-	42.7 (-)	-	-
Bo and Sminchisescu (2009)	23.2 (0.6)	30.5 (0.4)	34.4 (0.4)	37.6 (0.5)
Yang et al (2009b)	27.7 (0.5)	34.0 (0.4)	37.5 (0.6)	40.1 (0.9)
Gehler and Nowozin (2009)	34.2 (-)	45.8 (-)	-	-
VanGemert et al (2010)	-	27.2 (0.4)	-	-
Wang et al (2010)	34.4 (-)	41.2 (-)	45.3 (-)	47.7 (-)
Boureau et al (2011)	-	41.7 (0.8)	-	-
Feng et al (2011)	35.8 (-)	43.2 (-)	47.3 (-)	-
Kulkarni and Li (2011)	39.4 (-)	45.8 (-)	49.3 (-)	51.4 (-)
Bergamo and Torresani (2012)	39.5 (-)	45.8 (-)	-	-
Bo et al (2012)	40.5 (0.4)	48.0 (0.2)	51.9 (0.2)	55.2 (0.3)
FV (SIFT)	38.5 (0.2)	47.4 (0.1)	52.1 (0.4)	54.8 (0.4)
FV (SIFT+LCS)	41.0 (0.3)	49.4 (0.2)	54.3 (0.3)	57.3 (0.2)

Table 5 Comparison of the FV with the state-of-the-art on SUN 397.

Method	ntrain=5	ntrain=10	ntrain=20	ntrain=50
Xiao et al (2010)	14.5	20.9	28.1	38.0
FV (SIFT)	19.2 (0.4)	26.6 (0.4)	34.2 (0.3)	43.3 (0.2)
FV (SIFT+LCS)	21.1 (0.3)	29.1 (0.3)	37.4 (0.3)	47.2 (0.2)

dataset.[6] We measure top-1 accuracy for each class and report the average as well as the standard deviation. As was the case for Caltech 256, we first compare in Figure 7(b), a soft-BoV baseline with 32K Gaussians and the FV with 256 Gaussians using only SIFT descriptors. Hence both the BoV and FV representations have the same dimensionality: 128K-dim. As was the case on the PASCAL VOC and Caltech datasets, the FV consistently outperforms the BoV and the performance difference increases when more training samples are available.

The only other results we are aware of on this dataset are those of its authors whose system combined 12 feature types (Xiao et al, 2010). The comparison is reported in Table 5. We observe that the proposed FV performs significantly better than the baseline of Xiao et al (2010), even when using only SIFT descriptors.

[6] See http://people.csail.mit.edu/jxiao/SUN/

4 Fisher vector compression with PQ codes

Having now established that the FV is a competitive image representation, at least for small- to medium-scale problems, we now turn to the large-scale challenge.

One of the major issues to address when scaling the FV to large amounts of data is the memory usage. As an example, in Sánchez and Perronnin (2011) we used FV representations with up to 512K dimensions. Using a 4 byte floating point representation, a single signature requires 2MB of storage. Storing the signatures for the approx. 1.4M images of the ILSVRC 2010 dataset (Berg et al, 2010) would take almost 3TBs, and storing the signatures for the approx. 14M of the full ImageNet dataset (Deng et al, 2009) around 27TBs. We underline that this is not purely a storage problem. Handling TBs of data makes experimentation very difficult if not impractical. Indeed, much more time can be spent writing / reading data on disk than performing any useful calculation.

In what follows, we first introduce Product Quantization (PQ) as an efficient and effective approach to perform lossy compression of FVs. We then describe a complementary lossless compression scheme based on sparsity encoding. Subsequently, we explain how PQ encoding / decoding can be combined with Stochastic Gradient Descent (SGD) learning for large-scale optimization. Finally, we provide a theoretical analysis of the effect of lossy quantization on the learning objective function.

4.1 Vector quantization and product quantization

Vector Quantization (VQ). A vector quantizer $q : \Re^E \rightarrow \mathscr{C}$ maps a vector $v \in \Re^E$ to a codeword $c_k \in \Re^E$ in the codebook $\mathscr{C} = \{c_k, k = 1, \ldots, K\}$ (Gray and Neuhoff, 1998). The cardinality K of the set $\mathscr{C}$, known as the codebook size, defines the compression level of the VQ as $\lceil \log_2 K \rceil$ bits are needed to identify the K codeword indices. If one considers the Mean-Squared Error (MSE) as the distortion measure then the Lloyd optimality conditions lead to k-means training of the VQ. The MSE for a quantizer q is given as the expected squared error between $v \in \Re^E$ and its reproduction value $q(v) \in \mathscr{C}$ (Jégou et al, 2011):

$$MSE(q) = \int p(v) \|q(v) - v\|^2 dv, \tag{42}$$

where p is a density function defined over the input vector space.

If we use on average b bits per dimension to encode a given image signature (b might be a fractional value), then the cardinality of the codebook is 2^{bE}. However, for $E = O(10^5)$, even for a small number of bits (*e.g.* our target in this work is typically $b = 1$), the cost of learning and storing such a codebook – in $O(E2^{bE})$ – would be incommensurable.

Product Quantization (PQ). A solution is to use product quantizers which were introduced as a principled way to deal with high dimensional input spaces (see *e.g.* Jégou et al (2011) for an excellent introduction to the topic). A PQ $q : \Re^E \rightarrow \mathscr{C}$ splits a vector v into a set of M distinct sub-vectors of size $G = E/M$, *i.e.* $v = [v_1, \ldots, v_M]$. M sub-quantizers $\{q_m, m = 1 \ldots M\}$ operate independently on each of the sub-vectors. If $\mathscr{C}_m$ is the codebook associated with q_m, then $\mathscr{C}$ is the Cartesian product $\mathscr{C} = \mathscr{C}_1 \times \ldots \times \mathscr{C}_M$ and $q(v)$ is the concatenation of the $q_m(v_m)$'s.

The v_m's being the orthogonal projections of v onto disjoint groups of dimensions, the MSE for PQ takes the form:

$$\begin{aligned} MSE_{pq}(q) &= \sum_m MSE(q_m) \\ &= \sum_m \int p_m(v_m) \|q(v_m) - v_m\|^2 dv_m, \end{aligned} \tag{43}$$

which can be equivalently rewritten as:

$$MSE_{pq}(q) = \int \left(\prod_{m'} p_{m'}(v_{m'}) \right) \sum_m \|q(v_m) - v_m\|^2 dv. \tag{44}$$

The sum within the integral corresponds to the squared distortion for q. The term between parentheses can be seen as an approximation to the underlying distribution:

$$p(v) \approx \prod_k p_k(v_k). \tag{45}$$

When $M = E$, *i.e.* $G = 1$, the above approximation corresponds to a naive Bayes model where all dimensions are assumed to be independent, leading to a simple scalar quantizer. When $M = 1$, *i.e.* $G = E$, we are back to (42), *i.e.* to the original VQ problem on the full vector. Choosing different values for M impose different independence assumptions on p. Particularly, for groups m and m' we have:

$$Cov(v_m, v_{m'}) = 0_{G \times G}, \quad \forall m \neq m' \tag{46}$$

where $0_{G \times G}$ denotes the $G \times G$ matrix of zeros. Using a PQ with M groups can be seen as restricting the covariance structure of the original space to a block diagonal form.

In the FV case, we would expect this structure to be diagonal since the FIM is just the covariance of the score. However: *i)* the normalization by the inverse of the FIM is only approximate; *ii)* the ℓ_2-normalization (Sec. 2.3) induces dependencies between dimensions, and *iii)* the diagonal covariance matrix assumption in the model is probably incorrect. All these factors introduce dependencies among the FV dimensions. Allowing the quantizer to model some correlations between groups of dimensions, in particular those that correspond to the same Gaussian, can at least partially account for the dependencies in the FV.

Let b be the average number of bits per dimension (assuming that the bits are equally distributed across the codebooks $\mathscr{C}_m$) . The codebook size of $\mathscr{C}$ is $K = (2^{bG})^M = 2^{bE}$ which is unchanged with respect to the standard VQ. However the costs of learning and storing the codebook are now in $O(E2^{bG})$.

The choice of the parameters b and G should be motivated by the balance we wish to strike between three conflicting factors: 1) the quantization loss, 2) the quantization speed and 3) the memory/storage usage. We use the following approach to make this choice in a principled way. Given a memory/storage target, we choose the highest possible number of bits per dimension b we can afford (constraint 3). To keep the quantization cost reasonable we have to cap the value bG. In practice we choose G such that $bG \leq 8$ which ensures that (at least in our implementation) the cost of encoding a FV is not higher than the cost of extracting the FV itself (constraint 2). Obviously, different applications might have different constraints.

4.2 FV sparsity encoding

We mentioned earlier that the FV is dense: on average, only approximately 50% of the dimensions are zero (see also the paragraph "posterior thresholding" in Appendix B). Generally speaking, this does not lead to any gain in storage as encoding the index and the value for each dimension would take as much space (or close to). However, we can leverage the fact that the zeros are not randomly distributed in the FV but appear in a structure. Indeed, if no patch was assigned to Gaussian i (i.e. $\forall t$, $\gamma_t(i) = 0$), then in equations (17) and (18) all the gradients are zero. Hence, we can encode the sparsity on a per-Gaussian level instead of doing so per dimension.

The sparsity encoding works as follows. We add one bit per Gaussian. This bit is set to 0 if no low-level feature is assigned to the Gaussian, and 1 if at least one low-level feature is assigned to the Gaussian (with non-negligible probability). If this bit is zero for a given Gaussian, then we know that all the gradients for this Gaussian are exactly zero and therefore we do not need to encode the codewords for the sub-vectors of this Gaussian. If the bit is 1, then we encode the $2D$ mean and standard-deviation gradient values of this Gaussian using PQ.

Note that adding this per Gaussian bit can be viewed as a first step towards gain/shape coding (Sabin and Gray, 1984), *i.e.* encoding separately the norm and direction of the gradient vectors. We experimented with a more principled approach to gain/shape coding but did not observe any substantial improvement in terms of storage reduction.

4.3 SGD Learning with quantization

We propose to learn the linear classifiers directly in the uncompressed high-dimensional space rather than in the space of codebook indices. We therefore integrate the decompression algorithm in the SGD training code. All compressed signatures are kept in RAM if possible. When a signature is passed to the SGD algorithm, it is decompressed on the fly. This is an efficient operation since it only requires look-up table accesses. Once it has been processed, the decompressed version of the sample is discarded. Hence, only one decompressed sample at a time is kept in RAM. This makes our *learning scheme both efficient and scalable*.

While the proposed approach combines on-the-fly decompression with SGD learning, an alternative has been recently proposed by Vedaldi and Zisserman (2012) which avoids the decompression step and which leverages bundle methods with a non-isotropic regularizer. The latter method, however, is a batch solver that accesses all data for every update of the weight vector, and is therefore less suitable for large scale problems. The major advantage of our SGD-based approach is that we decompress only one sample at a time, and typically do not even need to access the complete dataset to obtain good results. Especially, we can sample only a fraction of the negatives and still converge to a reasonably accurate solution. This proves to be a crucial property when handling very large datasets such as ImageNet10K, see Section 5.

4.4 Analysis of the effect of quantization on learning

We now analyze the influence of the quantization on the classifier learning. We will first focus on the case of Vector Quantization and then turn to PQ.

Let $f(x;w) : \mathbb{R}^D \rightarrow \mathbb{R}$ be the prediction function. In what follows, we will focus on the linear case, i.e. $f(x;w) = w'x$. We assume that, given a sample (x,y) with $x \in \mathbb{R}^D$ and $y \in \{-1,+1\}$, we incur a loss:

$$\ell(yf(x;w)) = \ell(yw'x). \tag{47}$$

We assume that the training data is generated from a distribution p. In the case of an unregularized formulation, we typically seek w that minimizes the following expected loss:

$$L(w) = \int_{x,y} \ell(yw'x)p(x,y)dxdy. \tag{48}$$

Underlying the k-means algorithm used in VQ (and PQ) is the assumption that the data was generated by a Gaussian Mixture Model (GMM) with equal mixture weights and isotropic covariance matrices, *i.e.* covariance matrices which can be written as $\sigma^2 I$ where I is the identity matrix.[7] If

[7] Actually, any continuous distribution can be approximated with arbitrary precision by a GMM with isotropic covariance matrices.

we make this assumption, we can write (approximately) a random variable $x \sim p$ as the sum of two independent random variables: $x \approx q + \varepsilon$ where q draws values in the finite set of codebook entries $\mathscr{C}$ with equal probabilities and $\varepsilon \sim \mathscr{N}(0, \sigma^2 I)$ is a white Gaussian noise. We can therefore approximate the objective function (48) as[8]:

$$L(w) \approx \int_{q,\varepsilon,y} \ell(yw'(q+\varepsilon))p(q,\varepsilon,y)dqd\varepsilon dy \tag{49}$$

We further assume that the loss function $\ell(u)$ is twice differentiable. While this is not true of the hinge loss in the SVM case, this assumption is verified for other popular losses such as the quadratic loss or the log loss. If σ^2 is small, we can approximate $\ell(yw'(q+\varepsilon))$ by its second order Taylor expansion around q:

$$\begin{aligned} &\ell(yw'(q+\varepsilon)) \\ &\approx \ell(yw'q) + \varepsilon'\nabla_q \ell(yw'q) + \frac{1}{2}\varepsilon'\nabla_q^2 \ell(yw'q)\varepsilon \\ &= \ell(yw'q) + \varepsilon' yw\ell'(yw'q) + \frac{1}{2}\varepsilon' ww'\varepsilon\ell''(yw'q). \end{aligned} \tag{50}$$

where $\ell'(u) = \partial\ell(u)/\partial u$ and $\ell''(u) = \partial^2\ell(u)/(\partial u)^2$ and we have used the fact that $y^2 = 1$. Note that this expansion is exact for the quadratic loss.

In what follows, we further make the assumption that the label y is independent of the noise ε knowing q, *i.e.* $p(y|q,\varepsilon) = p(y|q)$. This means that the label y of a sample x is fully determined by its quantization q and that ε can be viewed as a noise. For instance, in the case where $\sigma \to 0$ – *i.e.* the soft assignment becomes hard and each codeword is associated with a Voronoi region – this conditional independence means that (the distribution on) the label is constant over each Voronoi region. In such a case, using also the independence of q and ε, *i.e.* the fact that $p(q,\varepsilon) = p(q)p(\varepsilon)$, it is easily shown that:

$$p(q,\varepsilon,y) = p(q,y)p(\varepsilon). \tag{51}$$

If we inject (50) and (51) in (49), we obtain:

$$\begin{aligned} L(w) \approx &\int_{q,\varepsilon,y} \ell(yw'q)p(q,y)dqdy \\ &+ \int_\varepsilon \varepsilon' p(\varepsilon)d\varepsilon \int_{q,y} yw\ell'(yw'q)p(q,y)dqdy \\ &+ \frac{1}{2}w'\left(\int_\varepsilon \varepsilon\varepsilon' p(\varepsilon)d\varepsilon\right) w \int_q \ell''(yw'q)p(q)dq. \end{aligned} \tag{52}$$

Since $\varepsilon \sim \mathscr{N}(0, \sigma^2 I)$, we have:

$$\int_\varepsilon \varepsilon' p(\varepsilon)d\varepsilon = 0 \tag{53}$$

$$\int_\varepsilon \varepsilon\varepsilon' p(\varepsilon)d\varepsilon = \sigma^2 I. \tag{54}$$

[8] Note that since q draws values in a finite set, we could replace the $\int_q$ by $\sum_q$ in the following equations but we will keep the integral notation for simplicity.

Therefore, we can rewrite:

$$\begin{aligned} L(w) \approx &\int_{q,y} \ell(yw'q)p(q,y)dqdy \\ &+ \frac{\sigma^2}{2}||w||^2 \int_q \ell''(yw'q)p(q)dq \end{aligned} \tag{55}$$

The first term corresponds to the expected loss in the case where we replace each training sample by its quantized version. Hence, the previous approximation tells us that, up to the first order, the expected losses in the quantized and unquantized cases are approximately equal. This provides a strong justification for using k-means quantization when training linear classifiers. If we go to the second order, a second term appears. We now study its influence for two standard twice-differentiable losses: the quadratic and log losses respectively.

- In the case of the quadratic loss, we have $\ell(u) = (1-u)^2$ and $\ell''(u) = 2$. and the regularization simplifies to $\sigma^2||w^2||$, *i.e.* a standard regularizer. This result is in line with Bishop (1995) which shows that adding Gaussian noise can be a way to perform regularization for the quadratic loss. Here, we show that quantization actually has an *"unregularization"* effect since the loss in quantized case can be written approximately as the loss in the unquantized case *minus* a regularization term. Note that this unregularization effect could be counter-balanced in theory by cross-validating the regularization parameter λ.
- In the case of the log loss, we have $\ell(u) = -\log\sigma(u) = \log(1+e^{-u})$, where $\sigma(\cdot)$ is the sigmoid function, and $\ell''(z) = \sigma(u)\sigma(-u)$ which only depends on the absolute value of u. Therefore, the second term of (55) can be written as:

$$\frac{\sigma^2}{2}||w||^2 \int_q \sigma(w'q)\sigma(-w'q)p(q)dq \tag{56}$$

which depends on the data distribution $p(q)$ but does not depend on the label distribution $p(y|q)$. We can observe two conflicting effects in (56). Indeed, as the norm $||w||$ increases, the value of the term $\sigma(w'q)\sigma(-w'q)$ decreases. Hence, it is unclear whether this term acts as a regularizer or an "unregularizer". Again, this might depend on the data distribution. We will study empirically its effect in section 5.1.

To summarize, we have made three approximations: 1) p can be approximated by a mixture of isotropic Gaussians, 2) ℓ can be approximated by its second order Taylor expansion and 3) y is independent of ε knowing q. We note that these three approximations become more and more exact as the number of codebook entries K increases, *i.e.* as the variance σ^2 of the noise decreases.

We underline that the previous analysis remains valid in the PQ case since the codebook is a Cartesian product

of codebooks. Actually, PQ is an efficient way to increase the codebook size (and therefore reduce σ^2) at an affordable cost. Also, the previous analysis remains valid beyond Gaussian noise, as long as ε is independent of q and has zero mean. Finally, although we typically train SVM classifiers, *i.e.* we use a hinge loss, we believe that the intuitions gained from the twice differentiable losses are still valid, especially those drawn from the log-loss whose shape is similar.

5 Large-scale experiments

We now report results on the large-scale ILSVRC 2010 and ImageNet10K datasets. The FV computation settings are almost identical to those of the small-scale experiments. The only two differences are the following ones. First, we do not make use of Spatial Pyramids and extract the FVs on the whole images to reduce the signature dimensionality and therefore speed-up the processing. Second, because of implementation issues, we found it easier to extract one SIFT FV and one LCS FV per image and to concatenate them using an early fusion strategy before feeding them to the SVM classifiers (while in our previous experiments, we trained two classifiers separately and peformed late fusion of the classifier scores).

As for the SVM training, we also use SGD to train one-vs-rest linear SVM classifiers. Given the size of these datasets, at each pass of the SGD routine we sample all positives but only a random subset of negatives (Perronnin et al, 2012, Sánchez and Perronnin, 2011).

5.1 ILSVRC 2010

ILSVRC 2010 (Berg et al, 2010) contains approx. 1.4M images of 1K classes. We use the standard protocol which consists in training on the "train" set (1.2M images), validating on the "val" set (50K images) and testing on the "test" set (150K) images. We report the top-5 classification accuracy (in %) as is standard practice on this dataset.

Impact of compression parameters. We first study the impact of the compression parameters on the accuracy. We can vary the average number of bits per dimension b and the group size G. We show results on 4K-dim and 64K-dim FV features in Figure 8 (using respectively a GMM with 16 and 256 Gaussians). Only the training samples are compressed, not the test samples. In the case of the 4K FVs, we were able to run the uncompressed baseline as the uncompressed training set (approx. 19 GBs) could fit in the RAM of our servers. However, this was not possible in the case of the 64K-dim FVs (approx. 310 GBs). As expected, the accuracy increases with b: more bits per dimension lead to a better preservation of the information for a given G. Also, as expected, the accuracy increases with G: taking into account the correlation between the dimensions also leads to less loss of information for a given b. Note that by using $b = 1$ and $G = 8$, we reduce the storage by a factor 32 with a very limited loss of accuracy. For instance, for 4K-dim FVs, the drop with respect to the uncompressed baseline is only 2.1% (from 64.2 to 62.1).

Table 6 Memory required to store the ILSVRC 2010 training set using 4K-dim or 64K-dim FVs. For PQ, we used $b = 1$ and $G = 8$.

	Uncompressed	PQ	PQ + sparsity
4K-dim FV	19.0 GBs	610 MBs	540 MBs
64K-dim FV	310 GBs	9.6 GBs	6.3 GBs

We underline that the previous results only make use of PQ compression but do not include the sparsity compression described in section 4.2. We report in Table 6 the compression gains with and without sparsity compression for $b = 1$ and $G = 8$. We note that the impact of the sparsity compression is limited for 4K-dim FVs: around 10% savings. This is to be expected given the very tiny number of visual words in the GMM vocabulary in such a case (only 16). In the case of 64K-dim FVs, the gain from the sparsity encoding is more substantial: around 30%. This gain increases with the GMM vocabulary size and it would be also larger if we made use of spatial pyramids (this was verified experimentally through preliminary experiments). In what follows, except where the contrary is specified, we use as default compression parameters b =1 and G =8.

Compression and regularization. We now evaluate how compression impacts regularization, *i.e.* whether the optimal regularization parameter changes with compression. Since we want to compare systems with compressed and uncompressed features, we focus on the 4K-dim FVs. Results are shown in Figure 9 for three losses: the quadratic and log losses which are twice differentiable and which are discussed in section 4.4 and the hinge loss which corresponds to the SVM classifier but to which our theoretical analysis is not applicable. We also test two compression settings: our default setting with G =8 and b =1 and a lower-performing scalar quantization setting with G =1 and b =1. We can see that for all three losses the optimal regularization parameter is the same or very similar with and without compression. Especially, in the quadratic case, as opposed to what is suggested by our analysis, we do not manage to improve the accuracy with compressed features by cross-validating the regularization parameter. This might indicate that our analysis is too simple to represent real-world datasets and that we need to take into account more complex phenomena, *e.g.* by considering a more elaborate noise model or by including higher orders in the Taylor expansion.

K-nearest neighbors search with PQ compression. We now compare the effect of PQ compression on the linear SVM classifier to its effect on a k-Nearest Neighbors (k-

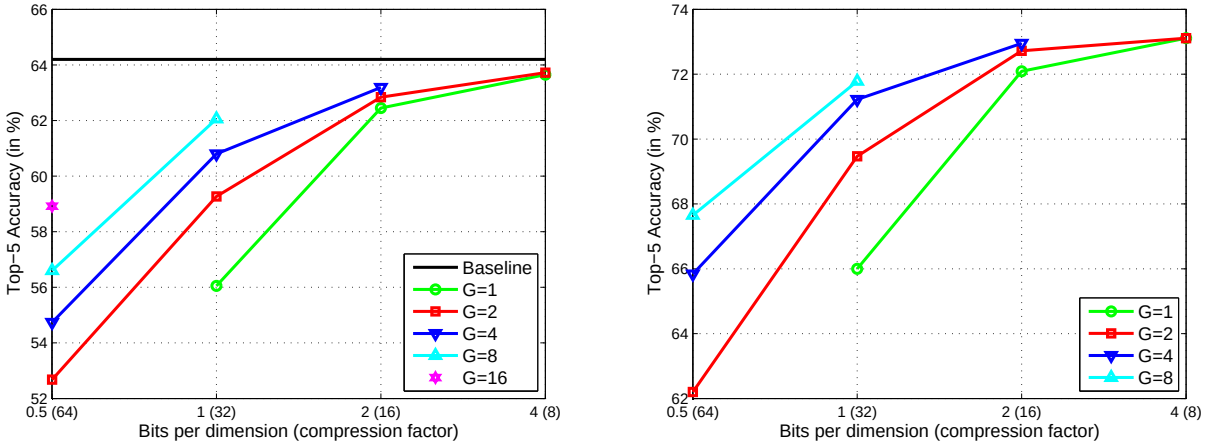

Fig. 8 Compression results on ILSVRC 2010 with 4K-dim FVs (left) and 64K-dim FVs when varying the number of bits per dimension b and the sub-vector dimensionality G.

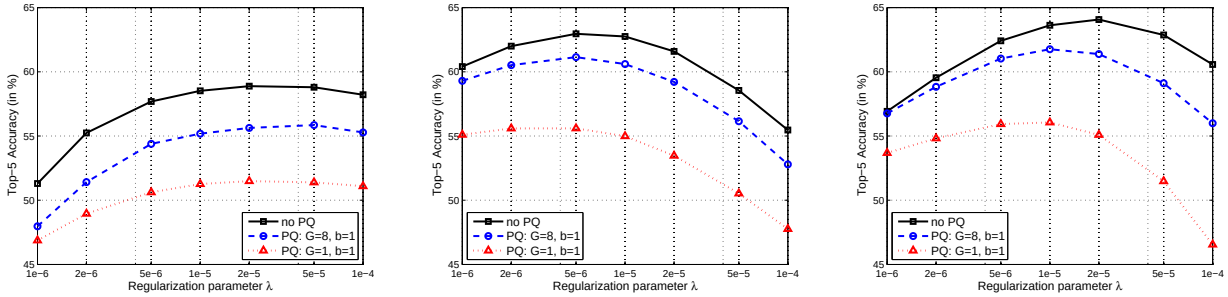

Fig. 9 Impact of the regularization on uncompressed and compressed features for the quadratic loss (left), the log loss (middle) and the hinge loss (right). Results on ILSVRC 2010 with 4K-dim FVs. We experimented with two compression settings: our default setting ($G = 8$ and $b = 1$) and a degraded setting corresponding to scalar quantization ($G = 1$ and $b = 1$).

NN) classifier. For this evaluation we use the 4K- and 64K-dim FVs. We experiment with two compression settings: a "strong" compression with $b = 1$ bit per dimension and $G = 8$ (compression factor 32, our default setting) and and a "weak" compression with $b = 4$ bits per dimension and $G = 2$ (compression factor 8).

We follow the asymmetric testing strategy, where only the training images are PQ encoded and the test images are uncompressed. This has also been used for nearest neighbor image retrieval using an PQ encoded dataset and uncompressed query images (Jégou et al, 2011). We compare three versions of the k-NN classifier. The first directly uses the PQ-encoded Fisher vectors. Note that since the PQ compression is lossy, the norm of the reconstructed vectors is generally not preserved. Since we use ℓ_2 normalized signatures, however, we can correct the norms after decompression. In the second version we re-normalize the decompressed vectors to have unit ℓ_2 norm on both parts corresponding to the SIFT and the LCS color descriptors. Finally, we consider a third variant that simply re-normalizes the complete vector, without taking into account that it consists of two subvectors that should have unit norm each.

Because of the cost of running k-NN classification, we used only a subset of 5K test images to evaluate accuracy. In preliminary experiments on the 4K features, the difference in performance between running the test on the full 150K images and the subset of 5K was small: we observed differences in the order of 0.5%. Hence, we believe this subset of 5K images to be representative of the full set. Note that we could not run uncompressed experiments in a reasonable amount of time with the 64K-dim features, even on this reduced test set. For the SVM, although running experiments on the full test set is possible, we report here results on the same subset of 5K test images for a fair comparison.

For all k-NN experiments we select the best parameter k on the test set for simplicity. Typically for the 4K-dim features and Euclidean distance around 100 neighbors are used, while for the 64K-dim features around 200 neighbors are used. When using the re-normalization the optimal number of neighbors is reduced by a factor two, for both 4K and 64K dimensional features. In either case, the performance is stable over a reasonably large range of values for k.

In Table 7 we show the results of our comparison, using the first and second variants. We make the following ob-

Table 7 Comparison of top-5 accuracy of SVM and k-NN classification without compression ("exact"), with "weak" PQ compression ($b = 4$) and "strong" compression ($b = 1$).

	4K-dim FV			64K-dim FV	
	exact	weak	strong	weak	strong
k-NN, direct	44.3	44.2	42.3	42.7	37.8
k-NN, re-normalization		44.2	42.7	47.0	45.3
SVM	64.1	63.3	61.7	73.3	71.6

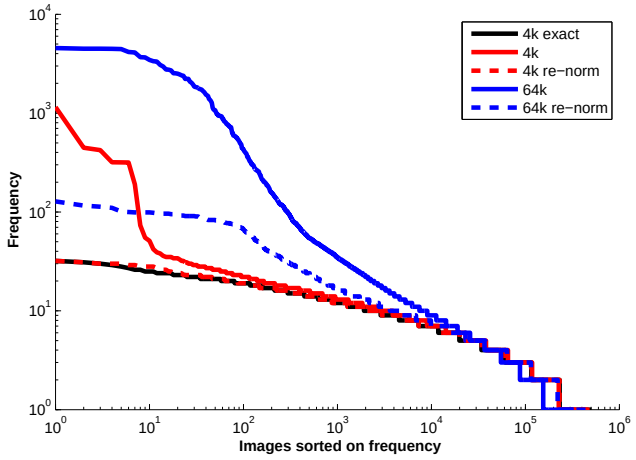

Fig. 10 Impact of the re-normalization on the frequencies of how often an image is selected as neighbor. We analyze the 200 nearest neighbors from the 5K queries, for the 4K and 64K dimensional FV with strong compression and the exact non-compressed 4K FVs.

servations. First the performance of the k-NN classifier is always significantly inferior to that of the SVM classifier. Second, when no re-normalization is used, we observe a decrease in accuracy with the k-NN classifier when going from 4K to 64K dimensional FVs. When using re-normalization, however, the performance does improve from 4K to 64K features. The normalization per FV (SIFT and LCS) is also important; when only a single re-normalization is applied to the complete vector an accuracy of 45.9 is obtained for the 64K-dim feature in the weak compression setting, compared to 47.0 when both FVs are used separately. Third, the modest improvement of around 3 absolute points when going from 4K to 64K-dim features for the k-NN classifier might be explained because the k-NN classifier is not appropriate for high-dimensional features since all points are approximately at the same distance. In such a case, it can be beneficial to employ metric learning techniques, see *e.g.* Mensink et al (2012). However, this is outside the scope of the current paper.

Finally, to obtain a better insight on the influence of the re-normalization, we analyze its impact on which images are selected as nearest neighbors. In Figure 10, we show for different settings the distribution of how often images are selected as one of the 200 nearest neighbors for the 5K queries. In particular, for each train image we count how often it is referenced as a neighbor, then sort these counts in decreasing order, and plot these curves on log-log axes. We compare the distributions for strongly compressed 4K and 64K FVs, with and without re-normalization. Moreover, we also include the distribution for the exact (non-compressed) 4K features. Since there are 5K test images, each of which has 200 neighbors, and about 1M training images, a uniform distribution over the neighbor selection would roughly select each training image once. We observe that without re-normalization, however, a few images are selected very often. For example, for the 64K features there are images that are referenced as a neighbor for more than 4500 queries. When using the re-normalization the neighbor frequencies are more well behaved, resulting in better classification performance, see Table 7. In this case, for the 64K features the most frequent neighbor is referenced less that 150 times: a reduction by a factor 30 as compared to the maximum frequency without re-normalization. For the 4K features the correction of the neighbor frequencies is even more striking: with re-normalization the neighbor frequencies essentially match those of the exact (non-compressed) features.

Comparison with the state-of-the-art. We now compare our results with the state-of-the-art. Using 64K-dim FVs and the "weak" compression setting described earlier, we achieve 73.1% top-5 accuracy. Using our default compression setting, we achieve 72.0%. This is to be compared with the winning NEC-UIUC-Rutgers system which obtained 71.8% accuracy during the challenge (Berg et al, 2010), see also (Lin et al, 2011). Their system combined 6 sub-systems with different patch descriptors, patch encoding schemes and spatial pyramids. (Sánchez and Perronnin, 2011) reported slightly better results – 74.5% top-5 accuracy – using 1M-dim FVs (with more Gaussians and spatial pyramids) but such high-dimensional features are significantly more costly than the 64K-dim features we used in the present paper. In a very recent work, Krizhevsky et al (2012) reported significantly better results using a deep learning network. They achieved an 83.0% top-5 accuracy using a network with 8 layers. We note that to increase the size of the training set – which was found to be necessary to train their large network – they used different data augmentation strategies (random cropping of sub-images and random perturbations of the illumination) which we did not use.

We can also compare our results to those of Bergamo and Torresani (2012) whose purpose was not to obtain the best possible results at any cost but the best possible results for a given storage budget. Using a 15,458-dim binary representation based on meta-class features, Bergamo and Torresani (2012) manage to compress the training set to 2.16 GBs and report a top-1 classification accuracy of 36.0%. Using 4K-dim FVs, our top-1 accuracy is 39.7% while our storage requirements are four times smaller, see Table 6. We underline that the FV representation and the metaclass features of

Bergamo and Torresani (2012) are not exclusive but complementary. Indeed, the metaclass features could be computed over FV representations, thus leading to potential improvements.

5.2 ImageNet10K

ImageNet10K Deng et al (2010) is a subset of ImageNet which contains approximately 9M images corresponding to roughly 10K classes.

For these experiments, we used the exact same setting as for our ILSVRC 2010 experiments: we do not use spatial pyramids and the SIFT- and LCS-FVs are concatenated to obtain either a 4K-dim of a 64K-dim FV. For the compression, we use the default setting ($b = 1$ bit per dimension and $G = 8$). To train the one-vs-rest linear SVMs, we also follow Perronnin et al (2012) and subsample the negatives. At test time, we also compress FVs because of the large of number of test images to store on disk (4.5M).

In our experiments, we follow the protocol of Sánchez and Perronnin (2011) and use half of the images for training, 50K for validation and the rest for testing. We compute the top-1 classification accuracy for each class and report the average per-class acuracy as is standard on this dataset (Deng et al, 2010, Sánchez and Perronnin, 2011). Using the 4K-dim FVs, we achieve a top-1 accuracy of 14.0% and using the 64K-dim FVs 21.9%.

Comparison with the state-of-the-art. Deng et al (2010) achive 6.4% accuracy using a BoV representation and the fast Intesection kernel SVM (IKSVM) technique of Maji and Berg (2009). Our compressed FV results are more than 3 times higher. Sánchez and Perronnin (2011) reported a 16.7% accuracy using FVs but without color information. Using similar features Perronnin et al (2012) improved these results to 19.1% by carefully cross-validating the balance between the positive and negative samples, a good practice we also used in the current work. (Mensink et al, 2012) obtained 13.9% using the same features and PQ compression as we used in this paper, but with a nearest mean classifier which only requires a fraction of the training time.

Le et al (2012) and Krizhevsky et al (2012) also report results on the same subset of 10K classes using deep architectures. Both networks have 9 layers but they are quite different. In (Le et al, 2012), the features are learned using a deep autoencoder which is constructed by replicating three times the same three layers – made of local filtering, local pooling and contrast normalization. Classification is then performed using linear classifiers (trained with a logistic loss). In (Krizhevsky et al, 2012) the network consists of 6 convolutional layers plus three fully connected layers. The output of the last fully connected layer is fed to a softmax which produces a distribution over the class labels. Le et al (2012) reports a top-1 per-image accuracy of 19.2% and Krizhevsky et al (2012) of 32.6%.[9]

6 Conclusion

In this work, we proposed the Fisher Vector representation as an alternative to the popular Bag-of-Visual (BoV) words encoding technique commonly adopted in the image classification and retrieval literature. Within the Fisher Vector framework, images are characterized by first extracting a set of low-level patch descriptors and then computing their deviations from a "universal" generative model, *i.e.* a probabilistic *visual vocabulary* learned offline from a large set of samples. This characterization is given as a gradient vector w.r.t. the parameters of the model, which we choose to be a Gaussian Mixture with diagonal covariances.

Compared to the BoV, the Fisher Vector offers a more complete representation of the sample set, as it encodes not only the (probabilistic) count of occurrences but also higher order statistics related to its distribution w.r.t. the words in the vocabulary. The better use of the information provided by the model translates also into a more efficient representation, since much smaller vocabularies are required in order to achieve a given performance. We showed experimentally on three challenging small and medium-scale datasets that this additional information brings large improvements in terms of classification accuracy and, importantly, that state-of-the-art performances can be achieved with efficient linear classifiers. This makes the Fisher Vector well suited for large-scale classification.

However, being very high-dimensional and dense, the Fisher Vector becomes impractical for large-scale applications due to storage limitations. We addressed this problem by using Product Quantization, which enables balancing accuracy, CPU cost, and memory usage. We provided a theoretical analysis of the influence of Product Quantization on the classifier learning. For the linear case, we showed that compression using quantization has an "unregularization" effect that can be compensated by properly cross-validating the regularization parameters when learning the classifiers.

Finally, we reported results on two large-scale datasets – including up to 9M images and 10K classes– and performed a detailed comparison with k-NN classification. We showed that Fisher Vectors can be compressed by a factor of 32 with a very little impact on classification accuracy.

[9] While it is standard practice to report per-class accuracy on this dataset (see Deng et al (2010), Sánchez and Perronnin (2011)), Krizhevsky et al (2012), Le et al (2012) report a per-image accuracy. This results in a more optimistic number since those classes which are over-represented in the test data also have more training samples and therefore have (on average) a higher accuracy than those classes which are under-represented. This was clarified through a personal correspondence with the first authors of Krizhevsky et al (2012), Le et al (2012).

A An approximation of the Fisher information matrix

In this appendix we show that, under the assumption that the posterior distribution $\gamma_x(k) = w_k u_k(x)/u_\lambda(x)$ is sharply peaked, the normalization with the Fisher Information Matrix (FIM) takes a diagonal form. Throughout this appendix we assume the data x to be one dimensional. The extension to the multidimensional data case is immediate for the mixtures of Gaussians with diagonal covariance matrices that we are interested in.

Under some mild regularity conditions on $u_\lambda(x)$, the entries of the FIM can be expressed as:

$$[F_\lambda]_{i,j} = \mathbb{E}\left[-\frac{\partial^2 \log u_\lambda(x)}{\partial\lambda_i \partial\lambda_j}\right]. \tag{57}$$

First, let us consider the partial derivatives of the posteriors w.r.t. the mean and variance parameters. If we use θ_s to denote one such parameter associated with $u_s(x)$, *i.e.* mixture component number s, then:

$$\begin{aligned}
\frac{\partial\gamma_x(k)}{\partial\theta_s} &= \gamma_x(k)\frac{\partial \log\gamma_x(k)}{\partial\theta_s} && (58)\\
&= \gamma_x(k)\frac{\partial}{\partial\theta_s}\left[\log w_k + \log u_k(x) - \log u_\lambda(x)\right] && (59)\\
&= \gamma_x(k)\left[[\![k=s]\!]\frac{\partial\log u_s(x)}{\partial\theta_s} - \frac{\partial\log u_\lambda(x)}{\partial\theta_s}\right] && (60)\\
&= \gamma_x(k)\left[[\![k=s]\!]\frac{\partial\log u_s(x)}{\partial\theta_s} - \gamma_x(s)\frac{\partial\log u_s(x)}{\partial\theta_s}\right] && (61)\\
&= \gamma_x(k)\Big([\![k=s]\!] - \gamma_x(s)\Big)\frac{\partial\log u_s(x)}{\partial\theta_s} \approx 0, && (62)
\end{aligned}$$

where $[\![\cdot]\!]$ is the Iverson bracket notation which equals one if the argument is true, and zero otherwise. It is easy to verify that the assumption that the posterior is sharply peaked implies that the partial derivative is approximately zero, since the assumption implies that (i) $\gamma_x(k)\gamma_x(s) \approx 0$ if $k \neq s$ and (ii) $\gamma_x(k) \approx \gamma_x(k)\gamma_x(s)$ if $k = s$.

From this result and equations (12), (13), and (14), it is then easy to see that second order derivatives are zero if (i) they involve mean or variance parameters corresponding to different mixture components ($k \neq s$), or if (ii) they involve a mixing weight parameter and a mean or variance parameter (possibly from the same component).

To see that the cross terms for mean and variance of the same mixture component are zero, we again rely on the observation that $\partial\gamma_x(k)/\partial\theta_s \approx 0$ to obtain:

$$\frac{\partial^2\log u_\lambda(x)}{\partial\sigma_k\partial\mu_k} \approx \gamma_x(k)(x-\mu_k)\frac{\partial\sigma_k^{-2}}{\partial\sigma_k} = -2\sigma_k^{-3}\gamma_x(k)(x-\mu_k) \tag{63}$$

Then by integration we obtain:

$$\begin{aligned}
[F_\lambda]_{\sigma_k,\mu_k} &= -\int_x u_\lambda(x)\frac{\partial^2\log u_\lambda(x)}{\partial\sigma_k\partial\mu_k}\mathrm{d}x && (64)\\
&\approx 2\sigma_k^{-3}\int_x u_\lambda(x)\gamma_x(k)(x-\mu_k)\mathrm{d}x && (65)\\
&= 2\sigma_k^{-3}w_k\int_x u_k(x)(x-\mu_k)\mathrm{d}x = 0 && (66)
\end{aligned}$$

We now compute the second order derivatives w.r.t. the means:

$$\frac{\partial^2\log u_\lambda(x)}{(\partial\mu_k)^2} \approx \sigma_k^{-2}\gamma_x(k)\frac{\partial(x-\mu_k)}{\partial\mu_k} = -\sigma_k^{-2}\gamma_x(k) \tag{67}$$

Integration then gives:

$$\begin{aligned}
[F_\lambda]_{\mu_k,\mu_k} &= -\int_x u_\lambda(x)\frac{\partial^2\log u_\lambda(x)}{(\partial\mu_k)^2}\mathrm{d}x \approx \sigma_k^{-2}\int_x u_\lambda(x)\gamma_x(k)\mathrm{d}x && (68)\\
&= \sigma_k^{-2}w_k\int_x u_k(x)\mathrm{d}x = \sigma_k^{-2}w_k, && (69)
\end{aligned}$$

and the corresponding entry in L_λ equals $\sigma_k/\sqrt{w_k}$. This leads to the normalized gradients as presented in (17).

Similarly, for the variance parameters we obtain:

$$\frac{\partial^2\log u_\lambda(x)}{(\partial\sigma_k)^2} \approx \sigma_k^{-2}\gamma_x(k)\Big(1-3(x-\mu_k)^2/\sigma_k^2\Big) \tag{70}$$

Integration then gives:

$$\begin{aligned}
[F_\lambda]_{\sigma_k,\sigma_k} &\approx \sigma_k^{-2}\int_x u_\lambda(x)\gamma_x(k)\Big(3(x-\mu_k)^2/\sigma_k^2 - 1\Big)\mathrm{d}x && (71)\\
&= \sigma_k^{-2}w_k\int_x u_k(x)\Big(3(x-\mu_k)^2/\sigma_k^2 - 1\Big)\mathrm{d}x = 2\sigma_k^{-2}w_k, && (72)
\end{aligned}$$

which leads to a corresponding entry in L_λ of $\sigma_k/\sqrt{2w_k}$. This leads to the normalized gradients as presented in (18).

Finally, the computation of the normalization coefficients for the mixing weights is somewhat more involved. To compute the second order derivatives involving mixing weight parameters only, we will make use of the partial derivative of the posterior probabilities $\gamma_x(k)$:

$$\frac{\partial\gamma_x(k)}{\partial\alpha_s} = \gamma_x(k)\Big([\![k=s]\!] - \gamma_x(s)\Big) \approx 0, \tag{73}$$

where the approximation follows from the same observations as used in (62). Using this approximation, the second order derivatives w.r.t. mixing weights are:

$$\frac{\partial^2\log u_\lambda(x)}{\partial\alpha_s\partial\alpha_k} = \frac{\partial\gamma_x(k)}{\partial\alpha_s} - \frac{\partial w_k}{\partial\alpha_s} \approx -\frac{\partial w_k}{\partial\alpha_s} = w_k\big([\![k=s]\!] - w_s\big) \tag{74}$$

Since this result is independent of x, the corresponding block of the FIM is simply obtained by collecting the negative second order gradients in matrix form:

$$[F_\lambda]_{\alpha,\alpha} = ww' - \mathrm{diag}(w), \tag{75}$$

where we used w and α to denote the vector of all mixing weights, and mixing weight parameters respectively.

Since the mixing weights sum to one, it is easy to show that this matrix is non-invertible by verifying that the constant vector is an eigenvector of this matrix with associated eigenvalue zero. In fact, since there are only $K-1$ degrees of freedom in the mixing weights, we can fix $\alpha_K = 0$ without loss of generality, and work with a reduced set of $K-1$ mixing weight parameters. Now, let us make the following definitions: let $\tilde{\alpha} = (\alpha_1, \ldots, \alpha_{K-1})^T$ denote the vector of the first $K-1$ mixing weight parameters, let $G^x_{\tilde{\alpha}}$ denote the gradient vector with respect to these, and $F_{\tilde{\alpha}}$ the corresponding matrix of second order derivatives. Using this definition $F_{\tilde{\alpha}}$ is invertible, and using Woodbury's matrix inversion lemma, we can show that

$$G^x_{\tilde{\alpha}}F^{-1}_{\tilde{\alpha}}G^y_{\tilde{\alpha}} = \sum_{k=1}^{K}(\gamma_x(k)-w_k)(\gamma_y(k)-w_k)/w_k. \tag{76}$$

The last form shows that the inner product, normalized by the inverse of the non-diagonal $K-1$ dimensional square matrix $F_{\tilde{\alpha}}$, can in fact be obtained as a simple inner product between the normalized version of the K dimensional gradient vectors as defined in (12), *i.e.* with entries $\big(\gamma_x(k)-w_k\big)/\sqrt{w_k}$. This leads to the normalized gradients as presented in (16).

Note also that, if we consider the complete data likelihood:

$$p(x,z|\lambda) = u_\lambda(x)p(z|x,\lambda) \tag{77}$$

the Fisher information decomposes as:

$$F_c = F_\lambda + F_r, \tag{78}$$

where F_c, F_λ and F_r denote the FIM of the complete, marginal (observed) and conditional terms. Using the 1-of-K formulation for z, it

can be shown that F_c has a diagonal form with entries given by (69), (72) and (76), respectively. Therefore, F_r can be seen as the amount of "information" lost by not knowing the true mixture component generating each of the x's Titterington et al (1985). By requiring the distribution of the $\gamma_x(k)$ to be "sharply peaked" we are making the approximation $z_k \approx \gamma_x(k)$.

From this derivation we conclude that the assumption of sharply peaked posteriors leads to a diagonal approximation of the FIM, which can therefore be taken into account by a coordinate-wise normalization of the gradient vectors.

B Good Practices for Gaussian Mixture Modeling

We now provide some good practices for Gaussian Mixture Modeling. For a public GMM implementation and for more details on how to train and test GMMs, we refer the reader to the excellent HMM ToolKit (HTK) Young et al (2002)[10].

Computation in the log domain. We first describe how to compute in practice the likelihood (8) and the soft-assignment (15). Since the low-level descriptors are quite high-dimensional (typically $D = 64$ in our experiments), the likelihood values $u_k(x)$ for each Gaussian can be extremely small (and even fall below machine precision if using floating point values) because of the exp of equation (9). Hence, for a stable implementation, it is of utmost importance to *perform all computations in the log domain*. In practice, for descriptor x, one never computes $u_k(x)$ but

$$\log u_k(x) = -\frac{1}{2}\sum_{d=1}^{D}\left[\log(2\pi) + \log(\sigma_{kd}^2) + \frac{(x_d - \mu_{kd})^2)}{\sigma_{kd}^2}\right] \tag{79}$$

where the subscript d denotes the d-th dimension of a vector. To compute the log-likelihood $\log u_\lambda(x) = \log \sum_{k=1}^{K} w_k u_k(x)$, one does so incrementally by writing $\log u_\lambda(x) = \log\left(w_1 u_1(x) + \sum_{k=2}^{K} w_k u_k(x)\right)$ and by using the fact that $\log(a+b) = \log(a) + \log\left(1 + \exp(\log(b) - \log(a))\right)$ to remain in the log domain.

Similarly, to compute the posterior probability (15), one writes $\gamma_k = \exp\left[\log(w_k u_k(x)) - \log(u_\lambda(x))\right]$ to operate in the log domain.

Variance flooring. Because the variance σ_k^2 appears in a log and as a denominator in equation (79), too small values of the variance can lead to instabilities in the Gaussian computations. In our case, this is even more likely to happen since we extract patches densely and we do not discard uniform patches. In our experience, such patches tend to cluster in a Gaussian mixture component with a tiny variance. To avoid this issue, we use variance flooring: we compute the global covariance matrix over all our training set and we enforce the variance of each Gaussian to be no smaller than a constant α times the global variance. Such an operation is referred to as variance flooring. HTK suggest a value $\alpha = 0.01$.

Posterior thresholding. To reduce the cost of training GMMs as well as the cost of computing FVs, we assume that all the posteriors $\gamma(k)$ which are below a given threshold θ are equal to exactly zero. In practice, we use a quite conservative threshold $\theta = 10^{-4}$ and for a GMM with 256 Gaussians, 5 to 10 Gaussians maximum exceed this threshold. After discarding some of the $\gamma(k)$ values, we renormalize the $\gamma's$ to ensure that we still have $\sum_{k=1}^{K} \gamma(k) = 1$.

Note that this operation does not only reduce the computational cost, it also sparsifies the FV (see section 4.2). Without such a posterior thresholding, the FV (or the soft-BOV) would be completely dense.

Incremental training. It is well known that the Maximum Likelihood (ML) estimation of a GMM is a non-convex optimization problem for more than one Gaussian. Hence, different initializations might lead to different solutions. While in our experience, we have never observed a drastic influence of the initialization on the end result, we strongly advise the use of an iterative process as suggested for instance in (Young et al, 2002). This iterative procedure consists in starting with a single Gaussian (for which a closed-form formula exists), splitting all Gaussians by slightly perturbing the mean and then re-estimating the GMM parameters with EM. This iterative splitting-training strategy enables cross-validating and monitoring the influence of the number of Gaussians K in a consistent way.

[10] Available at: http://htk.eng.cam.ac.uk/.

References

Amari S, Nagaoka H (2000) Methods of Information Geometry, Translations of Mathematical monographs, vol 191. Oxford University Press

Berg A, Deng J, Fei-Fei L (2010) ILSVRC 2010. http://www.image-net.org/challenges/LSVRC/2010/index

Bergamo A, Torresani L (2012) Meta-class features for large-scale object categorization on a budget. In: CVPR

Bishop C (1995) Training with noise is equivalent to tikhonov regularization. In: Neural computation, vol 7

Bo L, Sminchisescu C (2009) Efficient match kernels between sets of features for visual recognition. In: NIPS

Bo L, Ren X, Fox D (2012) Multipath sparse coding using hierarchical matching pursuit. In: NIPS workshop on deep learning

Boiman O, Shechtman E, Irani M (2008) In defense of nearest-neighbor based image classification. In: CVPR

Bottou L (2011) Stochastic gradient descent. http://leon.bottou.org/projects/sgd

Bottou L, Bousquet O (2007) The tradeoffs of large scale learning. In: NIPS

Boureau YL, Bach F, LeCun Y, Ponce J (2010) Learning mid-level features for recognition. In: CVPR

Boureau YL, LeRoux N, Bach F, Ponce J, LeCun Y (2011) Ask the locals: multi-way local pooling for image recognition. In: ICCV

Burrascano P (1991) A norm selection criterion for the generalized delta rule. IEEE Trans Neural Netw 2(1):125–30

Chatfield K, Lempitsky V, Vedaldi A, Zisserman A (2011) The devil is in the details: an evaluation of recent feature encoding methods. In: BMVC

Cinbis G, Verbeek J, Schmid C (2012) Image categorization using Fisher kernels of non-iid image models. In: CVPR

Clinchant S, Csurka G, Perronnin F, Renders JM (2007) XRCEs participation to imageval. In: ImageEval Workshop at CVIR

Csurka G, Dance C, Fan L, Willamowski J, Bray C (2004) Visual categorization with bags of keypoints. In: ECCV SLCV Workshop

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: CVPR

Deng J, Berg A, Li K, Fei-Fei L (2010) What does classifying more than 10,000 image categories tell us? In: ECCV

Everingham M, Gool LV, Williams C, Winn J, Zisserman A (2007) The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results

Everingham M, Gool LV, Williams C, Winn J, Zisserman A (2008) The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results

Everingham M, van Gool L, Williams C, Winn J, Zisserman A (2010) The pascal visual object classes (VOC) challenge. IJCV 88(2):303–338

Farquhar J, Szedmak S, Meng H, Shawe-Taylor J (2005) Improving "bag-of-keypoints" image categorisation. Tech. rep., University of Southampton

Feng J, Ni B, Tian Q, Yan S (2011) Geometric ℓ_p-norm feature pooling for image classification. In: CVPR

Gehler P, Nowozin S (2009) On feature combination for multiclass object classification. In: ICCV

Gray R, Neuhoff D (1998) Quantization. IEEE Trans on Information Theory 44(6)
Griffin G, Holub A, Perona P (2007) Caltech-256 object category dataset. Tech. Rep. 7694, California Institute of Technology, URL http://authors.library.caltech.edu/7694
Guillaumin M, Verbeek J, Schmid C (2010) Multimodal semi-supervised learning for image classification. In: CVPR
Harzallah H, Jurie F, Schmid C (2009) Combining efficient object localization and image classification. In: ICCV
Haussler D (1999) Convolution kernels on discrete structures. Tech. rep., UCSC
Jaakkola T, Haussler D (1998) Exploiting generative models in discriminative classifiers. In: NIPS
Jégou H, Douze M, Schmid C (2009) On the burstiness of visual elements. In: CVPR
Jégou H, Douze M, Schmid C, Pérez P (2010) Aggregating local descriptors into a compact image representation. In: CVPR
Jégou H, Douze M, Schmid C (2011) Product quantization for nearest neighbor search. IEEE PAMI
Jégou H, Perronnin F, Douze M, Sánchez J, Pérez P, Schmid C (2012) Aggregating local image descriptors into compact codes. IEEE Trans Pattern Anal Machine Intell 34(9)
Krapac J, Verbeek J, Jurie F (2011) Modeling spatial layout with fisher vectors for image categorization. In: ICCV
Krizhevsky A, Sutskever I, Hinton G (2012) Image classification with deep convolutional neural networks. In: NIPS
Kulkarni N, Li B (2011) Discriminative affine sparse codes for image classification. In: CVPR
Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR
Le Q, Ranzato M, Monga R, Devin M, Chen K, Corrado G, Dean J, Ng A (2012) Building high-level features using large scale unsupervised learning. In: ICML
Lin Y, Lv F, Zhu S, Yu K, Yang M, Cour T (2011) Large-scale image classification: fast feature extraction and svm training. In: CVPR
Liu Y, Perronnin F (2008) A similarity measure between unordered vector sets with application to image categorization. In: CVPR
Lowe D (2004) Distinctive image features from scale-invariant keypoints. IJCV 60(2)
Lyu S (2005) Mercer kernels for object recognition with local features. In: CVPR
Maji S, Berg A (2009) Max-margin additive classifiers for detection. In: ICCV
Maji S, Berg A, Malik J (2008) Classification using intersection kernel support vector machines is efficient. In: CVPR
Mensink T, Verbeek J, Csurka G, Perronnin F (2012) Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In: ECCV
Perronnin F, Dance C (2007) Fisher kernels on visual vocabularies for image categorization. In: CVPR
Perronnin F, Dance C, Csurka G, Bressan M (2006) Adapted vocabularies for generic visual categorization. In: ECCV
Perronnin F, Liu Y, Sánchez J, Poirier H (2010a) Large-scale image retrieval with compressed Fisher vectors. In: CVPR
Perronnin F, Sánchez J, Liu Y (2010b) Large-scale image categorization with explicit data embedding. In: CVPR
Perronnin F, Sánchez J, Mensink T (2010c) Improving the Fisher kernel for large-scale image classification. In: ECCV
Perronnin F, Akata Z, Harchaoui Z, Schmid C (2012) Towards good practice in large-scale learning for image classification. In: CVPR
Sabin M, Gray R (1984) Product code vector quantizers for waveform and voice coding. IEEE Transactions on Acoustics, Speech and Signal Processing 32(3)
Sánchez J, Perronnin F (2011) High-dimensional signature compression for large-scale image classification. In: CVPR
Sánchez J, Perronnin F, de Campos T (2012) Modeling the spatial layout of images beyond spatial pyramids. Pattern Recognition Letters 33(16):2216 – 2223
van de Sande K, Gevers T, Snoek C (2010) Evaluating color descriptors for object and scene recognition. IEEE PAMI 32(9)
Shalev-Shwartz S, Singer Y, Srebro N (2007) Pegasos: Primal estimate sub-gradient solver for SVM. In: ICML
Sivic J, Zisserman A (2003) Video Google: A text retrieval approach to object matching in videos. In: ICCV
Smith N, Gales M (2001) Speech recognition using SVMs. In: NIPS
Song D, Gupta AK (1997) Lp-norm uniform distribution. In: Proc. American Mathematical Society, vol 125, pp 595–601
Spruill M (2007) Asymptotic distribution of coordinates on high dimensional spheres. Electronic Communications in Probability 12
Sreekanth V, Vedaldi A, Jawahar C, Zisserman A (2010) Generalized rbf feature maps for efficient detection. In: BMVC
Titterington DM, Smith AFM, Makov UE (1985) Statistical Analysis of Finite Mixture Distributions. John Wiley, New York
Torralba A, Efros AA (2011) Unbiased look at dataset bias. In: CVPR
Uijlings J, Smeulders A, Scha R (2009) What is the spatial extent of an object? In: CVPR
VanGemert J, Veenman C, Smeulders A, Geusebroek J (2010) Visual word ambiguity. IEEE TPAMI
Vedaldi A, Zisserman A (2010) Efficient additive kernels via explicit feature maps. In: CVPR
Vedaldi A, Zisserman A (2012) Sparse kernel approximations for efficient classification and detection. In: CVPR
Wallraven C, Caputo B, Graf A (2003) Recognition with local features: the kernel recipe. In: ICCV
Wang G, Hoiem D, Forsyth D (2009) Learning image similarity from flickr groups using stochastic intersection kernel machines. In: ICCV
Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. In: CVPR
Winn J, Criminisi A, Minka T (2005) Object categorization by learned visual dictionary. In: ICCV
Xiao J, Hays J, Ehinger K, Oliva A, Torralba A (2010) SUN database: Large-scale scene recognition from abbey to zoo. In: CVPR
Yan S, Zhou X, Liu M, Hasegawa-Johnson M, Huang T (2008) Regression from patch-kernel. In: CVPR
Yang J, Li Y, Tian Y, Duan L, Gao W (2009a) Group sensitive multiple kernel learning for object categorization. In: ICCV
Yang J, Yu K, Gong Y, Huang T (2009b) Linear spatial pyramid matching using sparse coding for image classification. In: CVPR
Young S, Evermann G, Hain T, Kershaw D, Moore G, Odell J, Ollason D, Povey S, Valtchev V, Woodland P (2002) The HTK book (version 3.2.1). Cambridge University Engineering Department
Zhang J, Marszalek M, Lazebnik S, Schmid C (2007) Local features and kernels for classification of texture and object categories: a comprehensive study. IJCV 73(2)
Zhou Z, Yu K, Zhang T, Huang T (2010) Image classification using super-vector coding of local image descriptors. In: ECCV