A Theoretical Analysis of Feature Pooling in Visual Recognition

Y-Lan Boureau[2,3] YLAN@CS.NYU.EDU
Jean Ponce[1,2] JEAN.PONCE@ENS.FR
Yann LeCun[3] YANN@CS.NYU.EDU
[1]Laboratoire d'Informatique de l'Ecole Normale Supérieure, 45, rue d'Ulm 75230 Paris CEDEX 05, France
[2]INRIA - WILLOW Project (INRIA/ENS/CNRS UMR 8548), 23, avenue d'Italie, 75214 Paris, France
[3]Courant Institute of Mathematical Sciences New York University, NY 10003, USA

Abstract

Many modern visual recognition algorithms incorporate a step of spatial 'pooling', where the outputs of several nearby feature detectors are combined into a local or global 'bag of features', in a way that preserves task-related information while removing irrelevant details. Pooling is used to achieve invariance to image transformations, more compact representations, and better robustness to noise and clutter. Several papers have shown that the details of the pooling operation can greatly influence the performance, but studies have so far been purely empirical. In this paper, we show that the reasons underlying the performance of various pooling methods are obscured by several confounding factors, such as the link between the sample cardinality in a spatial pool and the resolution at which low-level features have been extracted. We provide a detailed theoretical analysis of max pooling and average pooling, and give extensive empirical comparisons for object recognition tasks.

1. Introduction

Modern computer vision architectures often comprise a spatial pooling step, which combines the responses of feature detectors obtained at nearby locations into some statistic that summarizes the joint distribution of the features over some region of interest. The idea of feature pooling originates in Hubel and Wiesel's seminal work on complex cells in the visual cortex (1962), and is related to Koenderink's concept of locally orderless images (1999). Pooling features over a local neighborhood to create invariance to small transformations of the input is used in a large number of models of visual recognition. The pooling operation is typically a sum, an average, a max, or more rarely some other commutative (i.e., independent of the order of the contributing features) combination rule. Biologically-inspired models of image recognition that use feature pooling include the neocognitron (Fukushima & Miyake, 1982), convolutional networks which use average pooling (LeCun et al., 1989; 1998), or max pooling (Ranzato et al., 2007; Jarrett et al., 2009), the HMAX class of models which uses max pooling (Serre et al., 2005), and some models of the primary visual cortex area V1 (Pinto et al., 2008) which use average pooling. Many popular methods for feature extraction also use pooling, including SIFT (Lowe, 2004), histograms of oriented gradients (HOG) (Dalal & Triggs, 2005) and their variations. In these methods, the dominant gradient orientations are measured in a number of regions, and are pooled over a neighborhood, resulting in a local histogram of orientations. Recent recognition systems often use pooling at a higher level to compute local or global bags of features. This is done by vector-quantizing feature descriptors and by computing the codeword counts over local or global areas (Sivic & Zisserman, 2003; Lazebnik et al., 2006; Zhang et al., 2007; Yang et al., 2009), which is equivalent to average-pooling vectors containing a single 1 at the index of the codeword, and 0 everywhere else (1-of-k codes).

In general terms, the objective of pooling is to transform the joint feature representation into a new, more usable one that preserves important information while discarding irrelevant detail, the crux of the matter being to determine what falls in which category. For example, the assumption underlying the computation of a histogram is that the average feature activation matters, but exact spatial localization does not. Achieving invariance to changes in position or lighting conditions, robustness to clutter, and compactness of representation, are all common goals of pooling.

The success of the spatial pyramid model (Lazebnik et al.,

Appearing in *Proceedings of the 27^{th} International Conference on Machine Learning*, Haifa, Israel, 2010.

2006), which obtains large increases in performance by performing pooling over the cells of a spatial pyramid rather than over the whole image as in plain bag-of-features models (Zhang et al., 2007), illustrates the importance of the spatial structure of pooling neighborhoods. Perhaps more intriguing is the dramatic influence of the way pooling is performed once a given region of interest has been chosen. Thus, Jarrett et al. (2009) have shown that pooling type matters more than careful unsupervised pretraining of features for classification problems with little training data, obtaining good results with random features when appropriate pooling is used. Yang et al. (2009) report much better classification performance on several object or scene classification benchmarks when using the maximum value of a feature rather than its average to summarize its activity over a region of interest. But no theoretical justification of these findings is given. In previous work (Boureau et al., 2010), we have shown that using max pooling on hard-vector quantized features (which produces a binary vector that records the presence of a feature in the pool) in a spatial pyramid brings the performance of linear classification to the level of that obtained by Lazebnik et al. (2006) with an intersection kernel, even though the resulting feature is binary. However, it remains unclear why max pooling performs well in a large variety of settings, and indeed whether similar or different factors come into play in each case.

This paper proposes to fill the gap and to conduct a thorough theoretical investigation of pooling. We compare different pooling operations in a categorization context, and examine how the behavior of the corresponding statistics may translate into easier or harder subsequent classification. We provide experiments in the context of visual object recognition, but the analysis applies to all tasks which incorporate some form of pooling (e.g., text processing from which the bag-of-features method was originally adapted). The main contributions of this paper are (1) an extensive analytical study of the discriminative powers of different pooling operations, (2) the discrimination of several factors affecting pooling performance, including smoothing and sparsity of the features, (3) the unification of several popular pooling types as belonging to a single continuum.

2. Pooling Binary Features

Consider a two-class categorization problem. Intuitively, classification is easier if the distributions from which points of the two classes are drawn have no overlap. In fact, if the distributions are simply shifted versions of one another (e.g., two Gaussian distributions with same variance), linear separability increases monotonically with the magnitude of the shift (e.g., with the distance between the means of two Gaussian distributions of same variance) (Bruckstein & Cover, 1985). In this section, we examine how pooling affects the separability of the resulting feature distributions when the features being pooled are binary vectors (e.g., 1-of-k codes obtained by vector quantization in bag-of-features models).

2.1. Model

Let us examine the contribution of a single feature in a bag-of-features representation (i.e., if the unpooled data is a $P \times k$ matrix of 1-of-k codes taken at P locations, we extract a single P-dimensional column $\boldsymbol{v}$ of 0s and 1s, indicating the absence or presence of the feature at each location). For simplicity, we model the P components of $\boldsymbol{v}$ as i.i.d. Bernoulli random variables. The independence assumption is clearly false since nearby image features are strongly correlated, but the analysis of this simple model nonetheless yields useful predictions that can be verified empirically. The vector $\boldsymbol{v}$ is reduced by a pooling operation f to a single scalar $f(\boldsymbol{v})$ (which would be one component of the k-dimensional representation using all features, e.g., one bin in a histogram). We consider two pooling types: average pooling $f_a(\boldsymbol{v}) = \frac{1}{P}\sum_{i=1}^{P} \boldsymbol{v}_i$, and max pooling $f_m(\boldsymbol{v}) = \max_i \boldsymbol{v}_i$.

2.2. Distribution Separability

Given two classes C_1 and C_2, we examine the separation of conditional distributions $p(f_m|C_1)$ and $p(f_m|C_2)$, and $p(f_a|C_1)$ and $p(f_a|C_2)$. Viewing separability as a signal-to-noise problem, better separability can be achieved by either increasing the distance between the means of the two class-conditional distributions, or reducing their standard deviation.

We first consider average pooling. The sum over P i.i.d. Bernoulli variables of mean α follows a binomial distribution $B(P, \alpha)$. Consequently, the distribution of f_a is a scaled-down version of the binomial distribution, with mean $\mu_a = \alpha$, and variance $\sigma_a^2 = \alpha(1-\alpha)/P$. The expected value of f_a is independent of sample size P, and the variance decreases like $\frac{1}{P}$; therefore the separation ratio of means' difference over standard deviation decreases monotonically like $\frac{1}{\sqrt{P}}$.

Max pooling is slightly less straightforward, so we examine means' separation and variance separately in the next two sections.

2.2.1. Means' Separation of Max-Pooled Features

f_m is a Bernoulli variable of mean $\mu_m = 1 - (1-\alpha)^P$ and variance $\sigma_m^2 = (1-(1-\alpha)^P)(1-\alpha)^P$. The mean increases monotonically from 0 to 1 with sample size P. Let ϕ denote the separation of class-conditional expectations of max-pooled features,

$$\phi(P) \triangleq |\mathbb{E}(f_m|C_1) - \mathbb{E}(f_m|C_2)| = |(1-\alpha_2)^P - (1-\alpha_1)^P|,$$

where $\alpha_1 \triangleq \mathbb{P}(\boldsymbol{v}_i = 1|C_1)$ and $\alpha_2 \triangleq \mathbb{P}(\boldsymbol{v}_i = 1|C_2)$. We abuse notation by using ϕ to refer both to the function defined on sample cardinality P and its extension to $\mathbb{R}$. It is easy to show that ϕ is increasing between 0 and

$$P_M \triangleq \left| \log\left(\frac{\log(1-\alpha_2)}{\log(1-\alpha_1)}\right) / \log\left(\frac{1-\alpha_1}{1-\alpha_2}\right) \right|,$$

and decreasing between P_M and ∞, with $\lim_0 \phi = \lim_\infty \phi = 0$.

Noting that $\phi(1) = |\alpha_1 - \alpha_2|$ is the distance between the class-conditional expectations of average-pooled features, there exists a range of pooling cardinalities for which the distance is greater with max pooling than average pooling if and only if $P_M > 1$. Assuming $\alpha_1 > \alpha_2$, it is easy to show that $P_M \leq 1 \Rightarrow \alpha_1 > 1 - \frac{1}{e} > 0.63$. This implies that the feature is selected to represent more than half the patches on average, which in practice does not happen in usual bag-of-features contexts, where codebooks comprise more than a hundred codewords.

2.2.2. Variance of Max-Pooled Features

The variance of the max-pooled feature is $\sigma_m^2 = (1-(1-\alpha)^P)(1-\alpha)^P$. A simple analysis of the continuous extension of this function to real numbers shows that it has limit 0 at 0 and ∞, and is increasing then decreasing, reaching its maximum of 0.5 at $\log(2)/|\log(1-\alpha)|$. The increase of the variance can play against the better separation of the expectations of the max-pooled feature activation, when parameter values α_1 and α_2 are too close for the two classes. Several regimes for the variation of means separation and standard deviations are shown in Fig. 1.

2.2.3. Conclusions and Predictions

Our simplified analysis leads to several predictions:

- Max pooling is particularly well suited to the separation of features that are very sparse (i.e., have a very low probability of being active)
- Using all available samples to perform the pooling may not be optimal
- The optimal pooling cardinality should increase with dictionary size

The first point can be formalized by observing that the characteristic pooling cardinality $|\frac{1}{\log(1-\alpha)}|$ ($\approx \frac{1}{\alpha}$ in the case $\alpha \ll 1$), scales the transition to the asymptotic regime (low variance, high probability of activation): the maximum of the variance is reached at $P = \log(2)/|\log(1-\alpha)|$, and:

$$\mathbb{P}(f_m(\boldsymbol{v}) = 1) > \lambda \Leftrightarrow P > \frac{\log(1-\lambda)}{\log(1-\alpha)}.$$

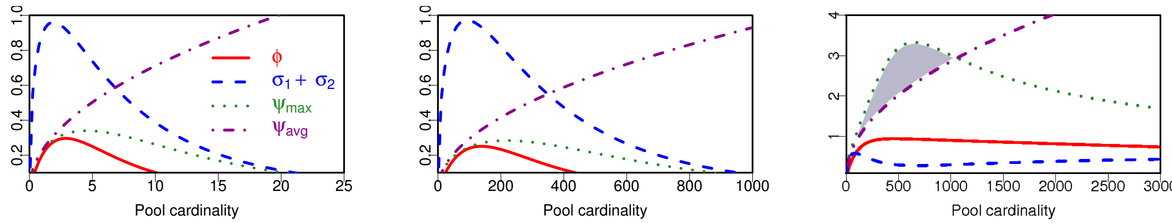
Consequently, the range of cardinalities for which max pooling achieves good separation between two classes doubles if the probability of activation of the feature for both classes is divided by two. A particularly favorable regime is $\alpha_2 \ll \alpha_1 \ll 1$ – that is, a feature which is rare, but relatively much more frequent in one of the two classes; in that case, both classes reach their asymptotic regime for very different sample cardinalities ($\frac{1}{\alpha_1}$ and $\frac{1}{\alpha_2}$).

We have recently conducted preliminary experiments related to the second point (Boureau et al., 2010) – namely, that better performance can be obtained by using smaller pooling cardinalities. We have compared the performance of whole-image pooling, regular two-level spatial pyramid pooling, and a two-level pyramid where the smaller pools are taken randomly instead of spatially. In the random pyramid setting, the performance of max pooling is intermediate between that obtained with whole-image and spatial pyramid pooling, while the classification using average pooling becomes worse than with whole-image pooling. However, a number of concurrent factors could explain the increased accuracy: (1) smaller pooling cardinality, (2) smoothing over multiple estimates (one per finer cell of the pyramid), (3) estimation of two distinct features (the maximum over the full and partial cardinalities, respectively). The more comprehensive experiments presented in the next section resolve this ambiguity by isolating each factor.

Finally, the increase of optimal pooling cardinality with dictionary size is related to the link underlined above between the sparsity of the features (defined here as the probability of them being 0) and the discriminative power of max-pooling, since the expected feature activations sum to one in the general bag-of-features setting (exactly one feature is activated at each location), resulting in a mean expected activation of $\frac{1}{k}$ with a k-word codebook. Thus, k gives an order of magnitude for the characteristic cardinality scale of the transition to the asymptotic regime, for a large enough codebook.

2.3. Experiments

We test our conjectures by running experiments on the Scenes (Lazebnik et al., 2006) and Caltech-101 (Fei-Fei et al., 2004) datasets, which respectively comprise 101 object categories (plus a "background" category) and fifteen scene categories. In all experiments, the features being pooled are local codes representing 16×16 SIFT descriptors that have been densely extracted using the parameters yielding the best accuracy in our previous



(a) $\alpha_1 = 0.4, \alpha_2 = 0.2$ (b) $\alpha_1 = 1.10^{-2}, \alpha_2 = 5.10^{-3}$ (c) $\alpha_1 = 1.10^{-2}, \alpha_2 = 1.10^{-4}$

Figure 1. $\phi(P) = |(1-\alpha_1)^P - (1-\alpha_2)^P|$, σ_1 and σ_2 denote the distance between the expectations of the max-pooled features of mean activation α_1 and α_2, and their standard deviations, respectively. $\psi_{max} = \phi/(\sigma_1+\sigma_2)$ and $\psi_{avg} = |\alpha_1-\alpha_2|.\sqrt{P}/(\sqrt{\alpha_1.(1-\alpha_1)}+\sqrt{\alpha_2.(1-\alpha_2)})$ give a measure of separability for max and average pooling. ϕ reaches its peak at smaller cardinalities than ψ_{max}. (a) When features have relatively large activations, the peak of separability is obtained for small cardinalities (b) With sparser feature activations, the range of the peak is much larger (note the change of scale in the x axis). (c) When one feature is much sparser than the other, ψ_{max} can be larger than ψ_{avg} for some cardinalities (shaded area). Best viewed in color.

work (2010) (every 8 pixels for the Scenes and every 4 pixels for Caltech-101). The codes jointly represent 2×2 neighborhoods of SIFT descriptors, with subsampling of 1 and 4 for the Scenes and Caltech-101, respectively. Features are pooled over the whole image using either average or max pooling. Classification is performed with a one-versus-one support vector machine (SVM) using a linear kernel, and 100 and 30 training images per class for the Scenes and Caltech-101 datasets, respectively, and the rest for testing, following the usual experimental setup. We report the average per-class recognition rate, averaged over 10 random splits of training and testing images.

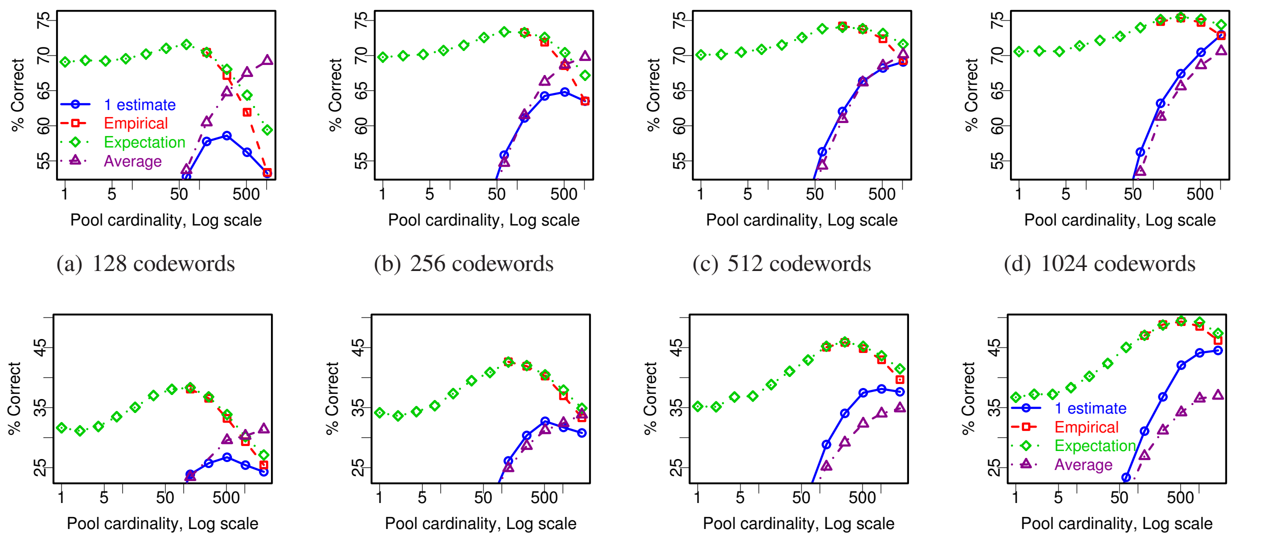
2.3.1. Optimal Pooling Cardinality

We first test whether recognition can indeed improve for some codebook sizes when max pooling is performed over samples of smaller cardinality, as predicted by our analysis. Recognition performance is compared using either average or max pooling, with various combinations of codebook sizes and pooling cardinalities. We use whole-image rather than pyramid or grid pooling, since having several cells of same cardinality provides some smoothing that is hard to quantify. Results are presented in Fig. 2. Recognition performance of average-pooled features (*Average* in Fig. 2) drops with pooling cardinality for all codebook sizes, as expected; performance also drops with max pooling (*1 estimate* in Fig. 2) when the codebook size is large. However, noticeable improvements appear at intermediate cardinalities for the smaller codebook sizes (compare blue, solid curves on the left and right of Fig. 2), as predicted by our analysis.

Next, we examine whether better recognition can be achieved when using a smoother estimate of the expected max-pooled feature activation. We consider two ways of refining the estimate. First, if only a fraction of all samples is used, a smoother estimate can be obtained by replacing the single max by an empirical average of the max over different subsamples, the limit case as pool cardinality decreases being average pooling. The second approach directly applies the formula for the expectation of the maximum $(1-(1-\alpha)^P$, using the same notation as before) to the empirical mean computed using all samples. This has the benefit of removing the constraint that P be smaller than the number of available samples, in addition to being computationally very simple. Results using these two smoothing strategies are plotted in Fig. 2 under labels *Empirical* and *Expectation*, respectively. Smoothing the estimate of the max-pooled features always helps, especially at smaller pooling cardinalities. The best performance is then obtained with pooling cardinalities smaller than the full cardinality in all our experiments. As predicted, the maximum of the curve shifts towards larger cardinality as codebook size increases. The best estimate of the max-pooled feature is the expectation computed from the empirical mean, $1-(1-\alpha)^P$. P here simply becomes the parameter of a nonlinear function applied to the mean. In all cases tested, using this nonlinear function with the optimal P outperforms both average and max pooling.

2.3.2. Combining Multiple Pooling Cardinalities

The maximum over a pool of smaller cardinality is not merely an estimator of the maximum over a large pool; therefore, using different pool cardinalities (e.g., using a spatial pyramid instead of a grid) may provide a more powerful feature, independently of the difference in spatial structure. Using a codebook of size 256, we compare recognition rates using jointly either one, two, or three different pooling cardinalities, with average pooling, max pooling with a single estimate per pooling cardinality, or max pooling smoothed by using the theoretical expectation. Results presented in Table 1 show that combining cardinalities improves performance with max pooling only if the estimate has not been smoothed. Thus, the simultaneous presence of multiple cardinalities does not seem to provide



(a) 128 codewords (b) 256 codewords (c) 512 codewords (d) 1024 codewords

(e) 128 codewords (f) 256 codewords (g) 512 codewords (h) 1024 codewords

Figure 2. Influence of pooling cardinality and smoothing on performance. Top row: Scenes dataset. Bottom row: Caltech-101 dataset. 1 estimate: max computed over a single pool. Empirical: empirical average of max-pooled features over several subsamples (not plotted for smaller sizes, when it reaches the expectation) Expectation: theoretical expectation of the maximum over P samples $1-(1-\alpha)^P$, computed from the empirical average α. Average: estimate of the average computed over a single pool. Best viewed in color.

any benefit beyond that of an approximate smoothing.

Table 1. Classification results with whole-image pooling over binary codes ($k = 256$). *One* indicates that features are pooled using a single cardinality, *Joint* that the larger cardinalities are also used. *SM*: smooth maximum $(1-(1-\alpha)^P)$.

Smallest cardinality		1024	512	256
Caltech 101	Avg, One	$\mathbf{32.4 \pm 1.1}$	31.3 ± 1.0	28.6 ± 1.1
	Avg, Joint		31.9 ± 1.2	32.1 ± 1.2
	Max, One	31.7 ± 1.4	32.7 ± 1.3	30.4 ± 2.3
	Max, Joint		34.4 ± 0.7	$\mathbf{35.8 \pm 0.9}$
	SM, One	37.9 ± 0.6	40.5 ± 0.7	$\mathbf{42.0 \pm 1.4}$
	SM, Joint		39.4 ± 1.3	40.6 ± 0.8
15 Scenes	Avg, One	$\mathbf{69.8 \pm 0.7}$	68.7 ± 0.8	66.3 ± 0.7
	Avg, Joint		69.6 ± 0.7	69.2 ± 1.0
	Max, One	63.5 ± 0.6	64.8 ± 0.7	64.3 ± 0.4
	Max, Joint		65.4 ± 0.6	$\mathbf{67.1 \pm 0.6}$
	SM, One	67.2 ± 0.8	70.4 ± 0.7	$\mathbf{72.6 \pm 0.7}$
	SM, Joint		69.2 ± 0.7	70.7 ± 0.7

2.3.3. Practical Consequences

In papers using a spatial pyramid (Lazebnik et al., 2006; Yang et al., 2009), there is a coupling between the pooling cardinality and other parameters of the experiment: the pooling cardinality is the density at which the underlying low-level feature representation have been extracted (e.g., SIFT features computed every 8 pixels in (Lazebnik et al., 2006)) multiplied by the spatial area of each spatial pool. While using all available samples is optimal for average pooling, this is usually not the case with max pooling over binary features, particularly when the size of the codebook is small. Instead, the pooling cardinality for max pooling should be adapted to the dictionary size, and the remaining samples should be used to smooth the estimate. Another, simpler way to achieve similar or better performance is to apply to the average-pooled feature the nonlinear transformation corresponding to the expectation of the maximum, (i.e., $1-(1-\alpha)^P$, using the same notation as before); in addition, the parameter P is then no longer limited by the number of available samples in a pool, which may be important for very large codebooks. Our experiments using binary features in a three-level pyramid show that this transformation yields improvement over max pooling for all codebook sizes (Table 2). The increase in accuracy is small, however the difference is consistently positive when looking at experimental runs individually instead of the difference in the averages over ten runs.

Table 2. Recognition accuracy with 3-level pyramid pooling over binary codes. One-vs-all classification has been used in this experiment. *Max*: max pooling using all samples. *SM*: smooth maximum (expected value of the maximum computed from the average $1-(1-\alpha)^P$), using a pooling cardinality of $P = 256$ for codebook sizes 256 and 512, $P = 512$ for codebook size 1024.

Codebook size		256	512	1024
Caltech 101	Max	67.5 ± 1.0	69.2 ± 1.1	71.0 ± 0.8
	SM	$\mathbf{68.6 \pm 0.9}$	$\mathbf{70.0 \pm 1.2}$	$\mathbf{71.8 \pm 0.8}$
15 Scenes	Max	77.9 ± 0.7	79.4 ± 0.5	80.2 ± 0.4
	SM	$\mathbf{78.2 \pm 0.4}$	$\mathbf{79.9 \pm 0.5}$	$\mathbf{80.5 \pm 0.6}$

3. Pooling Continuous Sparse Codes

Sparse codes have proven useful in many image applications such as image compression and deblurring. Combined with max pooling, they have led to state-of-the-art image recognition performance with a linear classifier (Yang et al., 2009; Boureau et al., 2010). However, the analysis developed for binary features in the previous section does not apply, and the underlying causes for this good performance seem to be different.

3.1. Influence of Pooling Cardinality

In the case of binary features, and when no smoothing is performed, we have seen above that there is an optimal pooling cardinality, which increases with the sparsity of the features. Smoothing the features displaces that optimum towards smaller cardinalities. In this section, we perform the same analysis for continuous features, and show that (1) it is always better to use all samples for max pooling when no smoothing is performed, (2) however the increase in signal-to-noise ratio (between means' separation and standard deviation) does not match the noise reduction obtained by averaging over all samples.

Let P denote cardinality of the pool. For a Gaussian distribution, a classical result is that the expectation of the max over samples from a distribution of variance σ^2 grows asymptotically (when $P \rightarrow \infty$) like $\sqrt{2\sigma^2 \log(P)}$. Thus, the separation of the maxima over two Gaussian samples increases indefinitely with sample size if their standard deviations are different, but the rate of growth is very slow.

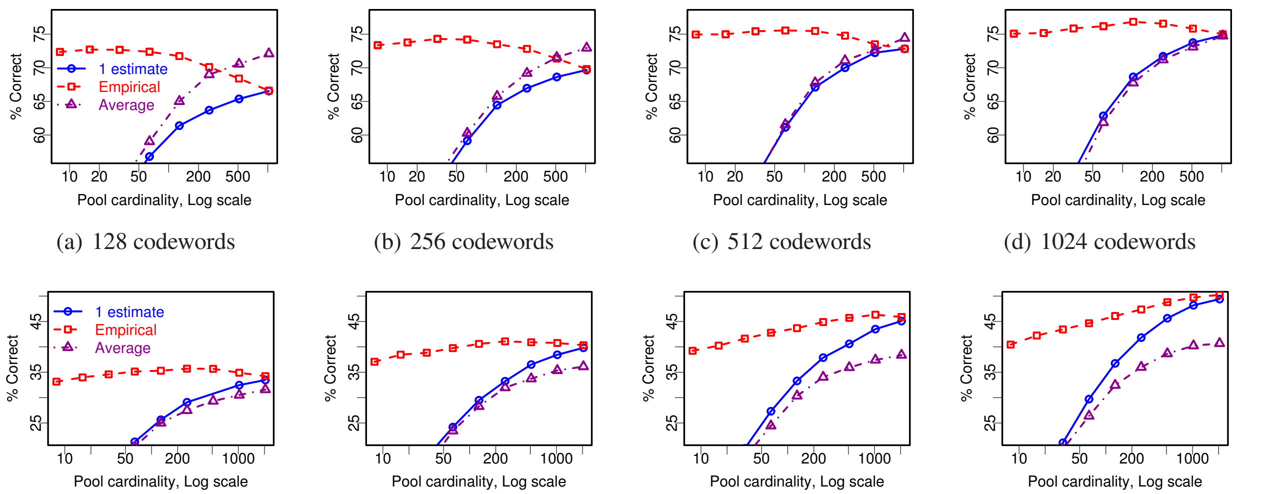
Exponential distribution (or Laplace distributions for feature values that may be negative) are often preferred to Gaussian distributions to model visual feature responses because they are highly kurtotic. In particular, they are a better model for sparse codes. Assume the distribution of the value of a feature for each patch is an exponential distribution with mean $\frac{1}{\lambda}$ and variance $\frac{1}{\lambda^2}$. The corresponding cumulative distribution function is $1 - e^{-\lambda x}$. The cumulative distribution function of the max-pooled feature is $(1 - e^{-\lambda x})^P$. The mean and variance of the distribution can be shown to be respectively $\mu_m = \frac{\mathcal{H}(P)}{\lambda}$ and $\sigma_m^2 = \frac{1}{\lambda^2} \sum_{l=1}^{P} \frac{1}{l}(2\mathcal{H}(l) - \mathcal{H}(P))$, where $\mathcal{H}(k) = \sum_{i=1}^{k} \frac{1}{i}$ denotes the harmonic series. Thus, for all P, $\frac{\mu_1}{\mu_2} = \frac{\sigma_1}{\sigma_2} = \frac{\lambda_1}{\lambda_2}$, and the distributions will be better separated if the scaling factor of the mean is bigger than the scaling factor of the standard deviations, i.e., $\mathcal{H}(P) > \sqrt{\sum_{l=1}^{P} \frac{1}{l}(2\mathcal{H}(l) - \mathcal{H}(P))}$, which is true for all P. Furthermore, since $\mathcal{H}(P) = \log(P) + \gamma + o(1)$ when $P \rightarrow \infty$ (where γ is Euler's constant), it can be shown that $\sum_{l=1}^{P} \frac{1}{l}(2\mathcal{H}(l) - \mathcal{H}(P)) = \log(P) + O(1)$, so that the distance between the means grows faster (like $\log(P)$) than the standard deviation, which grows like $\sqrt{\log(P)}$. Two conclusions can be drawn from this: (1) when no smoothing is performed, larger cardinalities provide a better signal-to-noise ratio, but (2) this ratio grows slower than when simply using the additional samples to smooth the estimate ($1/\sqrt{P}$ assuming independent samples, although in reality smoothing is less favorable since the independence assumption is clearly false in images).

3.2. Experiments

We perform the same experiments as in the previous section to test the influence of codebook size and pooling cardinalities, using continuous sparse codes instead of binary codes. Results are presented in Fig. 3. As expected from our analysis, using larger pooling cardinalities is always better with continuous codes when no smoothing is performed (blue solid curve): no bump is observed even with smaller dictionaries. Max pooling performs better than average pooling on the Caltech dataset (bottom row in Fig. 3); this is not predicted by the analysis using our very simple model. On the Scenes dataset (top row in Fig. 3), max pooling and average pooling perform equally well when the largest dictionary size tested (1024) is used. Slightly smoothing the estimate of max pooling by using a smaller sample cardinality results in a small improvement in performance; since the grid (or pyramid) pooling structure performs some smoothing (by providing several estimates for the sample cardinalities of the finer levels), this may explain why max pooling performs better than average pooling with grid and pyramid smoothing, even though average pooling may perform as well when a single estimate is given.

3.3. Mixture Distribution

Our simple model does not account for the better discrimination sometimes achieved by max pooling for continuous sparse codes with large dictionaries. In a previous paper (Boureau et al., 2010), we showed that max pooling may perform better than average pooling with exponential features sampled from mixture distributions, with one of the components of the mixture being shared between classes and having a lower expected activation. This may play a role in the better performance of max pooling. In fact, the sparse code vectors extracted on images have an overwhelming number of zero components, and may thus be better modeled as a mixture distribution of a Dirac delta function and an exponential distribution, than as a single exponential distribution. Assuming the mixture coefficients vary between images, the mean of the distribution computed over an image shifts between 0 and the mean of the exponential; this may result in a larger overlap between class-conditional distributions when using average pooling than max pooling, as illustrated in our work (Boureau et al., 2010).



(a) 128 codewords (b) 256 codewords (c) 512 codewords (d) 1024 codewords

(e) 128 codewords (f) 256 codewords (g) 512 codewords (h) 1024 codewords

Figure 3. Influence of pooling cardinality and smoothing on performance. Top row: Scenes dataset. Bottom row: Caltech-101 dataset. 1 estimate: maximum computed over a single pool. Empirical: empirical average of max-pooled features over several subsamples of smaller cardinality. Average: estimate of the average computed over a single pool. Best viewed in color.

4. Transition from Average to Max Pooling

The previous sections have shown that depending on the data and features, either max or average pooling may perform best. The optimal pooling type for a given classification problem may be neither max nor average pooling, but something in between; in fact, we have shown that it is often better to take the max over a fraction of all available feature points, rather than over the whole sample. This can be viewed as an intermediate position in a parametrization from average pooling to max pooling over a sample of fixed size, where the parameter is the number of feature points over which the max is computed: the expected value of the max computed over one feature is the average, while the max computed over the whole sample is obviously the real max.

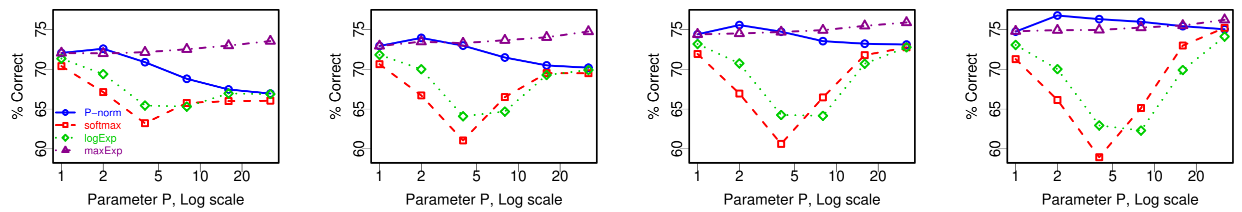
This is only one of several possible parametrizations that continuously transition from average to max pooling. The P-norm of a vector (more accurately, a version of it normalized by the number of samples N) is another well-known one: $f_P(\boldsymbol{v}) = \left(\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{v}_i^P\right)^{\frac{1}{P}}$, which gives the average for $P = 1$ and the max for $P \rightarrow \infty$. This parametrization accommodates square-root pooling (for $P = 2$) and absolute value pooling (for $P = 1$), that have also used in the literature (e.g. (Yang et al., 2009)).

A third parametrization is the sum of samples weighted by a softmax function: $\sum_i \frac{\exp(\beta \boldsymbol{x}_i)}{\sum_j \exp(\beta \boldsymbol{x}_j)} \boldsymbol{x}_i$. This gives average pooling for $\beta = 0$ and max pooling for $\beta \rightarrow \infty$. Finally, a fourth parametrization is $\frac{1}{\beta} \log \frac{1}{N} \sum_i \exp(\beta \boldsymbol{x}_i)$, which gives the average for $\beta \rightarrow 0$ and the max for $\beta \rightarrow \infty$. As with the P-norm, the result only depends on the empirical feature activation mean in the case of binary vectors; thus, these functions can be applied to an already obtained average pool.

Fig. 4 plots the recognition rate obtained on the Scenes dataset using sparse codes and each of the four parametrizations mentioned. Instead of using the expectation of the maximum for exponential distributions, we have used the expectation of the maximum of binary codes $(1-(1-\alpha)^P)$, applied to the average, as we have observed that it works well; we refer to this function as the expectation of the maximum (*maxExp* in Fig. 4), although it does not converge to the maximum when $P \rightarrow \infty$ for continuous codes. Both this parametrization and the P-norm perform better than the two other pooling functions tested, which present a marked dip in performance for intermediate values.

5. Discussion

This paper has looked more closely into the factors underlying the recognition performance of pooling operations. By carefully adjusting the pooling step of feature extraction, relatively simple systems of local features and classifiers can become competitive to more complex ones: our previous work (Boureau et al., 2010) had already demonstrated that it was possible to achieve similar levels of performance with a linear kernel as with Lazebnik et al. (2006)'s intersection kernel, using vector quantized binary codes. Here, we have investigated what properties of max pooling may account for this good performance, and shown that this pooling strategy was well adapted to



(a) 128 codewords (b) 256 codewords (c) 512 codewords (d) 1024 codewords

Figure 4. Recognition rate obtained using several pooling functions that perform a continuous transition from average to max pooling when varying parameter P (see text). Best viewed in color.

features with a low probability of activation. We have proposed several methods for further improving pooling: (1) use directly the formula for the expectation of the maximum to obtain a smoother estimate in the case of binary codes, (2) pool over smaller samples and take the average. When using sparse coding, some limited improvement may be obtained by pooling over subsamples of smaller cardinalities and averaging, and conducting a search for the optimal pooling cardinality, but this is not always the case. Further directions we envision include pooling across several feature types, and adapting pooling parameters separately for each feature. We hope that paying more attention to the choice of the pooling function will lead to better designed recognition architectures.

Acknowledgments. This work was funded in part by NSF grant EFRI/COPN-0835878 to NYU, and ONR contract N00014-09-1-0473 to NYU.

References

Boureau, Y, Bach, F, LeCun, Y, and Ponce, J. Learning mid-level features for recognition. In *CVPR*, 2010.

Bruckstein, A and Cover, T. Monotonicity of linear separability under translation. *IEEE TRANS. PATTERN ANAL. MACH. INTELLIG.*, 7(3):355–357, 1985.

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

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

Fukushima, K and Miyake, S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 1982.

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

Jarrett, K, Kavukcuoglu, K, Ranzato, M, and LeCun, Y. What is the best multi-stage architecture for object recognition? In *(ICCV'09)*. IEEE, 2009.

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

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

LeCun, Y, Boser, B, Denker, J. S, Henderson, D, Howard, R. E, Hubbard, W, and Jackel, L. D. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1989.

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

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

Pinto, N, Cox, D, and DiCarlo, J. Why is real-world visual object recognition hard. *PLoS Computational Biology*, 4 (1):151–156, 2008.

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

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

Sivic, J and Zisserman, A. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. URL http://www.robots.ox.ac.uk/~vgg.

Yang, J, Yu, K, Gong, Y, and Huang, T. Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. In *CVPR*, 2009.

Zhang, J, Marszalek, M, Lazebnik, S, and Schmid, C. Local features and kernels for classifcation of texture and object categories: An in-depth study. *Int. J. of Comp. Vision*, 73(2):213–238, 2007.