High-Dimensional Signature Compression for Large-Scale Image Classification

Jorge Sánchez and Florent Perronnin
Textual and Visual Pattern Analysis (TVPA) group
Xerox Research Centre Europe (XRCE)

Abstract

We address image classification on a large-scale, i.e. *when a large number of images and classes are involved. First, we study classification accuracy as a function of the image signature dimensionality and the training set size. We show experimentally that the larger the training set, the higher the impact of the dimensionality on the accuracy. In other words, high-dimensional signatures are important to obtain state-of-the-art results on large datasets. Second, we tackle the problem of data compression on very large signatures (on the order of 10^5 dimensions) using two lossy compression strategies: a dimensionality reduction technique known as the hash kernel and an encoding technique based on product quantizers. We explain how the gain in storage can be traded against a loss in accuracy and / or an increase in CPU cost. We report results on two large databases – ImageNet and a dataset of 1M Flickr images – showing that we can reduce the storage of our signatures by a factor 64 to 128 with little loss in accuracy. Integrating the decompression in the classifier learning yields an efficient and scalable training algorithm.* **On ILSVRC2010 we report a 74.3% accuracy at top-5, which corresponds to a 2.5% absolute improvement with respect to the state-of-the-art. On a subset of 10K classes of ImageNet we report a top-1 accuracy of 16.7%, a relative improvement of 160% with respect to the state-of-the-art.**

1. Introduction

Scaling-up image classification systems is a problem which is receiving an increasing attention as larger labeled image datasets are becoming available. For instance, ImageNet (www.image-net.org) consists of more than 12M images of 17K concepts [7] and Flickr contains thousands of groups (www.flickr.com/groups) – some of which with hundreds of thousands of pictures – which can be readily used to learn object classifiers [31, 22].

The focus in the image classification community was initially on developing systems which would yield the best possible accuracy fairly independently of their cost. The winners of the PASCAL VOC 2007 [8] and 2008 [9] competitions used a similar paradigm: many types of low-level local features are extracted (referred to as "channels"), one bag-of-visual-words (BOV) histogram is computed for each channel and non-linear kernel classifiers such as SVMs are used to perform classification [38, 29]. The use of many channels and costly non-linear SVMs was made possible by the modest size of the available databases.

In recent years only has the computational cost become a central issue in image classification / object detection. In [19], Maji *et al*. showed that the runtime cost of an intersection kernel (IK) SVM could be made independent of the number of support vectors. Maji and Berg [18] and Wang *et al*. [31] then proposed efficient algorithms to learn IKSVMs. Vedaldi and Zisserman [30] and Perronnin *et al*. [21] subsequently generalized this principle to any additive classifier. Another line of research consists in computing image representations which are directly amenable to costless linear classification. Yang *et al*. [36], Wang *et al*. [32] and Boureau *et al*. [4] showed that replacing the average pooling stage in the BOV computation by a max-pooling yielded excellent results. To go beyond the BOV, *i.e*. beyond counting, it has been proposed to include higher order statistics in the image signature. This includes modeling an image by a probability distribution [17, 35] or using the Fisher kernel framework [20]. Especially, it was shown that the Fisher Vector (FV) could yield high accuracy with linear classifiers [22].

If one wants to stick to efficient linear classifiers, the image representations should be high-dimensional to ensure linear separability of the classes. Therefore, we argue that *the storage/memory cost is becoming a central issue in large-scale image classification*. As an example, in this paper we consider almost dense image representations – based on the improved FV of [22] – with up to $524K$ dimensions. Using a 4 byte floating point representation, a single signature requires 2MB of storage. Storing the ILSVRC2010 dataset [2] would take approximately 2.8TBs and storing the full ImageNet dataset around 23TBs. Obviously, these numbers have to be multiplied by the number of channels, *i.e*. feature types. As another example, the

Authorized licensed use limited to: New York University AbuDhabi Campus. Downloaded on February 25,2024 at 00:04:40 UTC from IEEE Xplore. Restrictions apply.

PASCAL VOC 2009 winners mention in [10] (slide 27) that their GMM-based [39] and sparse-coding-based [37] representations have respectively 655K and 2M dimensions. Handling TBs of data makes experimentation very difficult if not impractical. Indeed, much more time can be spent writing / reading data than performing calculations.

In this paper we address the compression of high-dimensional image signatures (on the order of 10^5 dimensions) for large-scale image classification. The gain in storage has to be traded against a loss in accuracy and / or an increase in CPU cost. Most previous works on image signature compression (c.f. section 2) have considered much smaller signatures and have focused on retrieval. Comparatively, compression is relatively unexplored in the classification domain (beyond nearest neighbor approaches).

Our contributions are the following ones:

- We study image classification accuracy as a function of the signature dimensionality and the training set size. This study is conducted on two large datasets containing roughly 1M training images each. A key conclusion is that, with linear classifiers, *high-dimensional signatures are necessary to obtain state-of-the-art results on large datasets*. This justifies the use of high-dimensional signatures in large-scale classification and, therefore, the need for data compression.

- We first tackle data compression with dimensionality reduction using the Hash Kernel (HK) which has recently been shown to yield state-of-the-art results on very high-dimensional data [26, 33]. However, we show experimentally that even when reducing the dimensionality by a small (< 10) factor, the HK leads to a significant decrease in accuracy.

- Second, we use product quantization [11] for data compression. It is shown to be particularly well suited to such high-dimensional signatures as it enables to balance classification accuracy, computational cost and storage cost. We will show that, in combination with a FV-specific sparsity encoding, we can reduce the storage cost by a factor 64-128 with little loss in accuracy and at reasonable CPU cost. Also, by integrating the decompression in the classifier learning we obtain a very efficient and scalable training algorithm.

The remainder of this article is organized as follows. In the next section, we discuss related work. In section 3, we briefly review the high-dimensional image signature we will use throughout this paper. In section 4 we study the image classification accuracy as a function of the training set size. In section 5, we explore the compression of very high-dimensional signatures for large-scale image classification. In section 6, we report compression experiments.

2. Related Work

We review related work on large-scale image classification, dimensionality reduction and data compression.

Large-Scale Classification. Li *et al*. [16] study landmark classification on a collection of 500 landmarks and 2 million images. On a small subset of 10 classes, they could improve BOV classification by increasing the visual vocabulary up to 80K visual words. However, for practical reasons, a much smaller number of visual words was used for larger-scale experiments. Perronnin *et al*. [22] use up to 350K Flickr group images and a signature similar to the one we employ in this work. In our experience, it is difficult to handle significantly more signatures without compression. Deng *et al*. [6] report experiments with up to 10K classes and 10M images. While data compression is mentioned in the appendix, standard strategies based on sparse encoding and scalar quantization were employed. A major difference between [6] and our work is that we advocate for significantly higher-dimensional image signatures (our signatures are an order of magnitude larger and much denser) and more elaborate compression schemes. High-dimensional signatures enable us to stick to costless linear classifiers.

Dimensionality Reduction. Among the most popular techniques are those based on dense projections such as Principal Component Analysis (PCA), Partial Least Squares (PLS) or Gaussian Random Projections (RPs). Since the projection cost is proportional to the number of input and output dimensions, this is computationally tractable for high-dimensional input data only if the number of output dimensions is tiny. For instance Schwartz *et al*. [24] managed to reduce the dimensionality of their features from 170K to as little as 20. A small output dimensionality is sufficient for a small number of classes (*e.g*. 2 in [24]) but should deteriorate performance for a large number of classes. Hash kernels [26, 33] , which are closely related to the “database-friendly” sparse RPs of Achlioptas [1], were recently proposed to address this issue. They have shown excellent performance in the compression of text features [26, 33] and faces [25]. Other works have proposed to perform a semantic dimensionality reduction [31, 28]. An image is scored against a set of concepts and the vector of concept scores is then used as input to the classifiers. However, in our opinion, such an approach does not solve our problem. Indeed, we now need to learn classifiers for the concepts which can be more costly than learning the original classifiers. For instance, in [28], the problem of learning 256 classes is turned into one of learning 2,659 concepts. More generally, it is unclear how the number of concepts should scale with the number of classes.

Data Compression. Image signature compression has received a lot of attention since Torralba *et al*. showed that small codes enable efficient image retrieval and k-NN classification on a large scale [27]. However, this work as well

Authorized licensed use limited to: New York University AbuDhabi Campus. Downloaded on February 25,2024 at 00:04:40 UTC from IEEE Xplore. Restrictions apply.

as many subsequent ones [34, 12, 23, 5, 14] focused on small image signatures (typically GIST or BOV) with a few hundreds or a few thousands of dimensions. This is two to three orders of magnitude smaller than the signatures we employ in this work. Such small signatures do not contain enough information for highly accurate image annotation (c.f. section 4). Moreover the techniques which are proposed in these papers are typically too costly to be applied to our high-dimensional signatures. For instance, [23] requires dense projections and [34, 5, 14] require a global PCA. Recently Jégou *et al.* [13] proposed product quantizers (PQ) for NN-search of SIFT vectors. Although PQ had only been applied to fairly small dimensional vectors [13, 14], it is scalable and enables the balancing of classification accuracy, computational cost and storage cost.

3. The Fisher Vector

We only provide a brief introduction to the FV. More details can be found in [20, 22]. The FV extends the BOV by going beyond counting (0-order statistics) and by encoding statistics (up to the second order) about the distribution of descriptors assigned to each visual word. A significant advantage with respect to the BOV is that high-dimensional discriminative signatures can be obtained even with small vocabularies, and therefore at a low CPU cost.

The FV $\mathcal{G}_\lambda^X$ characterizes a sample X by its deviation from a distribution u_λ (with parameters λ):

$$\mathcal{G}_\lambda^X = L_\lambda G_\lambda^X. \tag{1}$$

G_λ^X is the gradient of the log-likelihood with respect to λ:

$$G_\lambda^X = \frac{1}{T}\nabla_\lambda \log u_\lambda(X). \tag{2}$$

L_λ is the Cholesky decomposition of the inverse of the Fisher information matrix F_λ of u_λ, *i.e.* $F_\lambda^{-1} = L'_\lambda L_\lambda$. In our case, $X = \{x_t, t = 1\dots T\}$ is the set of T local descriptors extracted from an image and $u_\lambda = \sum_{i=1}^N w_i u_i$ is a GMM (with diagonal covariance matrices) which models the generation process of local descriptors. $\lambda = \{w_i, \mu_i, \sigma_i, i = 1\dots N\}$ where w_i, μ_i and σ_i are respectively the mixture weight, mean vector and standard deviation vector of Gaussian u_i. Let $\gamma_t(i)$ be the soft assignment of descriptor x_t to Gaussian i. Assuming that the descriptors x_t are iid we obtain the following formulas for the gradients with respect to μ_i and σ_i[1]:

$$\mathcal{G}_{\mu,i}^X = \frac{1}{T\sqrt{w_i}}\sum_{t=1}^T \gamma_t(i)\left(\frac{x_t-\mu_i}{\sigma_i}\right), \tag{3}$$

$$\mathcal{G}_{\sigma,i}^X = \frac{1}{T\sqrt{2w_i}}\sum_{t=1}^T \gamma_t(i)\left[\frac{(x_t-\mu_i)^2}{\sigma_i^2}-1\right]. \tag{4}$$

[1]Following [20, 22] we discard the partial derivatives with respect to the mixture weights as they carry little discriminative information.

The FV $\mathcal{G}_\lambda^X$ is the concatenation of the $\mathcal{G}_{\mu,i}^X$ and $\mathcal{G}_{\sigma,i}^X$ vectors. As shown in [22], square-rooting and L2-normalizing the FV can greatly enhance the classification accuracy. Also, following the pyramid matching approach of Lazebnik *et al.* [15], one can split an image into several regions, compute one FV per region and concatenate the per-region FVs.

Let D be the dimensionality of the local descriptors, N be the number of Gaussians and R be the number of image regions. The resulting vector is $E = 2DNR$ dimensional.

4. Large Signatures Make a Large Difference

The goal of this first set of experiments is to study the joint influence of the training set size and signature complexity (*i.e.* dimensionality) on the classification accuracy when no compression is applied to image signatures. The training set size depends on two factors: the number of classes and the (average) number of images per class. This study complements a recent work by [16] (a difference is that [16] considers only the number of classes, not the number of images per class).

Datasets. We use two databases with similar training set sizes (roughly 1M images) but different class statistics. The first dataset, ILSVRC2010 [2], contains a large number of classes (1K) but a relatively small number of images per class. The training set consists of 1.26M images (*i.e.* on average 1.26K images per class). We use the validation set (50K images) for validation purposes and the accuracy is evaluated on the test set (150K images). Accuracy is reported as the percentage of images whose correct tag is in the top-5 predicted tags[2]. The second dataset – dubbed FLICKR1M – was inspired by [31, 22]. We downloaded close to 1M images from 18 Flickr groups corresponding to 18 of the 20 PASCAL VOC classes (the only classes for which we did not find large enough groups were "sofa" and "tv"). Hence, this dataset contains a small number of classes but a large number of images per class: 50K on average with a minimum of 2.8K for "diningtable" and a maximum of 136K for "car". We use as validation set the trainval set of VOC 2007 and accuracy is evaluated on the VOC 2007 test set (we remove from the training set of Flickr group images all overlapping test images). Accuracy is reported as the Average Precision (AP), averaged over the 18 classes.

Image signatures. Images are resized to 100K pixels (if larger). SIFT descriptors are extracted densely at multiple scales. To keep the computational cost reasonable, we extract on the order of 1K patches per image (except where the contrary is specified). The feature dimensionality is reduced to $D = 64$ with PCA. To simplify the analysis we do not perform any partitioning of the image and therefore extract a single FV for the full image (*i.e.* $R = 1$). Hence, in the following experiments, the single factor which affects

[2]The flat cost used during the challenge is 1 minus this quantity.

Authorized licensed use limited to: New York University AbuDhabi Campus. Downloaded on February 25,2024 at 00:04:40 UTC from IEEE Xplore. Restrictions apply.

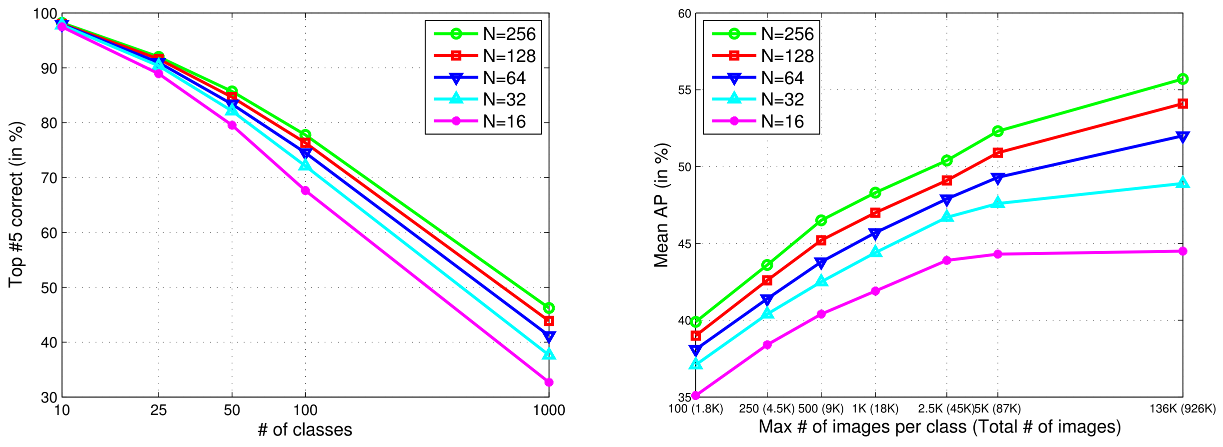


Figure 1. Accuracy as a function of the number of Gaussians N and the training set size. Left: results on ILSVRC2010 with a variable number of classes. Right: results on FLICKR1M with a variable number of images per class.

the signature dimensionality is the number N of Gaussians. We vary N between 16 and 256 which leads to signatures ranging in size from $2,048$ to $32,768$ dimensions.

Classification. We use linear SVMs trained with Stochastic Gradient Descent (SGD) [3] using one sample at a time. For FLICKR1M, the 18 classifiers are trained on all positive and negative samples. For ILSVRC2010, given the high computational cost of learning 1K classifiers, we subsample negatives: at each iteration we use all positive samples and we randomly sample the same number of negatives. We typically use 50 SGD iterations, *i.e.* we see on average 5% of negatives when training a classifier. Increasing significantly the number of iterations only has a limited impact on the classification accuracy (*e.g.* +1-2% for $\times$ 10 more iterations) showing that *it is not necessary to view all negatives to obtain good results*.

Experimental setup. The influence of the number C of classes is studied on ILSVRC2010 and we vary C from 10 to 1K. For a given number C, we split the classes into $1,000/C$ groups of C classes, run experiments for each group and average results. The influence of the number of images per class is studied on FLICKR1M. We vary this number from 100 per class to the full training set. We repeat all experiments (but those carried-out on the full dataset) five times and average results.

Results. Results are shown in Figure 1.

On ILSVRC2010, when considering 10 classes, the accuracy increases from 97.5% for N=16 to 98.2% for N=256 (+0.7% absolute and less than 1% in relative). When considering the 1K classes, the accuracy increases from 32.7% to 46.2% (13.5% absolute and +41.3% relative).

On FLICKR1M, when considering 100 images per class, the accuracy increases from 35.1% for N=16 to 39.9% for N=256 (+4.8% absolute and +13.7% relative). On the full training set, the accuracy increases from 44.5% to 55.7% (+11.2% absolute and +25.2% relative).

We can draw the following conclusion: the larger the training set, *i.e.* the larger the number of classes and/or the number of images per class, the more high-dimensional signatures make a difference. In other words, *we need high-dimensional signatures to obtain state-of-the-art results on large datasets*. Conversely, while the training set size has a less pronounced influence on small signatures, it has a greater impact on large signatures. For instance, for N=16 the increase in accuracy from 45K training images to the full 926K is modest (+0.6% absolute) while for N=256 the accuracy increases significantly (+5.3% absolute).

5. High-Dimensional Signature Compression

The previous experiments clearly show that high-dimensional signatures are important to obtain state-of-the-art accuracy in large-scale (linear) classification. However such high-dimensional signatures come at a high storage/memory cost (TBs of data) which makes learning difficult if not infeasible. We now address the compression of such high-dimensional signatures. We explore two lossy compression schemes for high-dimensional signatures: dimensionality reduction based on hash kernels and data encoding with product quantizers. We also propose a FV-specific sparsity encoding scheme.

5.1. Hash kernels

The Hash Kernel (HK) [26, 33] is a sparse random projection technique which was proposed for very high-dimensional data. It consists in accumulating all the coordinates of a vector for which a hash function generates the same value. The HK is the dot-product between vec-

Authorized licensed use limited to: New York University AbuDhabi Campus. Downloaded on February 25,2024 at 00:04:40 UTC from IEEE Xplore. Restrictions apply.

tors in this reduced space. A similar idea was proposed by Jégou *et al*. [12] for the dimensionality reduction of BOF histograms.

The HK may come in different flavors. The aggregation process may be unbalanced [26, 33] or balanced [12], *i.e*. a variable or constant number of input dimensions may be collapsed into a given dimension. Also one may consider an unsigned sum [26] or a signed sum [33] of hash features. In the unsigned case, the HK leads to a biased estimate of the dot-product in the original space while in the signed case it leads to an unbiased estimate. This is a very important property as the dot-product is a good measure of similarity between FVs [22]. Therefore, learning linear classifiers on hashed FVs makes sense. We will evaluate in section 6 the 4 possible combinations (balanced/unbalanced and biased/unbiased).

5.2. Product quantization

Product Quantizer (PQ). A Vector Quantizer (VQ) $q : \mathbb{R}^E \rightarrow \mathcal{C}$ maps a vector $v \in \mathbb{R}^E$ to a codeword $c_k \in \mathbb{R}^E$ in the codebook $\mathcal{C} = \{c_k, k = 1 \dots K\}$ [11]. The cardinality K of the set $\mathcal{C}$, 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 codewords. 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. 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.

A solution is to use PQs which were introduced as a principled way to deal with high dimensional input spaces (see e.g. [13] for an excellent introduction to the topic). A PQ $q : \mathbb{R}^E \rightarrow \mathcal{C}$ splits a vector v into a set of M distinct subvectors of size $G = E/M$, *i.e.* $v = [v_1, \dots, v_M]$. M subquantizers $\{q_m, m = 1 \dots M\}$ operate independently on each of the sub-vectors. If $\mathcal{C}_m$ is the codebook associated with q_m, then $\mathcal{C}$ is the Cartesian product $\mathcal{C} = \mathcal{C}_1 \times \dots \mathcal{C}_M$ and $q(v)$ is the concatenation of the $q_m(v_m)$'s. We note $G = 1$ corresponds to scalar quantization and that if $G = E$ we are back to the VQ problem on the full vector. Again, let b be the average number of bits per dimension (assuming that the bits are equally distributed across the codebooks $\mathcal{C}_m$) . The codebook size of $\mathcal{C}$ 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 $O(E2^{bG})$. To keep these costs reasonable we have to cap the value bG. In practice we enforce $bG \leq 8$ which ensures that, in our implementation, the cost of encoding a FV is not higher than the cost of extracting the FV itself. Obviously, different applications might have different constraints.

FV sparsity encoding. We mentioned earlier that the FV is dense: on average, only approximately 50% of the dimensions are zero. 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 (3) and (4) all the gradients are zero. Hence, *we do not need to encode the sparsity on a per-dimension basis but on a per-Gaussian basis*.

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. 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 $2D = 128$ gradient values of this Gaussian using PQ.

Note that adding this per Gaussian bit can be viewed as a first step toward gain/shape coding, *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.

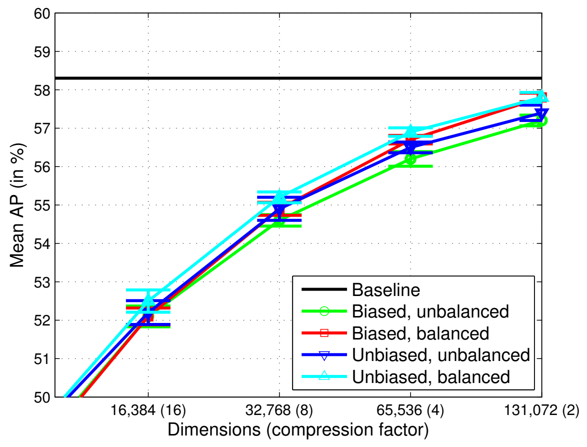
SGD Learning. Since we cannot learn a linear classifier directly in the compressed space (*i.e*. in the space of codebook indices), we have to learn the classifier in the original high dimensional space. We therefore integrated the decompression algorithm in the SGD training code. All compressed signatures are kept in RAM if possible. When a signature is passed to the SGD algorithm, it is decompressed on the fly. 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. The decompression time is very reasonable compared to the training cost as it involves simple look-up table accesses. This makes our *learning scheme both efficient and scalable*.

Since we learn the linear classifier in the original space, test images do not need to be compressed. In our experiments, we will report results with uncompressed test images, except where the contrary is specified.

6. Compression Results

We start experiments with the small PASCAL VOC 2007 dataset [8]. This enables to run an uncompressed baseline even with very large image signatures. We then proceed with ILSVRC2010 [2] and FLICKR1M. We finally report results on a subset of 10K classes of ImageNet [6].

Authorized licensed use limited to: New York University AbuDhabi Campus. Downloaded on February 25,2024 at 00:04:40 UTC from IEEE Xplore. Restrictions apply.



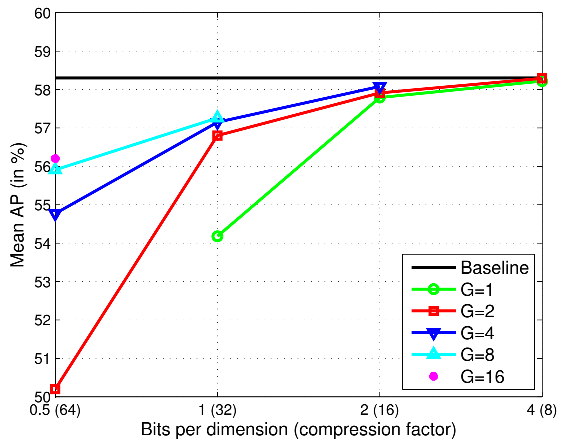


Figure 2. Compression results on VOC 2007. Left: HK results as a function of the number of dimensions. Right: PQ results as a function of the number b of bits per dimension and the group size G (without sparsity encoding). The baseline corresponds to the uncompressed signature (262,144 dimensions). For a given compression factor, PQ performs much better than HK.

6.1. PASCAL VOC 2007

We follow the standard protocol of training the classifiers on the trainval set (5K images) and evaluating the results on the test set (5K images). We report the accuracy as the Average Precision (AP), averaged over the 20 classes. We choose $N = 256$ Gaussians and $R = 8$ regions [3] which leads to 262,144-dimensional signatures. The uncompressed baseline yields 58.3%.

We first report compression with HK. We repeat the experiments 10 times (with 10 different hashing functions) and Figure 2 (left) shows the average and the standard deviation for the four different flavors. Our conclusions are the following ones. First, the unbiased balanced version seems to yield slightly better results than the other variations but the difference is quite small. Second, accuracy drops rapidly when decreasing the number of dimensions. For instance, reducing to 32K dimensions (*i.e.* by a factor 8) yields 55.2%. However, if we did not use spatial pyramids, we would obtain vectors of exactly the same size and achieve the same accuracy (55.3%). These results seem to indicate that our data does not lie in a much lower dimensional subspace of the original 262,144 dimensional space.

We now turn to PQ. We again repeat the experiments 10 times (with 10 different runs of k-means). We report the average on Figure 2 (right). The standard deviation is not shown because it is very small ($< 0.1\%$). A key conclusion is that, *for a given compression factor, PQ consistently outperforms HK*. Another conclusion is that, the smaller the number of bits per dimension b we can afford, the more the group size G makes a difference, *i.e.* the more important it is to capture the correlation between dimensions. For instance, for $b = 1$, the AP is increased from 54.2% for G=1 to 57.3% for G=8 and for $b = 0.5$ from 50.2% for G=1 to 56.2% for G=16. In conjunction with the sparse encoding (which results in an additional 50% saving), $b = 0.5$ and $b = 1$ lead respectively to a 64 and 128 fold reduction in memory. The previous PQ results do not consider any compression of test samples. Compressing test samples typically leads to an additional 1% loss (*e.g.* from 57.3% to 56.5% for $b = 1$ and $G = 8$).

[3]The 8 FVs are the following ones: one for the whole image, three for the top, middle and bottom regions and one for each of the four quadrants.

6.2. ILSVRC2010 and FLICKR1M

We now run experiments on the full ILSVRC2010 and FLICKR 1M datasets (c.f. section 4 for details on the protocol). In a first stage, we do not make use of spatial pyramids ($R = 1$). We set $N = 256$ which yields 2^{15}=32,768-dimensional signatures. Considering smaller signatures enables to run the uncompressed baseline. Table 1 provides HK results. We report the average and the standard deviation over 10 different HK runs. Again, we observe a very rapid decrease of the accuracy with the number of dimensions. We show on Figure 3 (top) PQ results. Again, PQ clearly outperforms HK for a given compression rate.

We also show results on 262,144-dimensional signatures ($N = 256, R = 8$) on Figure 3 (bottom). We were not able to train classifiers on these uncompressed signatures in a reasonable amount of time. The main issue is that, without compression, much more time is spent reading data than performing actual computation. Indeed, on our double quadcore multi-threaded machine with Intel Xeon processors (16 processing units), we could use on average only 1 processing unit when dealing with uncompressed data because of the throughput bottleneck. When compressing the

Authorized licensed use limited to: New York University AbuDhabi Campus. Downloaded on February 25,2024 at 00:04:40 UTC from IEEE Xplore. Restrictions apply.

data, we could use the 16 processing units simultaneously. For a number $b \leq 1$ of bits per dimension we could even fit the full training set in the RAM of our 48GB machine (for $b = 1$ the compressed ILSVRC2010 training set fits in 20GB). In this case, *the SGD learning algorithm trains the 1K classifiers on 1.2TBs of training data (after decompression) in approx. 5h30.*

We underline that using large signatures with compression yields much better results than using smaller signatures without compression. For instance, without spatial pyramids and without compression, the best result we obtain on ILSVRC2010 is 46.2%. Using spatial pyramids (8 times larger signatures) and a fairly aggressive compression rate of 64 (b=1 combined with sparsity encoding), we can reach 56.1% (+9.9% absolute).

Getting state-of-the-art results on ILSVRC2010. We now show that, using the FV-compressed features, we can get state-of-the-art accuray on ILSVRC2010. Compared to the previous experiments, we (a) increased the number of patches per image from 1K to 10K, (b) increased the dimensionality of our FVs to approximately 524K ($N = 1,024$ and $R = 4$ regions[4]) and (c) increased the number of SGD iterations from 50 to 200. With these parameters, we achieve 67.9% accuracy at top-5. Using a second type of low-level features (the 96 dimensional color descriptors of [22] reduced to 64 dimensions with PCA), and averaging the results of the two channels **we can further increase the top-5 accuracy to 74.3%. This is a 2.5% improvement with respect to the state-of-the-art** (the winning NEC-UIUC-Rutgers system achieved 71.8% accuracy during the challenge [2]). We also provide our top-1 accuracy: 54.3%.

Training one channel on a single machine with 16 cpus takes approx. 3.5 days: 2 days for the low-level feature extraction (SIFT or color) + FV computation/compression and 1.5 days for the SGD training. We underline that both steps could be easily parallelized on multiple machines.

6.3. ImageNet10K

We finally replicated the very large-scale experiments of Deng *et al.* [6]: we ran our system on the same 10,184 categories from the Fall 2009 release of ImageNet including both internal and leaf nodes with more than 200 images. This makes an approximate total of 9M images, half of which is used for training and half for testing. Out of the 4.5M training images, we keep a small subset of 50K images for validation purposes. We report the average per-class accuracy as in [6].

We extracted on the order of 10K SIFT descriptors per image (no color information). To keep the computational cost reasonable, we set the number of Gaussians to $N = 256$ and the number of regions to $R = 4$ which yields 131K dimensional signatures. We performed 500 iterations of SGD. **We report a top-1 accuracy of 16.7% to be compared to 6.4% in [6]. We underline that this corresponds to a relative improvement of 160%.**

[4]The 4 FVs correspond to: full image, top, middle and bottom.

References

[1] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66, 2003.

[2] A. Berg, J. Deng, and F.-F. Li. ILSVRC 2010. http://www.image-net.org/challenges/LSVRC/2010/index.

[3] L. Bottou. SGD. http://leon.bottou.org/projects/sgd.

[4] Y.-L. Bourreau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[5] J. Brandt. Transform coding for fast approximate nearest neighbor search in high dimensions. In *CVPR*, 2010.

[6] J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[8] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL VOC 2007 results.

[9] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL VOC 2008 results.

[10] Y. Gong, T. Huang, F. Lv, J. Wang, C. Wu, W. Xu, J. Yang, K. Yu, T. Zhang, and X. Zhou. Image classification using Gaussian mixture and local coordinate coding. PASCAL VOC 2009 workshop, http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2009/workshop/yu.pdf, 2009.

[11] R. Gray and D. Neuhoff. Quantization. *IEEE Trans. on IT*, 44(6), 2006.

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

[13] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on PAMI*, 33(1), 2011.

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

[15] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[16] Y. Li, D. Crandall, and D. Huttenlocher. Landmark classification in large-scale image collections. In *ICCV*, 2009.

[17] Y. Liu and F. Perronnin. A similarity measure between unordered vector sets with application to image categorization. In *CVPR*, 2008.

[18] S. Maji and A. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009.

[19] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008.

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

[21] F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *CVPR*, 2010.

[22] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010.

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

[24] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis. Human detection using partial least squares analysis. In *ICCV*, 2009.

[25] Q. Shi and H. Li. Rapid face recognition using hashing. In *CVPR*, 2010.

[26] Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, A. Strehl, and V. Vishwanathan. Hash kernels. In *AISTATS*, 2009.

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

Authorized licensed use limited to: New York University AbuDhabi Campus. Downloaded on February 25,2024 at 00:04:40 UTC from IEEE Xplore. Restrictions apply.

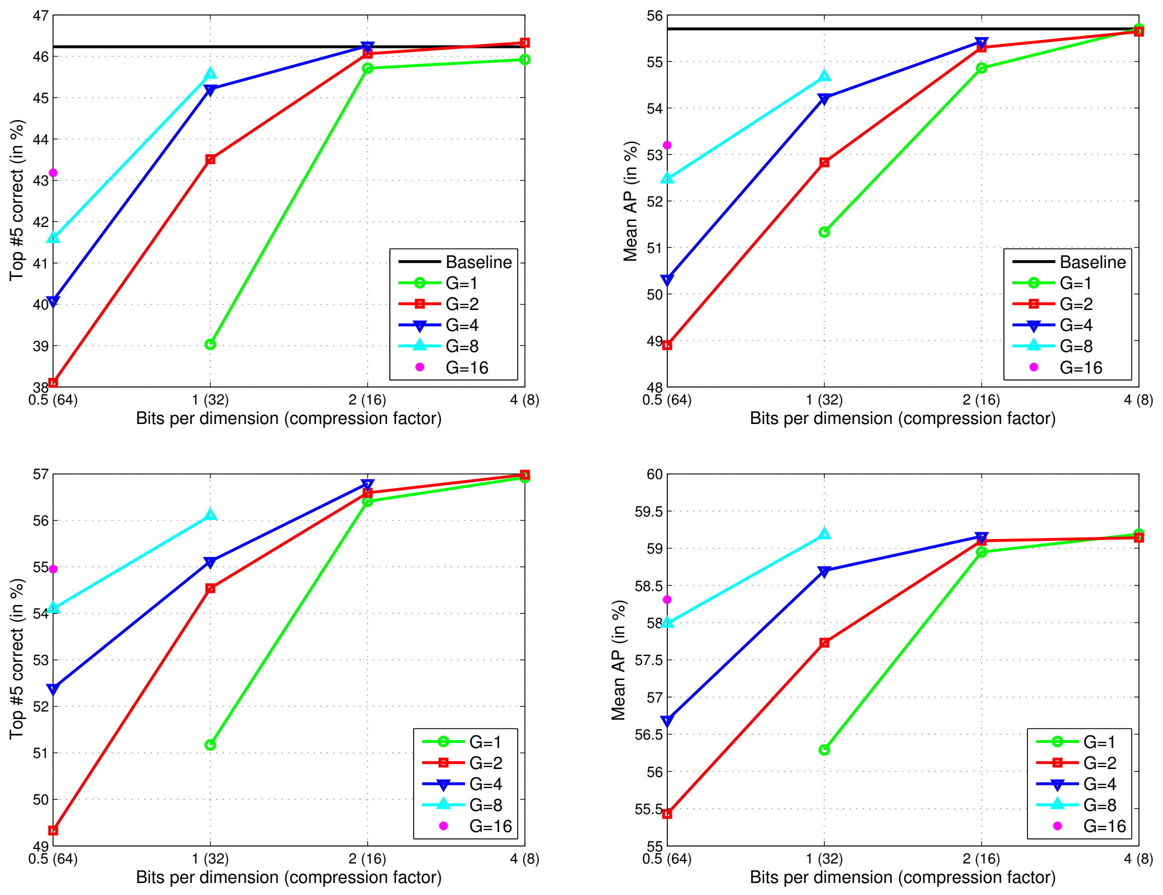


Figure 3. PQ results on 32,768 dimensional (top) and 262,144 dimensional (bottom) signatures as a function of the number b of bits per dimension and the group size G (without sparsity encoding). Left: ILSVRC2010. Right: FLICKR1M.

	Dimensions				
	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}
ILSVRC2010	46.2 (-)	43.8 (0.01)	40.4 (0.05)	35.6 (0.05)	29.0 (0.12)
FLICKR1M	55.7 (-)	54.1 (0.11)	51.5 (0.23)	47.5 (0.21)	41.9 (0.26)

Table 1. HK results on ILSVRC2010 (top-5 accuracy) and FLICKR1M (mean AP) with the balanced unbiased version. $2^{15} = 32,768$ dimensions corresponds to the uncompressed baseline (no spatial pyramid).

[28] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010.

[29] K. van de Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE PAMI*, 32(9), 2010.

[30] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

[31] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from Flickr groups using stochastic intersection kernel machines. In *ICCV*, 2009.

[32] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[33] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *ICML*, 2009.

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

[35] S. Yan, X. Zhou, M. Liu, M. Hasegawa-Johnson, and T. Huang. Regression from patch-kernel. In *CVPR*, 2008.

[36] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[37] J. Yang, K. Yu, and T. Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, 2010.

[38] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 73(2), 2007.

[39] Z. Zhou, K. Yu, T. Zhang, and T. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.

Authorized licensed use limited to: New York University AbuDhabi Campus. Downloaded on February 25,2024 at 00:04:40 UTC from IEEE Xplore. Restrictions apply.