Ask the locals: multi-way local pooling for image recognition

Y-Lan Boureau[1,3,⋆] Nicolas Le Roux[1,†] Francis Bach[1,†]
Jean Ponce[2,⋆] Yann LeCun[3]
[1]INRIA [2]Ecole Normale Supérieure [3]Courant Institute, New York University

Abstract

Invariant representations in object recognition systems are generally obtained by pooling feature vectors over spatially local neighborhoods. But pooling is not local in the feature vector space, so that widely dissimilar features may be pooled together if they are in nearby locations. Recent approaches rely on sophisticated encoding methods and more specialized codebooks (or dictionaries), e.g., learned on subsets of descriptors which are close in feature space, to circumvent this problem. In this work, we argue that a common trait found in much recent work in image recognition or retrieval is that it leverages locality in feature space on top of purely spatial locality. We propose to apply this idea in its simplest form to an object recognition system based on the spatial pyramid framework, to increase the performance of small dictionaries with very little added engineering. State-of-the-art results on several object recognition benchmarks show the promise of this approach.

1. Introduction

Much recent work in image recognition has underscored the importance of locality constraints for extracting good image representations. Methods that incorporate some way of taking locality into account define the state of the art on many challenging image classification benchmarks such as Pascal VOC, Caltech-101, Caltech-256, and 15-Scenes [11, 39, 42, 43, 45].

The spatial pyramid [25] has emerged as a popular framework to encapsulate more and more sophisticated feature extraction techniques [4, 11, 39, 41, 42, 45]. The global representation of an image is obtained by extracting image descriptors such as SIFT [28] or HOG [9] on a dense grid, encoding them over some learned codebook or dictionary (coding step), and then summarizing the distribution of the codes in the cells of a spatial pyramid by some well-chosen aggregation statistic (pooling step).

*WILLOW project-team, Laboratoire d'Informatique de l'Ecole Normale Supérieure, ENS/INRIA/CNRS UMR 8548.
†SIERRA project-team, Laboratoire d'Informatique de l'Ecole Normale Supérieure, ENS/INRIA/CNRS UMR 8548.

Several recent papers have focused on refining the coding step, one purpose of which is to produce representations that can be aggregated (pooled) without losing too much information in the process. Pooling, which has long been part of popular recognition architectures such as convolutional networks [26], gives robustness to small transformations of the image. It is related to Koenderink's concept of locally orderless images [24], and can be traced back to Hubel and Wiesel's seminal work on complex cells in the visual cortex [18]. The simplest pooling operation consists in averaging the feature vectors within a spatial neighborhood. One fact that makes the coding step necessary is that descriptors such as SIFT or HOG cannot be averaged with their neighbors without losing a considerable amount of information: the average of several widely different SIFT features doesn't tell us much about the content of the underlying image. Hence, coding is generally designed to produce representations that can be added with each other without diluting the signal. The original spatial pyramid proposal used vector quantization (K-means), which can be seen as a way of turning SIFT vectors into very sparse, 1-of-K codes. When obtained by averaging over spatial neighborhoods, the pooled codes can be interpreted as local histograms of visual words. Recent work has proposed to replace this hard quantization step of individual SIFT descriptors by soft vector quantization [36], or sparse coding [41], including sparse coding of local groups of SIFT descriptors (macrofeatures) instead of single ones, in our previous work [4].

While the pooling operations are often performed over local spatial neighborhoods, the neighborhoods may contain feature vectors that are very heterogeneous, possibly leading to the loss of a large amount of information about the distribution of features, as illustrated in Fig. (1). Restricting the pooling to feature vectors that are similar in the multidimensional input space (or nearby) [21, 45] remedies this problem. The usefulness of considering similar inputs for smoothing noisy data over a homogeneous sample to reduce noise without throwing out the signal has long been recognized in the image processing and denoising communities [6, 8], and has been successfully incorporated to denoising methods using sparse coding [30]. It is interesting to note that considerations of locality often pull cod-

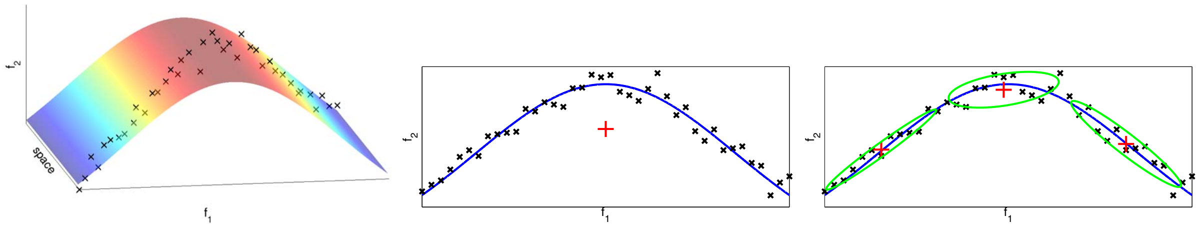


Figure 1. Cartoon representation of a distribution of descriptors that has a high curvature and is invariant to the spatial location in the image, with two feature components (left). The center and right figures show the samples projected across space in the 2D feature space. Due to the curvature of the surface, global pooling (center) loses most of the information contained in the descriptors; the red cross (average pooling of the samples) is far away from the lower-dimensional surface on which the samples lie. Clustering the samples and performing pooling inside each cluster preserves information since the surface is locally flat (right).

ing and pooling in opposite directions: they make coding smoother (neighbors are used to regularize coding so that noise is harder to represent) and pooling more restrictive (only neighbors are used so that the signal does not get averaged out). This can be viewed as an attempt to distribute smoothing more evenly between coding and pooling.

Authors of locality-preserving methods have often attributed their good results to the fact that the encoding uses only dictionary atoms that resemble the input [39, 43], or viewed them as a trick to learn huge specialized dictionaries, whose computational cost would be prohibitive with standard sparse coding [42]. However, this view may underestimate how local coding also makes pooling more local, by tying activation of a component more strongly to a region of the multidimensional input configuration space. We argue that more local pooling may be one factor in the success of methods that incorporate locality constraints into the training criterion of the codebook for sparse coding [11, 39, 43], or directly cluster the input data to learn one local dictionary per cluster [39, 42]. The question we attempt to answer in this paper is whether it is possible to leverage locality in the descriptor space once the descriptors have already been encoded. We argue that if the coding step has not been designed in such a way that the pooling operation preserves as much information as possible about the distribution of features, then *the pooling step itself should become more selective*.

The contributions of this work are threefold. First, we show how several recent feature extracting methods can be viewed in a unified perspective as preventing pooling from losing too much relevant information. Second, we demonstrate empirically that restricting pools to codes that are nearby not only in (2D) image space but also in descriptor space, boosts the performance even with relatively small dictionaries, yielding state-of-the-art performance or better on several benchmarks, without resorting to more complicated and expensive coding methods, or having to learn new dictionaries. Third, we propose some promising extensions.

The paper is organized as follows. Sec. 2 introduces the general classification pipeline that is used in this paper and much previous work and motivates our approach. Sec. 3 presents related work. Experiments are presented in Sec. 4.

2. General image recognition architecture and proposed approach

As argued in our previous work [4], many modern methods for image classification, such as convolutional and deep belief networks [17, 20, 26, 32], bags of features [34], histograms of gradients for pedestrian detection [9], or the spatial pyramid [25], implement an alternating series of coding and spatial pooling steps. Classification is then performed with some standard classifier such as a kernel or linear support vector machine (SVM), or logistic regression.

2.1. Feature extraction

Our setting for feature extraction is the following. Let I denote an input image. First, low-level descriptors $\boldsymbol{x}_i$ (e.g., SIFT or HOG) are extracted densely at N locations identified with their indices $i = 1, \ldots, N$. Coding is performed at each location by applying some operator that is chosen to ensure that the resulting codes $\boldsymbol{\alpha}_i$ retain useful information (e.g., input data can be predicted from them), while having some desirable properties (e.g., compactness). Here, we focus on hard vector quantization and sparse coding, that both minimize some regularized error between inputs and the reconstructions that can be obtained from the codes.

Hard vector quantization is the simplest coding step, used in the bag-of-features framework [34]. It models the data with K clusters, representing each x_i by a one-of-K encoding of its cluster assignment:

$$\boldsymbol{\alpha}_i \in \{0,1\}^K, \boldsymbol{\alpha}_{i,j} = 1 \text{ iff } j = \underset{k \leq K}{\operatorname{argmin}} \|\boldsymbol{x}_i - \mathbf{d}_k\|_2^2, \quad (1)$$

where $\mathbf{d}_k$ denotes the k-th codeword of a codebook that is usually learned by an unsupervised algorithm such as K-means. The extreme sparseness of the codes produced (only one component per code is non-zero) may be ill-suited to images, and in fact better results have been obtained by replacing hard vector quantization by a soft probabilistic version [36], or by sparse coding [4, 41].

Sparse coding [31] reconstructs the input as a linear combination of few codewords:

$$\boldsymbol{\alpha}_i = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\, L_i(\boldsymbol{\alpha}, \mathbf{D}) \triangleq \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1, \quad (2)$$

where $\|\boldsymbol{\alpha}\|_1$ denotes the ℓ_1 norm of $\boldsymbol{\alpha}$, λ is a parameter that controls the sparsity of $\boldsymbol{\alpha}$, and $\mathbf{D}$ is some dictionary, which can be obtained by K-means, or for better performance, trained by minimizing the average of $L_i(\boldsymbol{\alpha}_i, \mathbf{D})$ over all samples, alternatively over $\mathbf{D}$ and the $\boldsymbol{\alpha}_i$. It is well known that the ℓ_1 penalty induces sparsity and makes the problem tractable (e.g., [27, 29]).

A pooling operator then takes the varying number of codes that are located within M possibly overlapping regions of interest (e.g., the cells of a spatial pyramid), and summarizes them as a single vector of fixed length. The representation for the global image is obtained by concatenating the representations of each region of interest, possibly with a suitable weight. We denote by $\mathcal{Y}_m$ the set of locations/indices within region m. Here, we use the two common pooling strategies of average and max pooling.

Average pooling computes a histogram or take the average of the codes over the region (these two methods are equivalent after normalization):

$$\boldsymbol{h}_m = \frac{1}{|\mathcal{Y}_m|} \sum_{i \in \mathcal{Y}_m} \boldsymbol{\alpha}_{\boldsymbol{i}}, \quad (3)$$

where $\boldsymbol{h}_m$ is the vector representing region m.

"Max pooling" computes the maximum of each component instead of its average. It has recently gained popularity due to its better performance when paired with sparse coding and simple linear classifiers [4, 38, 41], and its statistical properties which make it well suited to sparse representations [5]. In our notation, max pooling is written:

$$\boldsymbol{h}_{m,j} = \max_{i \in \mathcal{Y}_m} \boldsymbol{\alpha}_{i,j}, \text{ for } j = 1, \ldots, K. \quad (4)$$

Max pooling is used for all experiments in this work, except those in Sec. 4.2.3.

2.2. Pooling more locally across the input space

We propose to streamline the approach in [42], which requires learning one different dictionary per cluster, and show that simply making the pooling step more selective can substantially enhance the performance of small dictionaries, and beat the state of the art on some object recognition benchmarks when large dictionaries are used, without requiring additional learning beyond obtaining an additional clustering codebook with K-means. Comparing the performance of our system with that obtained with individual dictionaries allows us to quantify the relative contributions of more selective pooling and more specialized, overcomplete dictionaries.

To clarify how our local pooling scheme differs from the usual local spatial pooling, an image feature can be viewed as a couple $z = (\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y} \in \mathbb{R}^2$ denotes a pixel location, and $\boldsymbol{x} \in \mathbb{R}^d$ is a vector, or configuration, encoding the local image structure at $\boldsymbol{y}$ (e.g., a SIFT descriptor, with $d = 128$). A feature set $\mathcal{Z}$ is associated with each image, its size potentially varying from one picture to the next.

Spatial pooling considers a fixed – that is, predetermined and image-independent – set of M possibly overlapping image regions (spatial bins) $\mathcal{Y}_1$ to $\mathcal{Y}_M$. To these, we add a fixed set of P (multi-dimensional) bins $\mathcal{X}_1$ to $\mathcal{X}_P$ in the configuration space. In this work, the spatial bins are the cells in a spatial pyramid, and the configuration space bins are the Voronoi cells of clusters obtained using K-means.

Denoting by g the pooling operator (average or max in the previous section), the pooled feature is obtained as:

$$\boldsymbol{h}_{(m,p)} = g_{(i \in \mathcal{Y}_m, j \in \mathcal{X}_p)}(\boldsymbol{\alpha}_{(i,j)}). \quad (5)$$

Bags of features can be viewed as a special case of this in two ways: either by considering the 1-of-K encoding presented above, followed by global pooling in the configuration space ($P = 1$), or with a simplistic encoding that maps all inputs to 1, but does fine configuration space binning ($P = K$). Accordingly, the feature extraction in this paper can be viewed either as extending the sparse coding spatial pyramid by making configuration space pooling local, or as extending the hard-vector-quantized spatial pyramid by replacing the simplistic code by sparse coding: descriptors are first decomposed by sparse coding over a dictionary of size K; the same descriptors are also clustered over a K-means dictionary of size P; finally, pooling of the sparse codes is then performed separately for each cluster (as in aggregated coding [21] – see Sec. 3 –, but with sparse codes), yielding a feature of size $K \times P \times S$ if there are S spatial bins.

While this does not apply to max pooling, local pooling can be viewed as implementing local bilinear classification when using average pooling and linear classification: the pooling operator and the classifier may be swapped, and classification of local features then involves computing $\boldsymbol{\beta}_{xy}^T W \boldsymbol{\alpha}$, where $\boldsymbol{\beta}_{xy}$ is a $(S \times P)$- dimensional binary vector that selects a subset of classifiers corresponding to the configuration space and spatial bins, and W is a $(S \times P) \times K$ matrix containing one K-dimensional local classifier per row.

3. Related work about locality in feature space

We start our review of previous work with a caveat about word choice. There exists an unfortunate divergence in the vocabulary used by different communities when it comes to naming methods leveraging neighborhood relationships in feature space: what is called ”non-local” in work in the vein of signal processing [6, 30] bears a close relationship to ”local” fitting and density estimation [11, 33, 39, 43]. Thus, non-local means [6] and locally-linear embedding [33] actually perform the same type of initial grouping of input data by minimal Euclidean distance. This discrepancy stems from the implicit understanding of ”local” as either ”spatially local”, or ”local in translation-invariant configuration space”.

3.1. Preserving neighborhood relationships during coding

Previous work has shown the effectiveness of preserving configuration space locality during coding, so that similar inputs lead to similar codes. This can be done by explicitly penalizing codes that differ for neighbors. The DrLIM system of siamese networks in [16], and neighborhood component analysis [13], learn a mapping that varies smoothly with some property of the input by minimizing a cost which encourages similar inputs to have similar codes (similarity can be defined arbitrarily, as locality in input space, or sharing the same illumination, orientation, etc.) Exploiting image self-similarities has also been used successfully for denoising [6, 8, 30].

Locality constraints imposed on the coding step have been adapted to classification tasks with good results. Laplacian sparse coding [11] uses a modified sparse coding step in the spatial pyramid framework. A similarity matrix of input SIFT descriptors is obtained by computing their intersection kernel, and used in an added term to the sparse coding cost. The penalty to pay for the discrepancy between a pair of codes is proportional to the similarity of the corresponding inputs. This method obtains state-of-the-art results on several object recognition benchmarks. Locality-constrained linear coding [39] (LLC) projects each descriptor on the space formed by its k nearest neighbors (k is small, e.g., $k = 5$). This procedure corresponds to performing the first two steps of the locally linear embedding algorithm [33] (LLE), except that the neighbors are selected among the atoms of a dictionary rather than actual descriptors, and the weights are used as features instead of being mere tools to learn an embedding.

Sparse coding methods incorporating a locality constraint share the property of indirectly limiting activation of a given component of the vectors representing descriptors to a certain region of the configuration space. This may play a role in their good performance. For example, in LLC coding, the component corresponding to a given dictionary atom will be non-zero only if that atom is one of the k nearest neighbors of the descriptor being encoded; the non-zero values aggregated during pooling then only come from these similar descriptors. Several approaches have implemented this strategy directly during the pooling step, and are presented in the next section.

3.2. Letting only neighbors vote during pooling

Pooling involves extracting an ensemble statistic from a potentially large group of inputs. However, pooling too drastically can damage performance, as shown in the spatial domain by the better performance of spatial pyramid pooling [25] compared to whole-image pooling.

Different groups have converged to a procedure involving preclustering of the input to create independent bins over which to pool the data. In fact, dividing the feature space into bins to compute correspondences has been proposed earlier by the pyramid match kernel approach [14]. However, newer work does not tile the feature space evenly, relying instead on unsupervised clustering techniques to adaptively produce the bins.

The methods described here all perform an initial (hard or soft) clustering to partition the training data according to appearance, as in the usual bag-of-words framework, but then assigning a vector to each cluster instead of a scalar. The representation is then a “super-vector” that concatenates these vectors instead of being a vector that concatenates scalars.

Aggregated coding [21] and super-vector coding [45] both compute, for each cluster, the average difference between the inputs in the cluster, and its centroid: (1) SIFT descriptors $\mathbf{x}_i$ are extracted at regions of interest, (2) visual words $\mathbf{c}_k$ are learned over the whole data by K-means, (3) descriptors of each image are clustered, (4) for each cluster C_k, the sum $\sum_{\mathbf{x}\in C_k}(\mathbf{x} - \mathbf{c}_k)$ is computed, (5) the image descriptor is obtained by concatenating the representations for each cluster.

If the centroids were computed using only the descriptors in a query image, the representation would be all zeros, because the centroids in K-means are also obtained by averaging the descriptors in each cluster. Instead, the centroids are computed using descriptors from the whole data, implicitly representing a “baseline image” against which each query image is compared. Thus, encoding relatively to the cluster centroid removes potentially complex but non-discriminative information. This representation performs very well on retrieval [21] and image classification [45] (Pascal VOC2009) benchmarks.

Another related method [42] that obtains high accuracy on the Pascal datasets combines the preclustering step of aggregated and super-vector coding, with sparse decomposition over individual local dictionaries learned inside each cluster. Both approaches using preclustering for image classification [42, 45] have only reported results using gigantic global descriptors for each image. Indeed, the high results obtained in [42] are attributed to the possibility of learning a very large overcomplete dictionary (more than 250,000 atoms) which would be computationally infeasible without preclustering, but can be done by assembling a thousand or more smaller local dictionaries. The experiments presented in the next section seek to isolate the effect of local pooling that is inherent in this scheme.

4. Experiments

We perform experiments on three image recognition datasets: 15-Scenes [25], Caltech-101 [10] and Caltech-256 [15]. All features are extracted from grayscale images. Large images are resized to fit inside a 300×300 box. SIFT descriptors are extracted densely over the image, and encoded into sparse vectors using the SPAMS toolbox [1]. We adopt the denser 2×2 macrofeatures of our previous work [4], extracted every 4 pixels, for the Caltech-256 and Caltech-101 databases, and every 8 pixels for the Scenes, except for some experiments on Caltech-256 where standard features extracted every 8 pixels are used for faster processing. The sparse codes are pooled inside the cells of a three-level pyramid (4×4, 2×2 and 1×1 grids); max pooling is used for all experiments except those in Sec. 4.2.3, which compare it to other pooling schemes. We apply an $\ell_{1.5}$ normalization to each vector, since it has shown slightly better performance than no normalization in our experiments (by contrast, normalizing by ℓ_1 or ℓ_2 norms worsens performance). One-versus-all classification is performed by training one linear SVM for each class using LIBSVM [7], and then taking the highest score to assign a label to the input. When local pooling in the configuration space is used ($P \geq 1$), clustering is performed using the K-means algorithm to obtain cluster centers. Following the usual practice [15, 25, 39], we use 30 training images on the Caltech-101 and Caltech-256 datasets, 100 training images on the Scenes dataset; the remaining images are used for testing, with a maximum of 50 and 20 test images for Caltech-101 and Caltech-256, respectively. Experiments are run ten times on ten random splits of training and testing data, and the reported result is the mean accuracy and standard deviation of these runs. Hyperparameters of the model (such as the regularization parameter of the SVM or the λ parameter of sparse coding) are selected by cross-validation within the training set. Patterns of results are very similar for all three datasets, so results are shown only on Caltech-101 for some of the experiments; more complete numerical results on all three datasets can be found in the supplemental material*.

4.1. Pooling locally in configuration space yields state-of-the-art performance

Experiments presented in Table 1 and 2 compare the performance of sparse coding with a variety of configuration space pooling schemes, with a list of published results of

	Caltech 30 tr.	Scenes
Boiman et al. [3]	70.4	-
Boureau et al. [4]	$\mathbf{75.7 \pm 1.1}$	85.6 ± 0.2
Gao et al. [11]	–	$\mathbf{89.8 \pm 0.5}$
Jain et al. [19]	69.6	-
Lazebnik et al. [25]	64.4 ± 0.8	81.4 ± 0.5
van Gemert et al. [36]	64.1 ± 1.2	76.7 ± 0.4
Wang et al. [39]	73.44	–
Yang et al. [41]	73.2 ± 0.5	80.3 ± 0.9
Zhang et al. [44]	66.2 ± 0.5	-
Zhou et al. [46]	–	84.1 ± 0.5
$K = 256$, Pre, $P = 1$	70.5 ± 0.8	78.8 ± 0.6
$P = 16$	74.0 ± 1.0	81.5 ± 0.8
$P = 64$	75.0 ± 0.8	81.1 ± 0.5
$P = 128$	75.5 ± 0.8	81.0 ± 0.3
$P = 1 + 16$	74.2 ± 1.1	81.5 ± 0.8
$P = 1 + 64$	75.6 ± 0.6	81.9 ± 0.7
$K = 256$, Post, $P = 16$	75.1 ± 0.8	80.9 ± 0.6
$P = 64$	76.4 ± 0.8	81.1 ± 0.6
$P = 128$	$\mathbf{76.7 \pm 0.8}$	81.1 ± 0.5
$K = 1024$, Pre, $P = 1$	75.6 ± 0.9	82.7 ± 0.7
$P = 16$	76.3 ± 1.1	82.7 ± 0.9
$P = 64$	76.2 ± 0.8	81.4 ± 0.7
$P = 1 + 16$	$\mathbf{76.9 \pm 1.0}$	83.3 ± 1.0
$P = 1 + 64$	$\mathbf{77.3 \pm 0.6}$	83.1 ± 0.7
$K = 1024$, Post, $P = 16$	$\mathbf{77.0 \pm 0.8}$	82.9 ± 0.6
$P = 64$	$\mathbf{77.1 \pm 0.7}$	82.4 ± 0.7

Table 1. Results on Caltech-101 (30 training samples per class) and 15-scenes for various methods. Results for our method are given as a function of whether clustering is performed before (Pre) or after (Post) the encoding, K: dictionary size, and P: number of configuration space bins.

methods using grayscale images and a single type of descriptor. Local pooling always improves results, except on the Scenes for a dictionary of size $K = 1024$. On the Caltech-256 benchmark, our performance of 41.7% accuracy with 30 training examples is similar to the best reported result of 41.2% that we are aware of (for methods using a single type of descriptors over grayscale), obtained by locality-constrained linear codes [39], using three scales of SIFT descriptors and a dictionary of size $K = 4096$.

4.1.1 Using pyramids in configuration space

We examine whether it is advantageous to combine fine and coarse clustering, in a way reminiscent of the levels of the spatial pyramid. With large dictionaries, local pooling in the configuration space does not always perform better than standard global pooling (see Tables 1 and 2). However, combining levels of different coarseness gives performance better than or similar to that of the best individual level, as has been observed with the spatial pyramid [25].

This significantly improves performance on the Caltech-101 dataset. To the best of our knowledge, our performance of 77.3% on the Caltech-101 benchmark, is above all previously published results for a single descriptor type

*http://cs.nyu.edu/~ylan/files/publi/boureau-iccv-11-supplemental.pdf

	Accuracy
Boiman et al. [3]	37.0
Gao et al. [11] ($K = 1024$)	35.7 ± 0.1
Kim et al. [23]	36.3
van Gemert et al. [36] ($K = 128$)	27.2 ± 0.5
Wang et al. [39] ($K = 4096$)	**41.2**
Yang et al. [41] ($K = 1024$)	34.0 ± 0.4
$K = 256$, Pre, $P = 1$	32.3 ± 0.8
$P = 16$	38.0 ± 0.5
$P = 64$	39.2 ± 0.5
$P = 128$	39.7 ± 0.6
$K = 256$, Post, $P = 16$	36.9 ± 0.7
$P = 64$	39.6 ± 0.5
$P = 128$	40.3 ± 0.6
$K = 1024$, Pre, $P = 1$	38.1 ± 0.6
$P = 16$	$\mathbf{41.6 \pm 0.6}$
$P = 64$	$\mathbf{41.7 \pm 0.8}$
$K = 1024$, Post, $P = 16$	40.4 ± 0.6

Table 2. Recognition accuracy on Caltech 256, 30 training examples, for several methods using a single descriptor over grayscale. For our method, results are shown as a function of whether clustering is performed before (Pre) or after (Post) the encoding, K: dictionary size, and P: number of configuration space bins.

using grayscale images – although better performance has been reported with color images (e.g., $78.5\% \pm 0.4$ with a saliency-based approach [22]), multiple descriptor types (e.g., methods using multiple kernel learning have achieved $77.7\% \pm 0.3$ [12], $78.0\% \pm 0.3$ [2, 37] and 84.3% [40] on Caltech-101 with 30 training examples), or subcategory learning (83% on Caltech-101 [35]). On the Scenes benchmark, preclustering does improve results for small dictionaries ($K \leq 256$, see supplemental material), but not for larger ones ($K = 1024$). While our method outperforms the Laplacian sparse coding approach [11] on the Caltech 256 dataset, our performance is much below that of Laplacian sparse coding on the Scenes database.

4.1.2 Pre- vs. Post-Clustering

One advantage of using the same dictionary for all features is that the clustering can be performed after the encoding. The instability of sparse coding could cause features similar in descriptor space to be mapped to dissimilar codes, which would then be pooled together. This does not happen if clustering is performed on the codes themselves. While pre-clustering may perform better for few clusters, post-clustering yields better results when enough clusters are used ($P \geq 64$); a dictionary of size $K = 1024$ reaches 77.1 ± 0.7 accuracy on Caltech-101 with $P = 64$ bins (to be compared to 76.2 ± 0.8 when clustering before coding), while a dictionary of size $K = 256$ yields 76.7 ± 0.8 with $P = 128$ bins (to be compared to 75.5 ± 0.8 with preclustering). Fig. 2 also shows that performance drops for larger P, irrespective of whether the clustering is performed before or after the encoding.

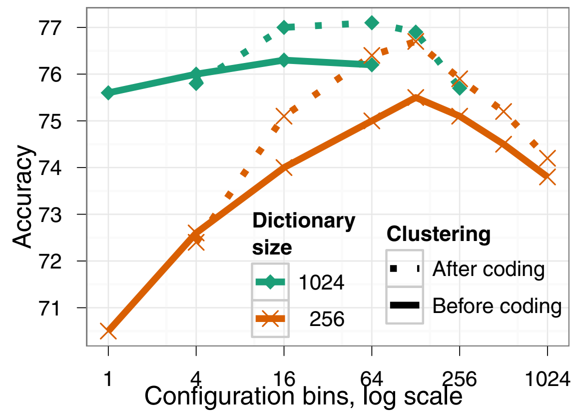


Figure 2. Recognition accuracy on Caltech-101, for clustering before or after encoding. Clustering after the encoding generally performs better; for both schemes, binning too finely in configuration space (large P) hurts performance. Best viewed in color.

4.2. Gaining a finer understanding of local configuration space pooling

In this section, we investigate how much local configuration space pooling can enhance the performance of small dictionaries, how it compares to learning one local dictionary per configuration bins, and what pooling and weighting schemes work best in our pipeline.

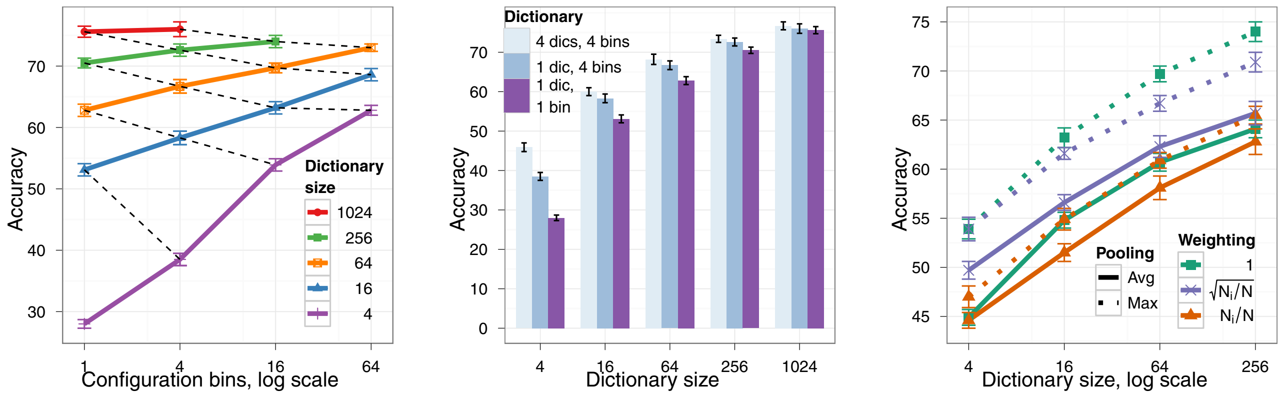
4.2.1 Local pooling boosts small dictionaries

Fig. 3(a) shows results for various assignments of components between atoms (K) and centroids (P). Pooling more locally in configuration space ($P > 1$) can considerably boost the performance of small dictionaries.

Unsurprisingly, larger dictionaries consistently beat smaller ones combined with preclustering-driven pooling, at same total number of components; this can be seen from the downwards slope of the gray dashed lines in Fig. 3(a) linking data points at constant $K * P$. However, if P is allowed to grow more, small dictionaries can outperform larger ones. This leads to good performance with a small dictionary; e.g., a dictionary of just $K = 64$ atoms coupled with a preclustering along $P = 64$ centroids achieves $73.0 \pm 0.6\%$ on Caltech-101.

4.2.2 Comparison with cluster-specific dictionaries

In addition to learning richer, more local dictionaries, learning one dictionary per cluster as done in [39, 42] inherently leads to more local pooling. Experiments in this section seek to disentangle these effects. As shown in Fig. 3(b), more than half of the improvement compared to no preclustering is usually due to the separate clustering rather than more specific dictionaries. The smaller the dictionary, the



(a) (b) (c)

Figure 3. Recognition accuracy on Caltech-101. Left: pooling locally in finer configuration space bins can boost the performance of small dictionaries. Dotted gray lines indicate constant product of dictionary size $\times$ number of configuration bins. Middle: a substantial part of the improvement observed when using multiple local dictionaries can be achieved without changing the encoding, by pooling locally in configuration space. $P = 4$ configuration space bins are used. Right: the best performance is obtained with max pooling and uniform weighting. Max pooling consistently outperforms average pooling for all weighting schemes. With average pooling, weighting by the square root of the cluster weight performs best. $P = 16$ configuration space bins are used. Results on the Caltech-256 and Scenes datasets show similar patterns (see supplemental material). Best viewed in color.

larger the proportion of the improvement due to clustering. This may be due to the fact that smaller dictionaries do not have enough atoms to implicitly link activation of an atom to cluster membership during coding, leaving more of that task to the explicit local configuration space pooling than when large dictionaries are used.

4.2.3 Pooling operator and cluster weighting

When concatenating the vectors corresponding to each pool, it is not clear whether they should be weighted according to the prominence of the cluster, measured as the ratio N_i/N of the number N_i of inputs falling into cluster i, over the total number N of inputs. Denoting by w_i the weight for cluster i, we compare three weighting schemes: identical weight ($w_i = 1$), a weight proportional to the square root of the ratio ($w_i = \sqrt{N_i/N}$) as proposed by Zhou et al. [45], or the ratio itself ($w_i = N_i/N$).

As shown in Fig. 3(c), the weighting scheme assigning the same weight to each cluster performs better when max pooling is used, except for very small dictionaries. When average pooling is used, the best weighting scheme is the square root weighting, which empirically validates the choice in [45], but performance is below that of max pooling. Based on these results, max pooling with identical weighting for all clusters has been used for all other experiments in the paper.

5. Conclusion

While there is no question that making coding more stable and more specific is advantageous, the simple procedure of clustering the data in order to make pooling local in configuration space is a powerful tool for image recognition. The main conclusions of this work are that (1) more local configuration space pooling in itself boosts performance, dramatically so with smaller dictionaries; (2) it is advantageous to use pyramids rather than grids, analogously to spatial pooling; (3) with enough configuration space bins, better performance may be obtained when the clustering is performed just before the pooling step, rather than before the coding step; (4) performance drops if too many bins are added.

Our ongoing efforts are focused on adapting the same ideas to pooling across features when there is a topography on the feature extractors, instead of simply pooling each component separately. We hope that considerating locality in all dimensions simultaneously (feature space, between feature extractors, and space itself) will lead to even more robust invariant recognition.

Acknowledgments. This work was supported by ONR contract N00014-09-1-0473 and NSF grant EFRI/COPN-0835878 to NYU, and by the European Research Council (VideoWorld and Sierra grants).

References

[1] http://www.di.ens.fr/willow/SPAMS/. 5

[2] http://www.robots.ox.ac.uk/~vgg/software/MKL/. 6

[3] O. Boiman, I. Rehovot, E. Shechtman, and M. Irani. In Defense of Nearest-Neighbor Based Image Classification. In *CVPR*, 2008. 5, 6

[4] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010. 1, 2, 3, 5

[5] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in vision algorithms. In *ICML*, 2010. 3

[6] A. Buades, B. Coll, and J. Morel. A non-local algorithm for image denoising. In *CVPR*, 2005. 1, 4

[7] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 5

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising with block-matching and 3D filtering. In *Proc. SPIE Electronic Imaging: Algorithms and Systems V*, volume 6064, San Jose, CA, USA, January 2006. 1, 4

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 2

[10] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples. In *CVPR Workshop GMBV*, 2004. 5

[11] S. Gao, I. Tsang, L. Chia, and P. Zhao. Local features are not lonely–Laplacian sparse coding for image classification. In *CVPR*, 2010. 1, 2, 4, 5, 6

[12] P. Gehler and S. Nowozin. On Feature Combination for Multiclass Object Classification. In *ICCV*, 2009. 6

[13] J. Goldberger, S. T. Roweis, G. E. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *NIPS*, 2004. 4

[14] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *ICCV*, 2005. 4

[15] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. 5

[16] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 4

[17] G. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 2006. 2

[18] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol*, 160:106–154, Jan 1962. 1

[19] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *CVPR*, 2008. 5

[20] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009. 2

[21] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 1, 3, 4

[22] C. Kanan and G. Cottrell. Robust classification of objects, faces, and flowers using natural image statistics. In *CVPR*, 2010. 6

[23] J. Kim and K. Grauman. Asymmetric Region-to-Image Matching for Comparing Images with Generic Object Categories. In *CVPR*, 2010. 6

[24] J. Koenderink and A. Van Doorn. The structure of locally orderless images. *IJCV*, 31(2/3):159–168, 1999. 1

[25] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 1, 2, 4, 5

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. 1, 2

[27] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006. 3

[28] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(4):91–110, 2004. 1

[29] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online Dictionary Learning for Sparse Coding. In *ICML*, 2009. 3

[30] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *CVPR 2009*. 1, 4

[31] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37:3311–3325, 1997. 3

[32] M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *NIPS 2007*, 2007. 2

[33] L. Saul and S. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *JMLR*, 4:119–155, 2003. 4

[34] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 2

[35] S. Todorovic and N. Ahuja. Learning subcategory relevances for category recognition. In *CVPR*, 2008. 6

[36] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. *PAMI*, 32(7):1271–1283, 2010. 1, 3, 5, 6

[37] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009. 6

[38] D. Wang, S. Hoi, and Y. He. An Effective Approach to Pose Invariant 3D Face Recognition. *Advances in Multimedia Modeling*, pages 217–228, 2011. 3

[39] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 1, 2, 4, 5, 6

[40] J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao. Group-sensitive multiple kernel learning for object categorization. In *ICCV*, 2009. 6

[41] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. In *CVPR*, 2009. 1, 3, 5, 6

[42] J. Yang, K. Yu, and T. Huang. Efficient Highly Over-Complete Sparse Coding using a Mixture Model. *ECCV*, 2010. 1, 2, 3, 4, 6

[43] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. *NIPS*, 2009. 1, 2, 4

[44] H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006. 5

[45] X. Zhou, K. Yu, T. Zhang, and T. Huang. Image Classification using Super-Vector Coding of Local Image Descriptors. *ECCV*, 2010. 1, 4, 7

[46] X. Zhou, X. D. Zhuang, H. Tang, M. H. Johnson, and T. S. Huang. A novel gaussianized vector representation for natural scene categorization. In *ICPR*, 2008. 5