Learning Mid-Level Features For Recognition

Y-Lan Boureau[1,3,4] Francis Bach[1,4] Yann LeCun[3] Jean Ponce[2,4]
[1]INRIA [2]Ecole Normale Supérieure [3]Courant Institute, New York University

Abstract

Many successful models for scene or object recognition transform low-level descriptors (such as Gabor filter responses, or SIFT descriptors) into richer representations of intermediate complexity. This process can often be broken down into two steps: (1) a coding step, which performs a pointwise transformation of the descriptors into a representation better adapted to the task, and (2) a pooling step, which summarizes the coded features over larger neighborhoods. Several combinations of coding and pooling schemes have been proposed in the literature. The goal of this paper is threefold. We seek to establish the relative importance of each step of mid-level feature extraction through a comprehensive cross evaluation of several types of coding modules (hard and soft vector quantization, sparse coding) and pooling schemes (by taking the average, or the maximum), which obtains state-of-the-art performance or better on several recognition benchmarks. We show how to improve the best performing coding scheme by learning a supervised discriminative dictionary for sparse coding. We provide theoretical and empirical insight into the remarkable performance of max pooling. By teasing apart components shared by modern mid-level feature extractors, our approach aims to facilitate the design of better recognition architectures.

1. Introduction

Finding good image features is critical in modern approaches to category-level image classification. Many methods first extract low-level descriptors (e.g., SIFT [18] or HOG descriptors [5]) at interest point locations, or nodes in a dense grid. This paper considers the problem of combining these local features into a global image representation suited to recognition using a common classifier such as a support vector machine. Since global features built upon low-level ones typically remain close to image-level information without attempts at high-level, structured image description (in terms of parts for example), we will refer to them as *mid-level* features.

Popular examples of mid-level features include bags of features [25], spatial pyramids [12], and the upper units of convolutional networks [13] or deep belief networks [8, 23]. Extracting these mid-level features involves a sequence of interchangeable modules similar to that identified by Winder and Brown for local image descriptors [29]. In this paper, we focus on two types of modules:

- **Coding:** Input features are locally transformed into representations that have some desirable properties such as compactness, sparseness (i.e., most components are 0), or statistical independence. The code is typically a vector with binary (vector quantization) or continuous (HOG, sparse coding) entries, obtained by decomposing the original feature on some codebook, or dictionary.

- **Spatial pooling:** The codes associated with local image features are pooled over some image neighborhood (e.g., the whole image for bags of features, a coarse grid of cells for the HOG approach to pedestrian detection, or a coarse hierarchy of cells for spatial pyramids). The codes within each cell are summarized by a single "semi-local" feature vector, common examples being the average of the codes (*average pooling*) or their maximum (*max pooling*).

The same coding and pooling modules can be plugged into various architectures. For example, average pooling is found in convolutional nets [13], bag-of-features methods, and HOG descriptors; max pooling is found in convolutional nets [16, 23], HMAX nets [24], and state-of-the-art variants of the spatial pyramid model [31]. The final global vector is formed by concatenating with suitable weights the semi-local vectors obtained for each pooling region.

High levels of performance have been reported for specific pairings of coding and pooling modules (e.g., sparse coding and max pooling [31]), but it is not always clear whether the improvement can be factored into independent contributions of each module (e.g., whether the better performance of max pooling would generalize to systems using vector quantization instead of sparse coding). In this

[4]WILLOW project-team, Laboratoire d'Informatique de l'Ecole Normale Supérieure, ENS/INRIA/CNRS UMR 8548.

work, we address this concern by presenting a comprehensive set of product pairings across known coding (hard and soft vector quantization, sparse coding) and pooling (average and max pooling) modules. We have chosen to restrict ourselves to the spatial pyramid framework since it has already been used in several comparative studies [27, 31], defining the state of the art on several benchmarks; but the insights gained within that framework should easily generalize to other models (e.g., models using interest point detectors, convolutional networks, deep belief networks). Two striking results of our evaluation are that (1) sparse coding systematically outperforms the other coding modules, irrespective of the pooling module, and (2) max pooling dramatically improves linear classification performance irrespective of the coding module, to the point that the worst-performing coding module (hard vector quantization) paired with max pooling outperforms the best coding module (sparse coding) paired with average pooling. The rest of our paper builds on these two findings. Noting that the dictionary used to perform sparse coding is trained to minimize reconstruction error, which might be suboptimal for classification, we propose a new supervised dictionary learning algorithm. As for the superiority of max pooling in linear classification, we complement the empirical finding by a theoretical analysis and new experiments. Our article thus makes three contributions:

- We systematically explore combinations of known modules appearing in the unified model presented in this paper, obtaining state-of-the-art results on two benchmarks (Sec. 3).
- We introduce a novel supervised sparse dictionary learning algorithm (Sec. 4).
- We present theoretical and experimental insights into the much better linear discrimination performance obtained with max pooling compared to average pooling, in a large variety of settings (Sec. 5).

2. Notation and Related Work

In this section, we introduce some notation used throughout this paper, and present coding and pooling modules previously used by other authors. Let an image I be represented by a set of low-level descriptors (e.g., SIFT) $\boldsymbol{x}_i$ at N locations identified with their indices $i = 1, \cdots, N$. M regions of interests are defined on the image (e.g., the $21 = 16 + 4 + 1$ cells of a three-level spatial pyramid), with $\mathcal{N}_m$ denoting the set of locations/indices within region m. Let f and g denote some coding and pooling operators, respectively. The vector $\boldsymbol{z}$ representing the whole image is obtained by sequentially coding, pooling over all regions, and concatenating:

$$\boldsymbol{\alpha}_i = f(\boldsymbol{x}_i),\ i = 1, \cdots, N \tag{1}$$

$$\boldsymbol{h}_m = g\left(\{\boldsymbol{\alpha}_i\}_{i \in \mathcal{N}_m}\right),\ m = 1, \cdots, M \tag{2}$$

$$\boldsymbol{z}^T = [\boldsymbol{h}_1^T \cdots \boldsymbol{h}_M^T]. \tag{3}$$

The goal is to determine which operators f and g provide the best classification performance using $\boldsymbol{z}$ as input to either a non-linear intersection kernel SVM [12], or a linear SVM.

In the usual bag-of-features framework [25], f minimizes the distance to a codebook, usually learned by an unsupervised algorithm (e.g., K-means), and g computes the average over the pooling region:

$$\boldsymbol{\alpha}_i \in \{0,1\}^K, \boldsymbol{\alpha}_{i,j} = 1 \text{ iff } j = \underset{k \leq K}{\operatorname{argmin}} \|\boldsymbol{x}_i - \mathbf{d}_k\|_2^2, \tag{4}$$

$$\boldsymbol{h}_m = \frac{1}{|\mathcal{N}_m|} \sum_{i \in \mathcal{N}_m} \boldsymbol{\alpha}_{\boldsymbol{i}}, \tag{5}$$

where $\mathbf{d}_k$ denotes the k-th codeword. Note that averaging and using uniform weighting is equivalent (up to a constant multiplicator) to using histograms with weights inversely proportional to the area of the pooling regions, as in [12].

Van Gemert et al. [27] have obtained improvements by replacing hard quantization by soft quantization:

$$\boldsymbol{\alpha}_{i,j} = \frac{\exp\left(-\beta\|\boldsymbol{x}_i - \mathbf{d}_j\|_2^2\right)}{\sum_{k=1}^K \exp\left(-\beta\|\boldsymbol{x}_i - \mathbf{d}_k\|_2^2\right)}, \tag{6}$$

where β is a parameter that controls the softness of the soft assignment (hard assignment is the limit when $\beta \rightarrow \infty$). This amounts to coding as in the E-step of the expectation-maximization algorithm to learn a Gaussian mixture model, using codewords of the dictionary as centers.

Sparse coding [22] uses a linear combination of a small number of codewords to approximate the $\boldsymbol{x}_i$. Yang et al. [31] have obtained state-of-the-art results by using sparse coding and max pooling:

$$\boldsymbol{\alpha}_i = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}}\, L(\boldsymbol{\alpha}, \mathbf{D}) \triangleq \|\boldsymbol{x}_i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1, \tag{7}$$

$$\boldsymbol{h}_{m,j} = \max_{i \in \mathcal{N}_m} \boldsymbol{\alpha}_{i,j},\ \text{for } j = 1, \cdots, K, \tag{8}$$

where $\|\boldsymbol{\alpha}\|_1$ denotes the ℓ_1 norm of $\boldsymbol{\alpha}$, λ is a parameter that controls the sparsity, and $\mathbf{D}$ is a dictionary trained by minimizing the average of $L(\boldsymbol{\alpha}_i, \mathbf{D})$ 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., [15, 19]).

3. Systematic Evaluation of Unsupervised Mid-Level Features

This section offers comprehensive comparisons of unsupervised coding schemes. In all experiments, we use the

Method	Caltech-101, 30 training examples		15 Scenes, 100 training examples	
	Average Pool	Max Pool	Average Pool	Max Pool
	Results with basic features, SIFT extracted each 8 pixels			
Hard quantization, linear kernel	51.4 ± 0.9 [256]	64.3 ± 0.9 [256]	73.9 ± 0.9 [1024]	80.1 ± 0.6 [1024]
Hard quantization, intersection kernel	64.2 ± 1.0 [256] **(1)**	64.3 ± 0.9 [256]	80.8 ± 0.4 [256] **(1)**	80.1 ± 0.6 [1024]
Soft quantization, linear kernel	57.9 ± 1.5 [1024]	69.0 ± 0.8 [256]	75.6 ± 0.5 [1024]	81.4 ± 0.6 [1024]
Soft quantization, intersection kernel	66.1 ± 1.2 [512] **(2)**	70.6 ± 1.0 [1024]	81.2 ± 0.4 [1024] **(2)**	83.0 ± 0.7 [1024]
Sparse codes, linear kernel	61.3 ± 1.3 [1024]	$\mathbf{71.5 \pm 1.1}$ [1024] **(3)**	76.9 ± 0.6 [1024]	83.1 ± 0.6 [1024] **(3)**
Sparse codes, intersection kernel	70.3 ± 1.3 [1024]	$\mathbf{71.8 \pm 1.0}$ [1024] **(4)**	83.2 ± 0.4 [1024]	$\mathbf{84.1 \pm 0.5}$ [1024] **(4)**
	Results with macrofeatures and denser SIFT sampling			
Hard quantization, linear kernel	55.6 ± 1.6 [256]	70.9 ± 1.0 [1024]	74.0 ± 0.5 [1024]	80.1 ± 0.5 [1024]
Hard quantization, intersection kernel	68.8 ± 1.4 [512]	70.9 ± 1.0 [1024]	81.0 ± 0.5 [1024]	80.1 ± 0.5 [1024]
Soft quantization, linear kernel	61.6 ± 1.6 [1024]	71.5 ± 1.0 [1024]	76.4 ± 0.7 [1024]	81.5 ± 0.4 [1024]
Soft quantization, intersection kernel	70.1 ± 1.3 [1024]	73.2 ± 1.0 [1024]	81.8 ± 0.4 [1024]	83.0 ± 0.4 [1024]
Sparse codes, linear kernel	65.7 ± 1.4 [1024]	$\mathbf{75.1 \pm 0.9}$ [1024]	78.2 ± 0.7 [1024]	83.6 ± 0.4 [1024]
Sparse codes, intersection kernel	73.7 ± 1.3 [1024]	$\mathbf{75.7 \pm 1.1}$ [1024]	83.5 ± 0.4 [1024]	$\mathbf{84.3 \pm 0.5}$ [1024]

Table 1. Average recognition rate on Caltech-101 and 15-Scenes benchmarks, for various combinations of coding, pooling, and classifier types. The codebook size shown inside brackets is the one that gives the best results among 256, 512 and 1024. Linear and histogram intersection kernels are identical when using hard quantization with max pooling (since taking the minimum or the product is the same for binary vectors), but results have been included for both to preserve the symmetry of the table. Top: Results with the baseline SIFT sampling density of 8 pixels and standard features. Bottom: Results with the set of parameters for SIFT sampling density and macrofeatures giving the best performance for sparse coding.

	Method	Caltech 15 tr.	Caltech 30 tr.	Scenes
Boiman et al. [3]	Nearest neighbor + spatial correspondence	65.0 ± 1.1	70.4	-
Jain et al. [9]	Fast image search for learned metrics	61.0	69.6	-
Lazebnik et al. [12]	**(1)** SP + hard quantization + kernel SVM	56.4	64.4 ± 0.8	81.4 ± 0.5
van Gemert et al. [27]	**(2)** SP + soft quantization + kernel SVM	–	64.1 ± 1.2	76.7 ± 0.4
Yang et al. [31]	**(3)** SP + sparse codes + max pooling + linear SVM	$\mathbf{67.0 \pm 0.5}$	$\mathbf{73.2 \pm 0.5}$	80.3 ± 0.9
Yang et al. [31]	**(4)** SP + sparse codes + max pooling + kernel SVM	60.4±1.0	–	77.7±0.7
Zhang et al. [32]	kNN-SVM	59.1 ± 0.6	66.2 ± 0.5	-
Zhou et al. [33]	SP + Gaussian mixture	–	–	$\mathbf{84.1 \pm 0.5}$

Table 2. Results obtained by several recognition schemes using a single type of descriptors. Bold numbers in parentheses preceding the method description indicate methods reimplemented in this paper. SP: spatial pyramid.

Caltech-101 [6] and Scenes datasets [12] as benchmarks. These datasets respectively comprise 101 object categories (plus a "background" category) and fifteen scene categories. Following the usual procedure [12, 31], we use 30 training images and the rest for testing (with a maximum of 50 test images) on the Caltech-101 dataset, and 100 training images and the rest for testing on the Scenes dataset. Experiments are conducted over 10 random splits of the data, and we report the mean accuracy and its standard deviation. Hyperparameters of the model are selected by cross-validation within the training set. The general architecture follows [12]. Low-level descriptors $\boldsymbol{x}_i$ are 128-dimensional SIFT descriptors [18] of 16×16 patches. The descriptors are extracted on a dense grid rather than at interest points, as this procedure has been shown to yield superior scene classification [17]. Pooling regions m comprise the cells of 4×4, 2×2 and 1×1 grids (forming a three-level pyramid). We use the SPAMS toolbox [1] to compute sparse codes.

3.1. Interaction Between Modules

Here, we perform a systematic cross evaluation of all the coding, pooling and classifier types presented in Sec. 2, with SIFT descriptors extracted densely every 8 pixels. Results are presented on Table 1. The ranking of performance when changing a particular module (e.g., coding) is quite consistent:

- Sparse coding improves over soft quantization, which improves over hard quantization;
- Max pooling almost always improves over average pooling, dramatically so when using a linear SVM;
- The intersection kernel SVM performs similarly or better than the linear SVM.

In particular, the global feature obtained when using hard vector quantization with max pooling achieves high accuracy with a linear classifier, while being *binary*, and merely recording the presence or absence of each codeword in the pools. While much research has been devoted to devising the best possible coding module, our results show that with linear classification, switching from average to max pooling increases accuracy more than switching from hard quantization to sparse coding. These results could serve as guidelines for the design of future architectures.

For comparison, previously published results obtained using one type of descriptors on the same dataset are shown on Table 2. Note that better performance has been reported with multiple descriptor types (e.g., methods using multiple kernel learning have achieved $77.7\% \pm 0.3$ [7] and $78.0\% \pm 0.3$ [2, 28] on Caltech-101 with 30 training examples), or subcategory learning (83% on Caltech-101 [26]). The coding and pooling module combinations used in [27, 31] are included in our comparative evaluation (bold numbers in parentheses on Tables 1 and 2). Overall, our results confirm the experimental findings in these works, except that we do not find superior performance for the linear SVM, compared to the intersection kernel SVM, with sparse codes and max pooling, contrary to Yang et al. [31]. Results of our reimplementation are similar to those in [12]. The better performance than that reported by Van Gemert et al. [27] or Yang et al. [31] on the Scenes is not surprising since their baseline accuracy for the method in [12] is also lower, which they attributed to implementation differences. Discrepancies with results from Yang et al. [31] may arise from their using a differentiable quadratic hinge loss instead of the standard hinge loss in the SVM, and a different type of normalization for SIFT descriptors.

3.2. Macrofeatures and denser SIFT sampling

In convolutional neural networks (e.g., [16, 23]), spatial neighborhoods of low-level features are encoded jointly. On the other hand, codewords in bag-of-features methods usually encode low-level features at a single location (see Fig. 1). We propose to adapt the joint encoding scheme to the spatial pyramid framework.

Jointly encoding L descriptors in a local spatial neighborhood $\mathcal{L}_i$ amounts to replacing Eq. (1) by:

$$\boldsymbol{\alpha}_i = f([\boldsymbol{x}_{i_1}^T \cdots \boldsymbol{x}_{i_L}^T]^T), \;\; i_1, \cdots, i_L \in \mathcal{L}_i. \tag{9}$$

In the following, we call *macrofeatures* vectors that jointly encode a small neighborhood of SIFT descriptors. The encoded neighborhoods are squares determined by two parameters: the side of the square (e.g., 2×2 square on Fig. 1), and a subsampling parameter determining how many SIFT descriptors to skip along each dimension when selecting neighboring features. For example, a 3×3 macrofeature with a subsampling parameter of 2 jointly encodes 9 descriptors out of a 6×6 grid, skipping every other column and row.

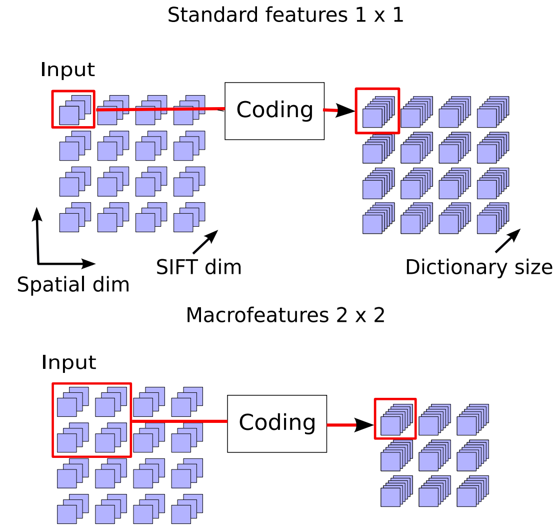

Figure 1. Standard features encode the SIFT features at a single spatial point. Macrofeatures jointly encode small spatial neighborhoods of SIFT features (i.e., the input of the coding module is formed by concatenating nearby SIFT descriptors).

We have experimented with different macrofeature parameters, and denser sampling of the underlying SIFT descriptor map (e.g., extracting SIFT every 4 pixels instead of 8 pixels as in the baseline of [12]). We have tested sampling densities of 2 to 10, and macrofeatures of side length 2 to 4 and subsampling parameter 1 to 4. When using sparse coding and max pooling, the best parameters (selected by cross-validation within the training set) for SIFT sampling density, macrofeature side length and subsampling parameter are respectively of 4, 2, 4 for the Caltech-101 dataset, and 8, 2, 1 for the Scenes dataset. Our results (Table 1, bottom) show that large improvements can be gained on the Caltech-101 benchmark, by merely sampling SIFT descriptors more finely, and jointly representing nearby descriptors, yielding a classification accuracy of 75.7%, which to the best of our knowledge is significantly better than all published classification schemes using a single type of low-level descriptor. However, we have not found finer sampling and joint encoding to help recognition significantly on the Scenes dataset.

4. Discriminative dictionaries

The feature extraction schemes presented so far are all unsupervised. When using sparse coding, an adaptive dictionary is learned by minimizing a regularized reconstruction error. While this ensures that the parameters of the dictionary are adapted to the statistics of the data, the dictionary is not optimized for the classification task. In this section, we introduce a novel supervised method to learn the dictionary.

Several authors have proposed methods to obtain discriminative codebooks. Lazebnik and Raginsky [11] incorporate discriminative information by minimizing the loss of mutual information between features and labels during the quantization step. Winn et al. [30] prune a large codebook iteratively by fusing codewords that do not contribute to discrimination. However these methods are optimized for vector quantization. Mairal et al. [20] have proposed an algorithm to train discriminative dictionaries for sparse coding, but it requires each encoded vector to be labelled. Instead, the approach we propose is adapted to global image statistics.

With the same notation as before, let us consider the extraction of a global image representation by sparse coding and average pooling over the whole image I:

$$\hat{\boldsymbol{x}}_i^T = [\boldsymbol{x}_{i_1}^T \cdots \boldsymbol{x}_{i_L}^T], \ \ i_1, \cdots, i_L \in \mathcal{L}_i, \tag{10}$$

$$\boldsymbol{\alpha}_i = \operatorname*{argmin}_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}, \mathbf{D}) \triangleq \|\hat{\boldsymbol{x}}_i - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1, \tag{11}$$

$$\boldsymbol{h} = \frac{1}{|I|} \sum_{i \in I} \boldsymbol{\alpha_i}, \tag{12}$$

$$\boldsymbol{z} = \boldsymbol{h}. \tag{13}$$

Consider a binary classification problem. Let $\boldsymbol{z}^{(n)}$ denote the global image representation for the n-th training image, and $y_n \in \{-1, 1\}$ the image label. A linear classifier is trained by minimizing with respect to parameter θ the regularized logistic cost:

$$C_s = \frac{1}{N} \sum_{n=1}^{N} \log\left(1 + e^{-y_n \theta^T \boldsymbol{z}^{(n)}}\right) + \lambda_r \|\theta\|_2^2, \tag{14}$$

where λ_r denotes a regularization parameter. We use logistic regression because its level of performance is typically similar to that of linear SVMs but unlike SVMs, its loss function is differentiable. We want to minimize the supervised cost C_s with respect to $\mathbf{D}$ to obtain a more discriminative dictionary. Using the chain rule, we obtain:

$$\frac{\partial C_s}{\partial \mathbf{D}_{jk}} = -\frac{1}{N} \sum_{n=1}^{N} y_n \left(1 - \sigma(y_n \theta . \boldsymbol{z}^{(n)})\right) \theta^T \frac{\partial \boldsymbol{z}^{(n)}}{\partial \mathbf{D}_{jk}} \tag{15}$$

$$\frac{\partial \boldsymbol{z}^{(n)}}{\partial \mathbf{D}_{jk}} = \frac{1}{|I^{(n)}|} \sum_{i \in I^{(n)}} \frac{\partial \boldsymbol{\alpha}_i^{(n)}}{\partial \mathbf{D}_{jk}}, \tag{16}$$

where σ denotes the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$. We need to compute the gradient $\nabla_{\mathbf{D}}(\boldsymbol{\alpha}_i)$. Since the $\boldsymbol{\alpha}_i$ minimize Eq. (11), they verify:

$$\boldsymbol{\alpha} = (\mathbf{D_{\boldsymbol{\alpha}}}^T \mathbf{D_{\boldsymbol{\alpha}}})^{-1} (\mathbf{D_{\boldsymbol{\alpha}}}^T \hat{\boldsymbol{x}} - \lambda \text{sign}(\boldsymbol{\alpha})), \tag{17}$$

where we have dropped subscript i to limit notation clutter, and $\mathbf{D}_{\boldsymbol{\alpha}}$ denotes the columns corresponding to the active set of $\boldsymbol{\alpha}$ (i.e., the few columns of $\mathbf{D}$ used in the decomposition of the input). Note that this formula cannot be used to compute $\boldsymbol{\alpha}$, as parts of the right-hand side of the equation depend on $\boldsymbol{\alpha}$ itself, but it can be used to compute a gradient once $\boldsymbol{\alpha}$ is known. When perturbations of the dictionary are small, the active set of $\boldsymbol{\alpha}$ often stays the same (since the correlation between the atoms of the dictionary and the input vector varies continuously with the dictionary). Assuming that it is constant, we can compute the gradient of the active coefficients with respect to the active columns of $\mathbf{D}$ (setting it to 0 elsewhere):

$$\frac{\partial \tilde{\boldsymbol{\alpha}}_k}{\partial (\mathbf{D_{\boldsymbol{\alpha}}})_{ij}} = \mathbf{b}_i \mathbf{A}_{kj} - \tilde{\boldsymbol{\alpha}}_j \mathbf{C}_{ki}, \tag{18}$$

$$\mathbf{A} \triangleq (\mathbf{D_{\boldsymbol{\alpha}}}^T \mathbf{D_{\boldsymbol{\alpha}}})^{-1}, \tag{19}$$

$$\mathbf{b} \triangleq \hat{\boldsymbol{x}} - \mathbf{D}\boldsymbol{\alpha}, \tag{20}$$

$$\mathbf{C} \triangleq \mathbf{A}\mathbf{D_{\boldsymbol{\alpha}}}^T, \tag{21}$$

where $\tilde{\boldsymbol{\alpha}}_k$ denotes the k-th non-zero component of $\boldsymbol{\alpha}$.

We train the discriminative dictionary by stochastic gradient descent [4, 14]. Recomputing the sparse decompositions $\boldsymbol{\alpha}_i$ at each location of a training image at each iteration is costly. To speed-up the computation while remaining closer to global image statistics than with individual patches, we approximate $\boldsymbol{z}^{(n)}$ by pooling over a random sample of ten locations of the image. Furthermore, we update only a random subset of coordinates at each iteration, since computation of the gradient is costly. We then test the dictionary with max pooling and a three-layer spatial pyramid, using either a linear or intersection kernel SVM.

	Unsup	Discr	Unsup	Discr
Linear	83.6 ± 0.4	$\mathbf{84.9 \pm 0.3}$	84.2 ± 0.3	$\mathbf{85.6 \pm 0.2}$
Intersect	84.3 ± 0.5	$\mathbf{84.7 \pm 0.4}$	84.6 ± 0.4	85.1 ± 0.5

Table 3. Results of learning discriminative dictionaries on the Scenes dataset, for dictionaries of size 1024 (left) and 2048 (right), with 2×2 macrofeatures and grid resolution of 8 pixels,

We compare performance of dictionaries of sizes 1024 and 2048 on the Scenes dataset, encoding 2×2 neighborhoods of SIFT. Results (Table 3) show that discriminative dictionaries perform significantly better than unsupervised dictionaries. A discriminative dictionary of 2048 codewords achieves 85.6% correct recognition performance, which to the best of our knowledge is the highest published classification accuracy on that dataset for a single feature type. Discriminative training of dictionaries with our method on the Caltech-101 dataset has yielded only very little improvement, probably due to the scarcity of training data.

5. Comparing Average and Max Pooling

One of the most striking results of our comparative evaluation is that the superiority of max pooling over average pooling generalizes to many combinations of coding schemes and classifiers. Several authors have already stressed the efficiency of max pooling [10, 31], but they have not given theoretical explanations to their findings. In this section, we study max pooling in more details theoretically and experimentally.

5.1. A Theoretical Comparison of Pooling Strategies

With the same notation as before, consider a binary linear classification task over cluttered images. Pooling is performed over the whole image, so that the pooled feature $\boldsymbol{h}$ is the global image representation. Linear classification requires distributions of $\boldsymbol{h}$ over examples from positive and negative classes (henceforth denoted by $+$ and $-$) to be well separated.

We model the distribution of image patches of a given class as a mixture of two distributions [21]: patches are taken from the actual class distribution (foreground) with probability $(1-w)$, and from a clutter distribution (background) with probability w, with clutter patches being present in both classes ($+$ or $-$). Crucially, we model the amount of clutter w as varying between images (while being fixed for a given image).

There are then two sources of variance for the distribution $p(\boldsymbol{h})$: the intrinsic variance caused by sampling from a finite pool for each image (which causes the actual value of $\boldsymbol{h}$ over foreground patches to deviate from its expectation), and the variance of w (which causes the expectation of $\boldsymbol{h}$ itself to fluctuate from image to image depending on their clutter level). If the pool cardinality N is large, average pooling is robust to intrinsic foreground variability, since the variance of the average decreases in $\frac{1}{N}$. This is usually not the case with max pooling, where the variance can increase with pool cardinality depending on the foreground distribution.

However, if the amount of clutter w has a high variance, it causes the distribution of the average over the image to spread, as the expectation of $\boldsymbol{h}$ for each image depends on w. Even if the foreground distributions are well separated, variance in the amount of clutter creates overlap between the mixture distributions if the mean of the background distribution is much lower than that of the foreground distributions. Conversely, max pooling can be robust to clutter if the mean of the background distribution is sufficiently low. This is illustrated on Fig. 2, where we have plotted the empirical distributions of the average of 10 pooled features sharing the same parameters. Simulations are run using 1000 images of each class, composed of $N = 500$ patches. For each image, the clutter level w is drawn from a truncated normal distribution with either low (top) or high (bottom) variance. Local feature values at each patch are drawn from a mixture of exponential distributions, with a lower mean for background patches than foreground patches of either

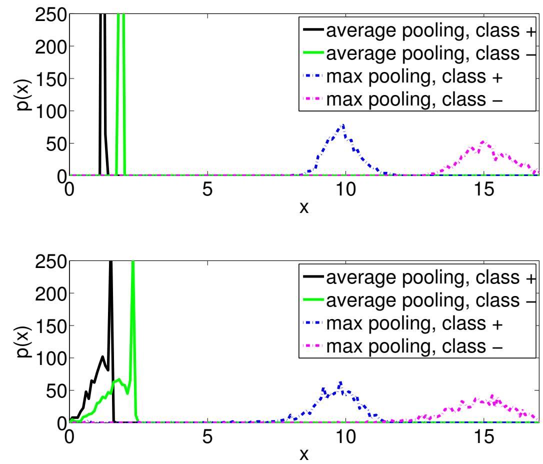

Figure 2. Empirical probability densities of $x = \frac{1}{K}\sum_{j=1}^{K} h_j$, simulated for two classes classes of images forming pools of cardinality $N = 500$. The local features are drawn from one of three exponential distributions. When the clutter is homogeneous across images (top), the distributions are well separated for average pooling and max pooling. When the clutter level has higher variance (bottom), the max pooling distributions (dashed lines) are still well separated while the average pooling distributions (solid lines) start overlapping.

class. When the clutter has high variance (Fig. 2, bottom), distributions remain well separated with max pooling, but have significant overlap with average pooling.

We now refine our analysis in two cases: sparse codes and vector quantized codes.

5.1.1 Sparse Codes.

In the case of a positive decomposition over a dictionary, we model the distribution of the value of feature j for each patch by an exponential distribution with mean μ_j, variance μ_j^2, and density $f(x) = \frac{1}{\mu_j} \exp^{-\frac{x}{\mu_j}}$. The choice of an exponential distribution (or a Laplace distribution when decompositions are not constrained to be positive) to model sparse codes seems appropriate because it is highly kurtotic and sparse codes have heavy tails.

The corresponding cumulative distribution function is $F(x) = 1 - e^{-\frac{x}{\mu_j}}$. The cumulative distribution function of the max-pooled feature is $F^N(x) = (1-e^{-\frac{x}{\mu_j}})^N$ for a pool of size N. Clutter patches are sampled from a distribution of mean μ_b. Let N_f and N_b denote respectively the number of foreground and background patches, $N = N_f + N_b$. Assuming N_f and N_b are large, Taylor expansions of the cumulative distribution functions of the maxima yield that 95% of the probability mass of the maximum over the background patches will be below 95% of the probability mass of the maximum over the foreground patches provided that

$N_b < |\log(0.95)| \left(\frac{N_f}{|\log(0.05)|}\right)^{\frac{\mu_j}{\mu_b}}$. In a binary discrimination task between two comparatively similar classes, if an image is cluttered by many background patches, with $\mu_b \ll \mu_j^+$ and $\mu_b \ll \mu_j^-$, max-pooling can be relatively immune to background patches, while average-pooling can create overlap between the distributions (see Fig. 2). For example, if $\mu_b < 2\mu_j$ and $N_f = 500$, having fewer than $N_b < 1400$ background patches virtually guarantees that the clutter will have no influence on the value of the maximum. Conversely, if $N_b < \frac{N_f}{59} \leq \frac{|\log(0.95)|}{|\log(0.05)|} N_f$, clutter will have little influence for μ_b up to μ_j. Thus, max-pooling creates immunity to two different types of clutter: ubiquitous with low feature activation, and infrequent with higher activation.

However, a downside is that the ratio of the mean to the standard deviation of the maximum distribution does *not* decrease as $\frac{1}{\sqrt{N}}$, as in the case of the distribution of the average. In fact, the mean and variance of the maximum distribution over N samples can be shown to be:

$$\nu = (\mathcal{H}(N)) .\mu_j,$$
$$\sigma^2 = \left(\sum_{l=1}^{N} \frac{1}{l}(2\mathcal{H}(l) - \mathcal{H}(N))\right) .\mu_j^2,$$

where $\mathcal{H}(k) = \sum_{i=1}^{k} \frac{1}{i}$ denotes the harmonic series, which grows like $\log(k)$. It can be shown that:

$$\sum_{l=1}^{N} \frac{1}{l}(2\mathcal{H}(l) - \mathcal{H}(N)) = \log(N) + O(1),$$

so that the ratio $\frac{\nu}{\sigma}$ decreases like $\frac{1}{\sqrt{\log(N)}}$. Thus, if the pool cardinality is too small, the distributions of foreground patches from both classes will be better separated with average pooling than max pooling.

5.1.2 Vector Quantization.

We model binary patch codes for feature j by i.i.d. Bernoulli random variables of mean μ_j. The distribution of the average-pooled feature also has mean μ_j, and its variance decreases like $\frac{1}{N}$. The maximum is a Bernoulli variable of mean $1 - (1 - \mu_j)^N$ and variance $(1 - (1 - \mu_j)^N)(1 - \mu_j)^N$. Thus, it is 1 with probability 0.95 if $N \geq \frac{\log(0.05)}{\log(1-\mu_j)} \approx \frac{|\log(0.05)|}{\mu_j}$, and 0 with probability 0.95 if $N \leq \frac{\log(0.95)}{\log(1-\mu_j)} \approx \frac{|\log(0.95)|}{\mu_j}$, for $\mu_j \ll 1$. The separability of classes depends on sample cardinality N. There exists a sample cardinality N for which the maximum over class $+$ is 0 with probability 0.95, while the maximum over class $-$ is 1 with probability 0.95, if:

$$\frac{\mu_j^-}{\mu_j^+} > \frac{\log(0.05)}{\log(0.95)}, \text{ e.g. if } \frac{\mu_j^-}{\mu_j^+} > 59.$$

As $\sum_j \mu_j = 1$ in the context of vector quantization, μ_j becomes very small on average if the codebook is very large. For $\mu_j \ll 1$, the characteristic scale of the transition from 0 to 1 is $\frac{1}{\mu_j}$, hence the pooling cardinality range corresponding to easily separable distributions can be quite large if the mean over foreground patches from one class is much higher than both the mean over foreground patches from the other class and the mean over background patches.

5.2. Experimental Validation

Our analysis suggests that there may be a purely statistical component to the improvement seen with max pooling when using pyramids instead of plain bags of features. Taking the maximum over several pools of smaller cardinality may lead to a richer estimate, since max pooling differs from average pooling in two important ways:

- the maximum over a pool of smaller cardinality is not merely an estimator of the maximum over a larger pool;
- the variance of the maximum is not inversely proportional to pool cardinality, so that summing over several estimates (one for each smaller pool) can provide a smoother output than if pooling had merely been performed over the merged smaller pools.

We have tested this hypothesis by comparing three types of pooling procedures: standard whole-image and two-level pyramid pooling, and random two-level pyramid pooling, where local features are randomly permuted before being pooled, effectively removing all spatial information.

For this experiment, SIFT features are extracted densely every 8 pixels, and encoded by hard quantization over a codebook of size 256 for Caltech-101, 1024 for the Scenes. The pooled features are concatenated and classified with a linear SVM, trained on 30 and 100 examples for Caltech-101 and the Scenes, respectively.

	Caltech 101		15 Scenes	
Pyramid	1×1	2×2	1×1	2×2
Avg, random	31.7 ± 1.0	29.5 ± 0.5	71.0 ± 0.8	69.4 ± 0.8
Avg, spatial		43.2 ± 1.4		73.2 ± 0.7
Max, random	26.2 ± 0.7	33.1 ± 0.9	69.5 ± 0.6	72.8 ± 0.3
Max, spatial		50.7 ± 0.8		77.2 ± 0.6

Table 4. Classification accuracy for different sets of pools and pooling operators.

Results (Table 4) show that with max pooling, a substantial part of the increase in accuracy seen when using a two-level pyramid instead of a plain bag of features is indeed still present when locations are randomly shuffled. On the contrary, the performance of average pooling tends to deteriorate with the pyramid, since the added smaller, random pools only contribute noisier, redundant information.

6. Discussion

By deconstructing the mid-level coding step of a well-accepted recognition architecture, it appears that any parameter in the architecture can contribute to recognition performance; in particular, surprisingly large performance increases can be obtained by merely sampling the low-level descriptor map more finely, and representing neighboring descriptors jointly. We have presented a scheme to train supervised discriminative dictionaries for sparse coding; our ongoing research focuses on extending this framework to the much harder PASCAL datasets, on which methods very similar to the ones discussed in this paper [31] currently define the state of the art. We plan to combine our discriminative sparse training algorithm with the various techniques (e.g., local coordinate coding) that have been successful on PASCAL. Another research direction we are pursuing is the analysis of pooling schemes. Understanding pooling operators is crucial to good model design, since common heuristics suited to average pooling may be suboptimal in other contexts. In this paper, we have only briefly touched upon the statistical properties of max pooling. We are currently investigating how to expand these theoretical insights, and turn them into guidelines for better architecture design.

Acknowledgements. This work was funded in part by NSF grant EFRI/COPN-0835878 to NYU, and ONR contract N00014-09-1-0473 to NYU. We would like to thank Sylvain Arlot, Olivier Duchenne and Julien Mairal for helpful discussions.

References

[1] `http://www.di.ens.fr/willow/SPAMS/`. 3

[2] `http://www.robots.ox.ac.uk/~vgg/software/MKL/`. 4

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

[4] L. Bottou. Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. 5

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

[6] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples. In *CVPR Workshop GMBV*, 2004. 3

[7] P. Gehler and S. Nowozin. On Feature Combination for Multiclass Object Classification. In *ICCV*, 2009. 4

[8] G. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 2006. 1

[9] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. In *CVPR*, 2008. 3

[10] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009. 6

[11] S. Lazebnik and M. Raginsky. Supervised Learning of Quantizer Codebooks by Information Loss Minimization. *PAMI*, 21, 2008. 4

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 1, 2, 3, 4

[13] 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

[14] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and M. K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998. 5

[15] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006. 2

[16] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 1, 4

[17] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005. 3

[18] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Comp. Vision*, 60(4):91–110, 2004. 1, 3

[19] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online Dictionary Learning for Sparse Coding. In *ICML*, 2009. 2

[20] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised Dictionary Learning. In *NIPS*, 2009. 5

[21] T. Minka. Expectation Propagation for approximate Bayesian inference. In *UAI*, 2001. 6

[22] 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. 2

[23] M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *NIPS 2007*, 2007. 1, 4

[24] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005. 1

[25] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 1, 2

[26] S. Todorovic and N. Ahuja. Learning subcategory relevances for category recognition. In *CVPR*, 2008. 4

[27] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. *PAMI*, (in press), 2010. 2, 3, 4

[28] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proc. Int. Conf. Comp. Vision*, 2009. 4

[29] S. Winder and M. Brown. Learning local image descriptors. In *CVPR*, 2007. 1

[30] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV 2005*. 5

[31] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. In *CVPR*, 2009. 1, 2, 3, 4, 6, 8

[32] H. Zhang, A. C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006. 3

[33] 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. 3