The devil is in the details: an evaluation of recent feature encoding methods

Ken Chatfield
http://www.robots.ox.ac.uk/~ken

Victor Lempitsky
http://www.robots.ox.ac.uk/~vilem

Andrea Vedaldi
http://www.vlfeat.org/~vedaldi

Andrew Zisserman
http://www.robots.ox.ac.uk/~az

Department of Engineering Science,
Oxford University

Abstract

A large number of novel encodings for bag of visual words models have been proposed in the past two years to improve on the standard histogram of quantized local features. Examples include locality-constrained linear encoding [23], improved Fisher encoding [17], super vector encoding [27], and kernel codebook encoding [20]. While several authors have reported very good results on the challenging PASCAL VOC classification data by means of these new techniques, differences in the feature computation and learning algorithms, missing details in the description of the methods, and different tuning of the various components, make it impossible to compare directly these methods and hard to reproduce the results reported. This paper addresses these shortcomings by carrying out a rigorous evaluation of these new techniques by: (1) fixing the other elements of the pipeline (features, learning, tuning); (2) disclosing all the implementation details, and (3) identifying both those aspects of each method which are particularly important to achieve good performance, and those aspects which are less critical. This allows a consistent comparative analysis of these encoding methods. Several conclusions drawn from our analysis cannot be inferred from the original publications.

1 Introduction

The typical object recognition pipeline is composed of the following three steps: (i) extraction of local image features (*e.g.*, SIFT descriptors), (ii) encoding of the local features in an image descriptor (*e.g.*, a histogram of the quantized local features), and (iii) classification of the image descriptor (*e.g.*, by a support vector machine). Recently several authors have focused on improving the second component, *i.e.* the encoding of the local features in global image statistics. The baseline method is to compute a spatial histogram of visual words (quantized local features) and was introduced in [4, 11, 19]. Recent advances replace the hard quantization of features involved in this method with alternative encodings that retain more information about the original image features. This has been done in two ways: (1) by expressing features as *combinations* of visual words (*e.g.*, soft quantization [20], local linear

encoding [23]), and (2) by recording the *difference* between the features and the visual words (*e.g.*, Fisher encoding [17], super-vector encoding [27]).

Papers have flourished [17, 23, 24, 25, 27], and some have reported very good results on challenging benchmarks such as the PASCAL VOC classification challenge [5]. Unfortunately, it is well known that the performance of object recognition methods depends very strongly on all the stages of the pipeline, and especially on the feature computation step. Moreover, papers sometimes lack descriptions of critical details that are required to reproduce the proposed methods, and do not discuss certain aspects such as memory consumption and speed that are of critical importance in applications. All in all, it is very hard to assess the relative merits of these new encodings.

In this paper, we attempt to address this situation. We compare five recent encoding methods: locality-constrained linear encoding, super vector encoding, improved Fisher encoding, kernel codebook encoding, and the standard spatial histograms baseline. The contribution offered over that of the original papers is that we: (1) compare the methods by fixing the underlying representation (SIFT descriptors), learning framework (linear SVM), and their tuning; (2) disclose the source code used to generate the experimental results and describe all the implementation details (including some that were omitted in the original publications and that were obtained from personal communications with the authors); (3) analyse which aspects of the different constructions are important for performance and which are not. The overall picture that emerges cannot be inferred from the original publications alone.

In Sect. 2 we discuss the details of the encoding approaches. We begin by considering those parts of the pipeline leading up to the encoding stage including: computation of low level features, quantization by k-means, large scale k-means, Gaussian mixture models (Sect. 2.1) before considering the encoding methods themselves: the histogram, kernel codebook, Fisher, super-vector, and locality-constrained encodings (Sect. 2.2). This is followed by a discussion of the part of the pipeline after the encoding stage, covering the use of non-linear kernels, spatial binning, and learning (Sect. 2.3). The detailed experimental settings and the results of the methods and their variations on PASCAL VOC 2007 [5] and Caltech-101 [6] datasets are reported in Sect. 3. The last part of the paper (Sect. 4) contains an analysis of comparative performance.

2 Methods

2.1 Local descriptors and quantization

The purpose of the encodings compared in this paper is to compute global image descriptors from large set of local descriptors. The local descriptors are fixed in all experiments to be SIFT descriptors [13] extracted with a spatial stride of between two and five pixels, and at four scales, defined by setting the width of the SIFT spatial bins to 4, 6, 8 and 10 pixels respectively. The rotation of the SIFT features is fixed to a constant value. Additionally, low contrast SIFT descriptors are detected by measuring the average of the gradient magnitude (in the descriptor support) and dropped when this magnitude is below a certain threshold (about 5% of the SIFT features in the PASCAL dataset satisfy this condition). To simplify reproducibility, the SIFT descriptors are computed by using the `vl_phow` command included in the publicly available VLFeat toolbox [21], version 0.9.13. Apart from the stride parameter, the defaults options are used, along with the `'fast'` option which selects the faster (but slightly approximated) extraction algorithm. This implementation is very close to

Lowe's original but is much faster for dense feature extraction, requiring well under a second for a typical PASCAL image. In some cases, the dimensionality of the SIFT descriptor will be reduced by using PCA [8].

K-means clustering. All the encodings considered here are based on the idea of partitioning the local descriptor space into informative regions whose internal structure can be disregarded or parametrized linearly. These regions are also called *visual words* and a collection of visual words is called a *visual vocabulary*.

k-means clustering is probably the most common way of constructing visual vocabularies. Given a set $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbf{R}^D$ of N training descriptors (to which PCA reduction may have been applied), k-means seeks K vectors $\mu_1, \ldots, \mu_K \in \mathbf{R}^D$ and a data-to-means assignments $q_1, \ldots, q_N \in \{1, \ldots, K\}$ such that the cumulative approximation error $\sum_{i=1}^{N} \|\mathbf{x}_i - \mu_{q_i}\|^2$ is minimized. Two k-means clustering algorithms are considered. The first one is the standard *Lloyd's algorithm* [12], an optimization method that alternates between seeking the best means given the assignments ($\mu_k = \text{avg}\{\mathbf{x}_i : q_i = k\}$), and seeking then the best assignments given the means

$$q_{ki} = \underset{k}{\text{argmin}} \|\mathbf{x}_i - \mu_k\|^2. \tag{1}$$

The second algorithm, used for large vocabularies, is an approximated version of Lloyd's one where the data-to-clusters assignments (1) are computed by using an *approximate nearest neighbour* (ANN) algorithm (we use as ANN the randomized best-bin-first KD-tree forest proposed in [16], as implemented in the VLFeat toolbox).

GMM clustering. A *Gaussian mixture model* (GMM) $p(\mathbf{x}|\theta)$ is the probability density on $\mathbf{R}^D$ given by

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} p(\mathbf{x}|\mu_k, \Sigma_k) \pi_k, \quad p(\mathbf{x}|\mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^D \det \Sigma_k}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^\top \Sigma_k^{-1}(\mathbf{x}-\mu_k)},$$

where $\theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_K, \mu_K, \Sigma_K)$ is the vector of parameters of the model, including the prior probability values $\pi_k \in \mathbf{R}_+$ (which sum to one), the means $\mu_k \in \mathbf{R}^D$, and the positive definite covariance matrices $\Sigma_k \in \mathbf{R}^{D \times D}$ of each Gaussian component. Here the covariance matrices are assumed to be *diagonal*, so that the GMM is fully specified by $(2D+1)K$ scalar parameters. The parameters are learnt by expectation maximization (EM, [14]) from a training set of descriptors $\mathbf{x}_1, \ldots, \mathbf{x}_N$, but making sure that the diagonal covariances of the components are never smaller than 0.01 times the overall diagonal covariance of the data. The GMM defines then the soft data-to-cluster assignments

$$q_{ki} = \frac{p(\mathbf{x}_i|\mu_k, \Sigma_k)\pi_k}{\sum_{j=1}^{K} p(\mathbf{x}_i|\mu_j, \Sigma_j)\pi_j}, \quad k = 1, \ldots, K. \tag{2}$$

2.2 Encodings

Histogram encoding is, as the name suggests, a histogram of the quantized local descriptors and constitutes the baseline encoding upon which the other methods improve. The construction of the encoding starts by learning a k-means visual vocabulary $\mu_1, \ldots, \mu_K$ (Sect. 2.1). Given a set of descriptors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ sampled from an image, let q_{ki} be the assignments of each descriptor $\mathbf{x}_i$ to the corresponding visual word as given by (1). The histogram encoding of the set of local descriptors is the non-negative vector $\mathbf{f}_{\text{hist}} \in \mathbf{R}^K$ such that $[\mathbf{f}_{\text{hist}}]_k = |\{i : q_{ki} = k\}|$.

Kernel codebook encoding [[18, 20]] is a variant in which descriptors are assigned to visual words in a soft manner. More specifically, each descriptor is encoded as $[\mathbf{f}_{\text{kcb}}(\mathbf{x}_i)]_k = K(\mathbf{x}_i, \mu_k) / \sum_{j=1}^{K} K(\mathbf{x}_i, \mu_j)$ where $K(\mathbf{x}, \mu) = \exp(-\frac{\gamma}{2} \|\mathbf{x} - \mu\|^2)$, and a set of N descriptors extracted from an image as $\mathbf{f}_{\text{kcb}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{f}_{\text{kcb}}(\mathbf{x}_i)$. Both give an encoding of size K.

Fisher encoding [[17]] captures the average first and second order differences between the image descriptors and the centres of a GMM, which can be thought of as a soft visual vocabulary. The construction of the encoding starts by learning a GMM model θ (Sect. 2.1). Given a set of descriptors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ sampled from an image, let $q_{ki}, k = 1, \ldots, K, i = 1, \ldots, N$ be the soft assignments of the N descriptors to the K Gaussian components as given by (2). For each $k = 1, \ldots, K$, define the vectors

$$\mathbf{u}_k = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_{ki} \Sigma_k^{-\frac{1}{2}} (\mathbf{x}_i - \mu_k), \quad \mathbf{v}_k = \frac{1}{N\sqrt{2\pi_k}} \sum_{i=1}^{N} q_{ki} \left[(\mathbf{x}_i - \mu_k) \Sigma_k^{-1} (\mathbf{x}_i - \mu_k) - 1 \right].$$

Note that, since the covariance matrices Σ_k are assumed to be diagonal, computing these quantities is quite fast. The Fisher encoding of the set of local descriptors is then given by the concatenation of $\mathbf{u}_k$ and $\mathbf{v}_k$ for all K components, giving an encoding of size $2DK$

$$\mathbf{f}_{\text{Fisher}} = [\mathbf{u}_1^\top, \mathbf{v}_1^\top, \ldots \mathbf{u}_K^\top, \mathbf{v}_K^\top]^\top. \tag{3}$$

Super vector encoding [[27]] is similar to the Fisher encoding. There are two variants of this encoding, based on hard assignment to the nearest codeword or soft assignment to several near neighbours. For the hard super vector encoding, let $q_{ki} = 1$ if $\mathbf{x}_i$ is assigned to cluster k by k-means and 0 otherwise (Sect. 2.1). [[27]] does not specify how q_{ki} are set in the soft assignment case. We defined q_{ki} to be essentially the same as for the Fisher encoding, setting $\Sigma_k = \sigma^2 \cdot \mathbf{I}$ and σ being twice the mean distance between points and means within the k-means algorithm and $\mathbf{I}$ the identity matrix. Define

$$p_k = \frac{1}{N} \sum_{i=1}^{N} q_{ki}, \quad s_k = s\sqrt{p_k}, \quad \mathbf{u}_k = \frac{1}{\sqrt{p_k}} \sum_{i=1}^{N} q_{ki} (\mathbf{x}_i - \mu_k),$$

where s is a constant chosen to balance s_k with $\mathbf{u}_k$ numerically. Then the super vector encoding is given by

$$\mathbf{f}_{\text{super}} = [s_1, \mathbf{u}_1^\top, \ldots, s_K, \mathbf{u}_K^\top]^\top. \tag{4}$$

This gives an encoding of size $K(D+1)$. Compared to the Fisher encoding, the super vector encoding: (1) considers only the first order differences $\mathbf{u}_k$ between features and cluster centres; (2) adds the components s_k which represent the mass of each cluster; (3) normalizes each cluster by the square root of the *posterior* probability $\sqrt{p_k}$ rather than of the *prior* probability $\sqrt{\pi_k}$.

Locality-constrained linear (LLC) encoding (LLC) [[23]] projects each image descriptors $\mathbf{x}_i$ down to the local linear subspace spanned by the $M \ll K$ visual words closest to $\mathbf{x}_i$. Let $\mu_1, \ldots, \mu_K$ be a visual vocabulary learnt by using k-means (Sect. 2.1). Let $\sigma_1, \ldots, \sigma_M$ be the indices of the M visual words μ_k closer to $\mathbf{x}_i$ (in Euclidean distance) and denote them collectively as $B = [\mu_{\sigma_1}, \ldots, \mu_{\sigma_M}]$. Let $\alpha \in \mathbb{R}^M$ be the coefficients of the approximation $\mathbf{x}_i \approx B\alpha$ (given below). The LLC encoding of the descriptor $\mathbf{x}_i$ is the K-dimensional vector $\mathbf{f}_{\text{LLC}}(\mathbf{x}_i)$ of all zeroes except for the M components $[\mathbf{f}_{\text{LLC}}(\mathbf{x}_i)]_{\sigma_m} = \alpha_m, m = 1, \ldots, M$. The LLC encoding of a set of descriptors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ is then obtained by max-pooling: $[\mathbf{f}_{\text{LLC}}]_j = \max_{i=1,\ldots,N} [\mathbf{f}_{\text{LLC}}(\mathbf{x}_i)]_j$. As with both histogram encoding and kernel codebook encoding, this gives an encoding of size K.

To project $\mathbf{x}_i$ down to the span of the visual words B, one solves the problem $\alpha^* = \operatorname{argmin}_{\mathbf{1}^\top \alpha=1} \|\mathbf{x}_i - B\alpha\|^2 + \beta \|\alpha\|^2$, where β is a small regularization constant. After a few algebraic manipulations, one gets the solution $\alpha^* \propto (\Delta_i^\top \Delta_i + \beta \mathbf{I})^{-1} \Delta_i \mathbf{1}$, where $\Delta_i = [\mathbf{x}_i - \mu_{\sigma_1}, \ldots, \mathbf{x}_i - \mu_{\sigma_M}]$ (the norm of α^* is fixed by the constraint $\mathbf{1}^\top \alpha = 1$).

2.3 Spatial binning, kernels, and learning

Non-linear kernels. For training with a support vector machine (SVM) all of the encodings $\mathbf{f}, \mathbf{g} \in \mathbb{R}^D$ discussed so far can be used directly with a linear kernel $K(\mathbf{f}, \mathbf{g}) = \langle \mathbf{f}, \mathbf{g} \rangle$. While linear kernels are very efficient [8] for training, non-linear kernels tend to yield better classification accuracy [26]. A class of kernels that are almost as efficient as the linear ones but usually much more accurate are the additive homogeneous kernels [22] $K(\mathbf{f}, \mathbf{g}) = \sum_{i=1}^{D} k(\mathbf{f}_i, \mathbf{g}_i)$, where k is itself a kernel on the non-negative reals. Examples of k include the Hellinger's (Bhattacharya's) kernel $k(f, g) = \sqrt{fg}$ and the χ^2 kernel $k(f, g) = 2fg/(f+g)$. While these kernels are normally defined for non-negative vectors (histograms), they can be extended to arbitrary vectors by setting $k'(f, g) = \operatorname{sign}(fg) k(|f|, |g|)$.

The computational advantage of using additive kernels is that they can be represented as linear ones up to the computation of an efficient *feature map*. For instance, for the Hellinger's kernel it suffices to consider the feature map defined by $[\Psi(\mathbf{f})]_i = \sqrt{\mathbf{f}_i}$, as in fact $K(\mathbf{f}, \mathbf{g}) = \langle \Psi(\mathbf{f}), \Psi(\mathbf{g}) \rangle = \sum_{i=1}^{D} \sqrt{\mathbf{f}_i} \sqrt{\mathbf{g}_i} = \sum_{i=1}^{D} \sqrt{\mathbf{f}_i \mathbf{g}_i}$. For the χ^2 and other kernels one can use the approximated feature maps introduced by [22], which are nearly as efficient.

SVMs usually work better if the data is properly normalized [22]. Here all encodings are used in combination with linear SVMs (potentially after the computation of a suitable feature map as explained above), so that the optimal normalization is l^2 [22].

Spatial binning. A standard way of introducing weak geometry in a bag-of-words representation is the use of spatial histograms [7, 11]. The concept can be extended to any of the encodings analysed here by computing one encoding for each spatial region and then stacking the results. Note that each spatial region is normalized individually prior to stacking. The l^1 norm is employed in the case of histogram and kernel codebook encodings, with the l^2 norm used for all other methods as suggested by the original publications. This was found to work better than the original approach [11] of normalizing groups of regions according to their size. After stacking, any feature map is applied to the entire histogram and finally the entire histogram is l^2 normalized to make it suitable for use within our linear SVM framework.

In the experiments the spatial regions are obtained by dividing the image in 1×1, 3×1 (three horizontal stripes), and 2×2 (four quadrants) grids, for a total of 8 regions, for the PASCAL VOC data, and in 1×1, 2×2, and 4×4, for a total of 21 regions, for the Caltech-101 data.

Pooling. When computing the encoding for each spatial region, the image features can be pooled in one of two ways: sum pooling, in which case the encodings of SIFT features in a given region are combined additively, or max pooling, in which case each bin in the encoding is assigned a value equal to the maximum across SIFT feature encodings in that region. We use max pooling for the LLC encoding, and sum pooling for all other methods, as in the original publications.

Learning. All the experiments use a linear SVM on top of each encoding. The parameter C of the SVM (regularization-loss trade off) is determined on a validation set (on the provided

train and val split in the PASCAL VOC data and on a random split in Caltech-101).

3 Experiments

This section describes the experimental setup, including the parameters used for each of the encodings. The PASCAL VOC 2007 data contains about 10,000 images split into train, validation, and test sets, and labelled with twenty object classes. A 1-vs-rest SVM classifier for each class is learnt and evaluated independently and the performance is measured as mean Average Precision (mAP) across all classes. The Caltech-101 data contains a variable number of images for 102 categories (one is background). Of this data, we take three splits at random, each containing 30 training images and up to 30 testing images per class. Similarly to PASCAL, 102 1-vs-rest SVM classifiers are learnt and evaluated, one for each class, but in the case of Caltech-101 the evaluation is conducted in terms of the average classification accuracy across all classes instead of the mAP.

Baseline: histogram encoding. The baseline encoding is a histogram of visual words obtained with hard quantization (Sect. 2.2). For each dataset a single dictionary of visual words is learnt for all classes. The number of visual words varies between 600 and 8,000 for the Caltech-101 data and between 4,000 and 25,000 for the PASCAL VOC data.

Kernel codebook encoding. This encoding is analogous to the histogram encoding, but uses soft quantization. It requires a single additional parameter γ (Sect. 2.2) which is determined on the data validation split of the data as for the other parameters. The codeword uncertainty method of [20] is used. For efficiency, each descriptor is encoded by considering only the top five nearest visual words.

LLC encoding. Similarly to the kernel codebook encoding, this encoding requires setting the number of neighbour visual words M considered for each encoded descriptor (Sect. 2.2). This is again set to five. The parameter β in the computation of the projections is set to 10^{-4}.

Improved Fisher encoding. As suggested in [17], we used GMM with $K = 256$ components (Sect. 2.1) after reducing the dimensionality of the SIFT descriptors to 80 by using PCA. PCA was found to improve the accuracy and decrease the memory footprint of this representation. Hellinger's kernel is also used with this encoding, which amounts to square rooting the features (up to rectification) and then l^2-normalizing the result. This was also the method used in [17], and is treated in full generality in Sect. 2.3.

Super vector encoding. Since [27] reports very similar performance for codebooks of size 1024 and 2048, we chose the former size to reduce computation and memory consumption, which still remained very high. In our experiments, PCA reduction to 80 dimensions degraded the performance considerably, and we therefore report the results for the original SIFT descriptors. The memory required for storing super-vectors is particularly high for Caltech-101 dataset (due to larger number of spatial bins – 21 versus 8, refer to section 2.3); therefore, only the performance on PASCAL VOC 2007 is reported.

We used the soft-assignment variant of super vector encoding with 5 (approximate) nearest neighbours used to encode each descriptor. At least with our implementation of soft assignment, the method turned out to be very sensitive to the number of nearest neighbours, with performance for 20 nearest neighbours and 1 nearest neighbour (*i.e.*, hard-assignment) being much worse (-5% and -7% mAP respectively) than for 5 nearest neighbours.

Method				mAP	✈	🚲	🐦	⛵	🍾	🚌	🚗	🐱
(a) FK	Lin	ss3	256	**61.69**	78.97	67.43	51.94	70.92	30.79	72.18	79.94	61.35
(b) SV	Lin	ss3	1024	58.16	74.32	63.79	47.02	69.44	29.06	66.46	77.31	60.18
(c) LLC	Lin	ss2	25k	57.60	71.05	62.85	47.40	67.67	25.21	62.70	77.02	59.59
(d) LLC-F	Lin	ss2	25k	59.32	74.10	64.92	51.48	68.33	27.18	62.89	78.44	61.39
(e) VQ	Chi	ss2	25k	56.07	70.00	58.90	42.86	66.75	26.59	62.27	75.67	57.09
(f) LLC	Lin	ss3	25k	57.27	71.35	62.65	46.12	68.98	26.04	63.92	76.98	59.71
(g) LLC	Sqr	ss3	25k	56.71	71.24	61.75	42.73	68.21	25.85	62.33	76.40	59.31
(h) LLC	Chi	ss3	25k	57.66	72.41	62.19	47.30	68.91	25.78	63.95	77.27	59.83
(i) LLC-F	Lin	ss3	25k	59.74	74.17	65.39	51.15	69.69	28.67	64.40	78.48	63.00
(j) VQ	Chi	ss3	25k	55.30	70.10	59.24	44.14	66.34	26.79	60.88	75.62	55.42
(k) KCB	Chi	ss3	25k	56.26	70.83	60.60	44.50	66.52	27.02	62.07	76.29	57.61
(l) LLC	Lin	ss5	25k	56.96	69.82	61.63	46.71	68.27	25.66	63.78	76.32	59.83
(m) LLC-F	Lin	ss5	25k	58.70	73.44	62.90	50.22	67.90	27.85	64.35	77.91	62.44
(n) VQ	Chi	ss5	25k	53.87	68.74	57.14	41.24	64.54	25.20	61.12	74.06	53.22
(o) LLC	Lin	ss3	14k	56.18	70.71	59.67	44.81	67.20	26.03	60.99	76.25	58.54
(p) VQ	Chi	ss3	14k	54.82	69.09	58.61	41.27	66.30	26.49	61.46	75.42	55.77
(q) LLC	Lin	ss3	10k	56.01	69.66	60.44	44.21	67.78	24.66	61.84	75.42	57.70
(r) VQ	Chi	ss3	10k	54.98	69.56	57.97	42.86	65.84	23.52	61.06	75.89	55.55
(s) LLC	Lin	ss3	4k	53.79	69.83	57.63	42.04	66.46	22.44	55.62	72.77	56.98
(t) LLC	Sqr	ss3	4k	52.07	68.52	54.62	40.14	65.34	21.53	51.89	71.54	55.19
(u) LLC	Chi	ss3	4k	53.47	70.17	56.20	42.73	65.27	22.23	55.18	72.78	56.95
(v) LLC-1	Lin	ss3	4k	36.06	53.39	43.20	22.47	46.32	11.40	29.48	64.66	45.41
(w) LLC-F	Lin	ss3	4k	55.87	72.27	61.41	44.08	67.85	24.97	57.92	75.40	59.44
(x) VQ	Sqr	ss3	4k	**51.97**	67.29	55.22	36.58	64.42	21.89	56.31	72.90	52.11
(y) VQ	Chi	ss3	4k	53.42	68.65	57.04	39.86	64.59	21.96	58.79	73.89	53.77
(z) VQ	Lin	ss3	4k	**46.54**	60.63	48.80	32.76	58.54	16.26	50.44	68.42	45.97
(α) KCB	Chi	ss3	4k	**54.60**	69.82	59.20	41.97	64.85	23.90	59.02	74.98	54.63

Table 1: Image classification results using Pascal VOC 2007 dataset *(continued on next page)*
VQ – baseline method; **FK** – Fisher kernel; **SV** – super vector coding; **KCB** – kernel codebook; **LLC** – locally-constrained linear coding; **LLC-F** – LLC encoding with original+left-right flipped training images; **LLC-1** – L1-normalized LLC encoding; **Lin/Sqr/Chi** – linear/hellinger/χ^2 kernel map; third column: SIFT sampling density; fourth column: visual vocabulary size

The accuracy achieved by the super vector encoding in our experiments (Table 1) is competitive with other methods (for example, it compares favourably to the 58.3% mAP reported in [17], albeit at a finer dense SIFT sampling of every 3 pixels compared to the 16 pixel spacing used in that paper, or the 59.3% mAP reported in [23]), yet still quite far from the performance (64% mAP) reported in [27]. However, as we learnt from personal communication with the authors, [27] used nontrivial modifications not discussed in the paper to achieve those results (these include using LDA to compute the SVM kernel and second order information as in the Fisher encoding). According to the authors, the performance achieved with our implementation is representative of their method, given that we did not apply these additional modifications.

4 Analysis

The results of our experiments over PASCAL VOC 2007 are shown in Table 1. As expected, all the recently introduced methods improve the classification accuracy over the spatial his-

(a) FK	55.98	49.61	58.40	44.77	78.84	70.81	84.96	31.72	51.00	56.41	80.24	57.46
(b) SV	50.19	46.46	51.86	44.07	77.85	67.12	83.07	27.56	48.50	51.10	75.50	52.26
(c) LLC	54.24	45.27	51.56	44.24	77.52	67.05	83.29	27.57	45.73	53.62	76.01	52.32
(d) LLC-F	54.41	47.24	52.75	44.55	78.10	68.48	83.73	29.90	50.95	55.48	78.55	53.62
(e) VQ	53.80	41.35	51.55	42.57	76.33	65.11	82.98	26.70	43.47	52.18	74.27	50.87
(f) LLC	53.96	46.34	52.10	42.39	77.17	67.15	83.36	23.11	44.45	52.12	75.36	52.21
(g) LLC	53.91	45.63	49.61	42.91	75.50	65.93	83.03	27.10	43.82	52.28	74.71	51.86
(h) LLC	54.25	45.97	51.11	43.22	76.74	67.07	83.49	27.73	44.77	52.77	76.00	52.47
(i) LLC-F	55.31	49.52	53.03	44.60	78.52	68.85	84.24	28.44	50.47	55.18	77.56	54.17
(j) VQ	53.12	40.40	51.27	39.16	76.87	63.09	82.16	20.96	42.96	51.40	75.13	50.94
(k) KCB	54.31	41.64	51.91	41.05	76.40	66.06	82.97	23.86	43.66	51.32	75.10	51.40
(l) LLC	53.17	44.89	51.03	40.70	76.88	66.84	82.78	26.19	45.46	52.17	75.11	52.03
(m) LLC-F	54.12	47.33	52.31	42.90	77.69	68.39	83.28	27.43	50.06	54.41	77.36	51.73
(n) VQ	52.08	38.96	48.66	38.64	74.84	62.40	81.24	25.08	40.46	49.35	72.19	48.22
(o) LLC	52.62	44.06	49.17	38.65	75.90	66.61	82.30	27.19	44.63	52.02	74.55	51.64
(p) VQ	52.68	40.18	50.58	38.14	75.25	63.87	82.23	21.36	42.19	51.72	74.10	49.64
(q) LLC	52.90	45.11	49.78	39.89	75.57	65.29	82.02	25.96	43.20	51.23	74.82	52.64
(r) VQ	52.97	41.09	49.64	41.28	76.33	64.24	81.94	21.49	41.91	52.95	74.06	49.51
(s) LLC	51.72	42.82	46.14	39.47	74.08	62.04	80.92	24.50	38.81	49.35	71.24	51.03
(t) LLC	50.83	40.10	42.63	39.47	71.69	58.57	79.74	22.89	38.98	47.78	70.87	49.09
(u) LLC	52.13	41.51	43.94	40.71	73.24	60.58	81.09	24.28	38.90	48.56	71.99	50.96
(v) LLC-1	32.84	19.06	21.34	22.72	53.28	33.59	68.74	11.38	6.17	37.97	60.44	37.30
(w) LLC-F	53.02	43.92	47.68	39.90	75.95	62.77	81.93	25.25	45.23	51.69	74.07	52.70
(x) VQ	51.51	38.23	46.50	34.99	74.62	60.71	80.05	18.79	37.13	50.22	71.71	48.32
(y) VQ	52.40	38.57	49.20	36.85	75.59	61.59	81.63	20.47	40.05	50.92	73.39	49.21
(z) VQ	48.25	28.31	41.73	30.21	70.51	55.24	77.79	14.79	31.37	39.73	68.27	42.73
(α) KCB	52.49	40.48	50.53	37.98	76.03	63.73	82.47	22.31	43.08	50.96	73.97	49.68

Table 1: Image classification results on Pascal VOC 2007 dataset *(continued from previous page)*

Method		256	600	codebook size 1500	2000	4000	8000
(a) FK	Lin	77.78 ± 0.56	–	–	–	–	–
(b) LLC	Lin	–	73.10 ± 1.09	74.84 ± 0.67	75.75 ± 0.71	76.15 ± 0.59	76.95 ± 0.39
(c) LLC	Chi	–	72.30 ± 1.08	74.23 ± 0.62	75.24 ± 0.71	75.95 ± 0.57	76.62 ± 0.61
(d) VQ	Chi	–	72.65 ± 0.77	73.62 ± 0.51	73.93 ± 0.79	74.41 ± 1.04	74.23 ± 0.65
(e) KCB	Chi	–	73.38 ± 0.65	75.24 ± 0.63	75.50 ± 0.65	75.92 ± 0.63	75.93 ± 0.57

Table 2: Image classification results on Caltech-101 dataset (30 training images)

togram of visual words baseline. This concurs with the fact that information is lost when a local descriptor is replaced with (assigned to) the nearest codeword. A detailed analysis follows; we will use letters in square brackets to indicate the corresponding rows of Table 1.

For small codebook sizes, the kernel codebook encoding with χ^2 kernel performs better than the newer LLC-encoding ([α] vs [s]), while the baseline encoding with χ^2 kernel performs almost as well [y] (the performance degrades dramatically with linear kernel [z]). For larger codebook sizes, LLC encoding performs better than the baseline and the kernel codebook ([f] vs [j] vs [k]), but the difference is not large. However, it is interesting to note that LLC-encoding using the linear kernel achieves comparable performance to the χ^2 kernel across different vocabulary sizes and outperforms the results using the Hellinger's kernel

feature map ([f] vs [g] and [h]; [s] vs [t] and [u]). This suggests that the linear kernel is sufficient to achieve good performance with the encoding, avoiding the computational expense of applying a non-linear kernel – even through a more efficient feature map.

Our comparisons suggest that the Fisher encoding [a] and super-vector encoding [b] have an edge over other encoding methods, although for PASCAL the advantage is not as dramatic as portrayed in the respective papers (cf. *e.g.* [b] vs [j]). The Fisher encoding also is the best performing in the Caltech-101 experiment (Table 2). Thus it can be concluded that encoding the relative displacement between a descriptor and a codeword, as with both the Fisher encoding and super-vector encoding, successfully retains some extra information lost in the quantization process.

Vocabulary size. The PASCAL experiments clearly demonstrate that larger vocabularies lead to higher accuracy. While the law of diminishing returns is clearly in place, it is most likely that the performance is not saturated even at 25K visual words ([s]-[q]-[o]-[f], [y]-[r]-[p]-[j]). A similar trend (although for a much smaller vocabulary size) is observed in Table 2. It should be noted that in the case of LLC encoding, even at a vocabulary size of 8,000 the performance achieved appears to be still increasing suggesting that further gains could be achieved by increasing the vocabulary size even further.

Sampling density/data augmentation. As has been observed in multiple previous works, the sampling density of the local descriptors matters, with denser sampling yielding higher accuracy ([f] vs [l]; [i] vs [m]; [j] vs [n]). The performance however saturates around the maximum sampling density that we considered here ([c] is better than [f] but [d] is not better than [i]). Nonetheless, the performance can be further improved considerably via simple data augmentation tricks, such as adding left-right flipped versions of the training data to the training set ([c] vs [d], [f] vs [i], [l] vs [m]). Further data augmentation strategies are likely to improve the performance even more.

Size of encoding. Soft-assignment methods (LLC, kernel codebooks) gain considerably from large (*e.g.* 25,000) vocabularies resulting in a correspondingly large encoding size (number of words times number of spatial bins, *e.g.* 200K floats per image). Fisher kernel and super-vector encodings produce even larger vectors (*e.g.* with the encoding parameters suggested in [27], each VOC image is encoded with a 1.4-million dimensional dense vector). In our implementation, the size of the Fisher kernel encoding is smaller than super-vector encoding (~ 1.26MB vs ~ 3.8MB) due to the smaller codebook employed and the use of PCA.

With our sampling density, neither of the encodings results in particularly sparse vectors: on PASCAL VOC, sparsity ranges from 30% non-zeros for the baseline encoding to close to 100% non-zeros for super-vector encoding and Fisher kernel encoding. Consequently, fitting the encoded training data in memory is not possible even for a relatively small datasets like PASCAL VOC 2007 and Caltech-101. In these cases computing the kernel matrix and solving the problem in the dual by using *e.g.* LIBSVM [2] reduces the memory footprint significantly, and this is the approach we use.

Speed of encoding. For the hard and soft-assignment based methods (LLC, kernel codebooks, the baseline) the encoding time is dominated by the approximate nearest neighbour search, which increases sub-linearly yet considerably not only with the size of the vocabulary but also the number of nearest neighbours sought. Therefore, the overhead of LLC and kernel codebook (KCB) encodings over the baseline are significant as a result of the additional time spent searching for K nearest neighbours per feature instead of just one. We also use a higher bound on the maximum number of comparisons per feature in our approximate

nearest neighbour algorithm for the LLC and KCB encodings to maintain performance when searching for the greater number of nearest neighbours required by these methods (we use a maximum of 500 comparisons per feature for LLC and KCB, which in our experiments both use $K = 5$ nearest neighbours, compared to 25 comparisons for the baseline method, which requires just the first nearest neighbour) leading to an encoding time of 20 seconds per image for LLC and 30 seconds for KCB compared to 0.5 seconds for the baseline encoding. The time required to compute the approximate nearest neighbours comprises the majority of this – around 17 seconds for the 5-NN search for LLC and KCB compared to < 0.5 seconds for the 1-NN search in the case of the baseline encoding. The relative sluggishness of the KCB encoding can be explained by the fact our implementation is unoptimized MATLAB code compared to the C++ implementations used for the baseline and LLC encodings. All timings are for runs on a 3.07GHz Intel CPU using a vocabulary of 10K visual words.

Despite their size, super-vector encodings and Fisher kernel encodings can potentially be faster (since they have to search neighbours/compute distances within a much smaller vocabulary). To achieve full speed, Fisher coding would probably benefit from some sparsification (*i.e.* $q_{ki} \approx 0$ should be set to 0), which we did not use. Super-vector encoding in our MATLAB implementation takes about 12 seconds per image and Fisher encoding about 9 seconds per image using a combined C++/MATLAB implementation.

It is noteworthy that for all encodings the cost of computing the SIFT descriptors is much smaller than the time required for any of the encodings (< 0.5 seconds per image).

5   Conclusions

We have presented a detailed empirical evaluation of several recent encoding methods for bag-of-word models for object recognition. In some cases, our observations differed significantly from the one presented in the original publications, which emphasizes the importance of controlling carefully all conditions when comparing different representations. We publish the MATLAB code for our experiments on the web page [3], in the hope that it would provide common ground for future comparisons, *e.g.* for new papers on feature encoding.

The list of encoding methods evaluated in this work is not exhaustive. In particular, quite a few recent works (e.g. [1, 10, 15]) investigated the supervised learning of encoding and all reported an improvement in classification performance due to the introduction of supervision. In future, we plan to extend our comparison to include some of those methods.

Acknowledgements

References

[1] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *Proc. CVPR*, pages 2559–2566, 2010.

[2] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001. URL http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[3] K. Chatfield, A. Vedaldi, L. Victor, and Z. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods webpage. URL http://www.robots.ox.ac.uk/~vgg/research/encoding_eval.

[4] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[5] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool. The PASCAL visual obiect classes challenge 2007 (VOC2007) results. Technical report, Pascal Challenge, 2007.

[6] L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proc. ICCV*, 2003.

[7] K. Grauman and T. Darrel. The pyramid match kernel: Discriminative classification with sets of image features. In *Proc. ICCV*, 2005.

[8] T. Joachims. Training linear SVMs in linear time. In *Proc. KDD*, pages 217–226, 2006.

[9] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proc. CVPR*, 2004.

[10] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *PAMI*, 31(7):1294–1309, 2009.

[11] S. Lazebnik, C. Schmid, and J Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proc. CVPR*, 2006.

[12] S. Lloyd. Least square quantization in PCM. *IEEE Trans. on Information Theory*, 28 (2), 1982.

[13] D. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, pages 1150–1157, 1999.

[14] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

[15] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *Proc. NIPS*, pages 985–992, 2006.

[16] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithmic configuration. In *Proc. VISAPP*, 2009.

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

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

[19] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, pages 1470–1477, 2003.

[20] J. C. van Gemert, J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *Proc. ECCV*, 2008.

[21] A. Vedaldi and B. Fulkerson. VLFeat - An open and portable library of computer vision algorithms. In *Proc. ACM Int. Conf. on Multimedia*, 2010.

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

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

[24] J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *Proc. CVPR*, 2010.

[25] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *NIPS*, 2009.

[26] J. Zhang, M. Marszałek, S. Lazebnik, and Schmid C. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 2007.

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