Histograms of Sparse Codes for Object Detection

Xiaofeng Ren*
Amazon.com
xiaofenr@amazon.com

Deva Ramanan
University of California, Irvine
dramanan@ics.uci.edu

Abstract

Object detection has seen huge progress in recent years, much thanks to the heavily-engineered Histograms of Oriented Gradients (HOG) features. Can we go beyond gradients and do better than HOG? We provide an affirmative answer by proposing and investigating a sparse representation for object detection, Histograms of Sparse Codes *(HSC). We compute sparse codes with dictionaries learned from data using K-SVD, and aggregate per-pixel sparse codes to form local histograms. We intentionally keep true to the sliding window framework (with mixtures and parts) and only change the underlying features. To keep training (and testing) efficient, we apply dimension reduction by computing SVD on learned models, and adopt supervised training where latent positions of roots and parts are given externally e.g. from a HOG-based detector. By learning and using local representations that are much more expressive than gradients, we demonstrate large improvements over the state of the art on the PASCAL benchmark for both root-only and part-based models.*

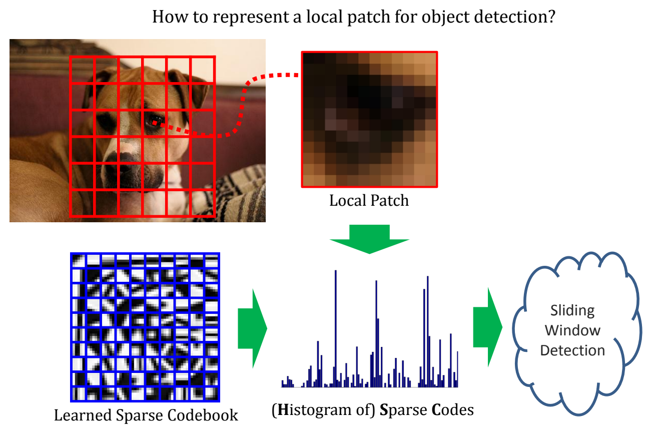

Figure 1: Can we find better features than HOG for object detection? We develop Histograms-of-Sparse-Codes (HSC), which represents local patches through learned sparse codes instead of gradients and outperforms HOG by a large margin in state-of-the-art sliding window detection.

1. Introduction

Object detection is a fundamental problem in computer vision and has been a major focus of research activities. There has been huge progress in object detection in recent years, much thanks to the celebrated Histograms of Oriented Gradients (HOG) features [8, 13]. The HOG features are the basis of the original Dalal-Triggs person detector [8], the popular Deformable Parts Model (DPM) [13], the Exemplar-SVM model [21], and pretty much every other modern object detector. HOG is also seeing increasing use in other domains such as pose estimation [34], face recognition [35], and scene classification [32].

The HOG features, heavily engineered for both accuracy and speed, are not without issues or limits. They are gradient-based and lack the ability to directly represent richer (and larger) patterns. There are multiple ad-hoc designs, such as 4-way normalization and 9 orientations, that are non-intuitive and unappealing. More importantly, such hand-crafted features are difficult to generalize or expand to novel domains (such as depth images or the time-domain), and they increasingly become a bottleneck as the Moore's Law drives up computational capabilities. There are evidences that local features are most crucial for detection [23], and we may already be saturating the capacity of HOG [36].

Can we *learn* representations that outperform a hand-engineered HOG? In the wake of recent advances in feature learning [16, 1] and its successes in many vision problems such as recognition [19] and grouping [26], it is promising to consider employing local features automatically learned from data. However, feature learning for detection is a challenging problem, which has seen only limited successes so far [7, 9], partly because the massive number of windows one needs to scan. One could also argue that HOG is already a high dimensional representation (for the entire object template), much higher than the number of typical positive training examples, and therefore it remains to be answered whether a richer, learned representation would fur-

*Work done while the author was at the Intel Science and Technology Center for Pervasive Computing, Intel Labs.

ther improve detection performance.

In this work, we show that indeed a local representation can be effectively learned for object detection, and the learned rich features outperform HOG by a large margin as demonstrated on the PASCAL and INRIA benchmarks. We compute per-pixel sparse codes using dictionaries learned through *K-SVD*, and aggregate them into "histograms" of sparse codes (HSC) in the spirit of HOG. For a fair comparison, we keep to the HOG-driven scanning window framework as much as possible, with identical settings for mixtures, parts, and training procedure. To enable efficient training (especially for part-based models), we use a supervised training strategy: instead of iterating over latent root and part locations in the semi-convex setting of DPM [13], we assume these locations are given and fixed (computed with a HOG-based detector). We also apply dimension reduction using learned models to effectively compress the high dimensional sparse code representations.

The resulting HSC-based object detectors perform above our expectations and go well beyond elaborate HOG-based systems. Using root-only models, we improve the mean average-precision on the 20 PASCAL2007 classes from 21.4% of HOG to 27.9% of HSC with identical settings. Using part-based models, we improve the mean AP from 30.1% to $\mathbf{35.2}\%$. In both cases, we also lead the widely used DPM system [14] by a considerable amount. We validate the benefits of richer representation through the use of increasingly large dictionary sizes and patch sizes. To the best of our knowledge, our work is the first to show that dictionary-based features can replace and significantly outperform HOG for general object detection.

2. Related Works

Object detection: Many contemporary approaches for object detection have converged on the paradigm of linear SVMs trained on HOG features [8, 13, 5, 21], as evidenced by benchmark evaluations such as PASCAL [12]. Most approaches have explored model structure, either through nonparametric mixtures or exemplars [21, 10], compositional grammar structure [15], supervised correspondences [5, 2], and low-dimensional projections [29, 25]. Alternative approaches explored the use of segmentation [20, 31]. Most, if not all such approaches have relied on fixed feature set of HOG descriptors. We focus on the underlying feature representation, and hence our work could in principle be applied to any of these models.

Image descriptors: Image descriptors for object recognition have long been studied and are usually hand-designed. A sampling of such descriptors include local binary patterns [17], integral channel features of gradients and color cues [11], RGB covariance features [28], and multi-scale spatial pyramids [4]. Such heterogeneous features are often combined by concatenation or through multiple kernels [30]. An alternative family of approaches directly learn thresholded pixel values, often selected through boosting [11] or randomized decision trees [22]. The closest to our approach is that of Dikmen *et al*. [9], which replaces HOG with a histogram of learned 3x3 filters with competitive performance on PASCAL. We extensively compare to this approach and find that we are able to learn much richer structures on larger patches through sparse coding and achieve substantial improvements over HOG.

Sparse features: Feature learning is an active field increasingly capturing the attention of researchers with their ability to utilize big data[16, 19]. Sparse coding is a popular way of learning feature representation [1, 24], commonly used in image classification settings [33, 6] but also explored for detection [18]. More recent uses of sparse coding are toward the pixel level, learning patch representations to replace SIFT features [3]. Such patch representations can be applied to other problems such as contour detection [26].

3. Feature Learning for Object Detection

Histograms of Oriented Gradients (HOG) are highly specialized features engineered for object detection, extremely popular and used in virtually every object detection system. The core of HOG is the representation of local patterns at every pixel using gradient orientations, originally developed for detecting people [8] and extended to contrast-sensitive gradients for general objects [13].

While HOG is very effective in capturing gradients, long known to be crucial for vision and robust to appearance and illumination changes, images are clearly more than just gradients. How to build a richer local representation that outperforms HOG is a key challenge for detection, which remains open despite efforts of designing features [17], learning them [9], or combining multiple features [30].

We seek to replace HOG with features automatically learned from data. A feature learning approach is attractive for its *scalability* and *adaptivity*: if we can effectively learn local features for detection, it would be relatively easy to expand such a feature set to higher dimensions and larger patches, and also adapt a detection system to specialized domains or novel sensor data such as RGB-D cameras.

In this section we will develop Histograms-of-Sparse-Codes (HSC), which resembles HOG but is based on well-developed sparse coding techniques that represent each local patch using a sparse set of codewords. The codewords (dictionary) are learned in an unsupervised way from data. Once per-pixel sparse codes are computed, we aggregate the codes into "histograms" on regular cells and use them to replace HOG in the standard Deformable Parts Model [13].

3.1. Local Representation via Sparse Coding

We use K-SVD [1] for dictionary learning, a standard unsupervised dictionary learning algorithm that generalizes

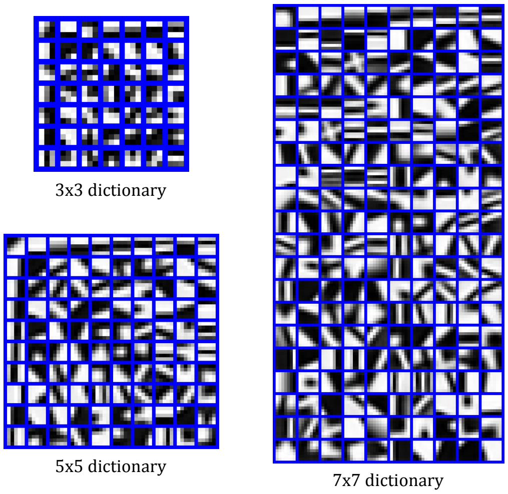

Figure 2: Dictionaries learned through K-SVD for three patch sizes. As patch size and dictionary size grow, increasingly complex patterns are represented in the dictionary. The ability to directly represent large, complex patterns gives us hope for outperforming HOG in detection.

K-means. Given a set of image patches $Y = [y_1, \cdots, y_n]$, K-SVD jointly finds a dictionary $D = [d_1, \cdots, d_m]$ and an associated sparse code matrix $X = [x_1, \cdots, x_n]$ by minimizing the reconstruction error

$$\min_{D,X} \|Y - DX\|_F^2 \ s.t. \ \forall i, \ \|x_i\|_0 \leq K \tag{1}$$

where x_i are the columns of X, the zero-norm $\|\cdot\|_0$ counts the non-zero entries in the sparse code x_i, and K is a predefined sparsity level. K-SVD solves this optimization by alternating between computing X and D. Given the dictionary D, computing the codes X can be efficiently solved using the greedy Orthogonal Matching Pursuit (OMP) [24]. Given the codes X, the dictionary D is updated sequentially by singular value decomposition. We subtract the mean from the patches in advance.

Once the dictionary D is learned, we again use Orthogonal Matching Pursuit to compute sparse codes at every pixel in an image pyramid. The batch version of the OMP algorithm [27] provides considerable speed-up by precomputing the inner products between patches and codewords.

Examples of the dictionaries learned are shown in Fig. 2. Comparing to the special-purpose algorithm developed in [9], K-SVD effectively learns common structures without any need for tweaking such as selecting sampling. As the patch size and dictionary size grow, more and more interesting structures are discovered (such as corners, thin lines, line endings, and high-frequency gratings).

3.2. Aggregation into Histograms of Sparse Codes

The sliding window framework of object detection divides an image into regular cells (8x8 pixels) and computes a feature vector of each cell, to be used in a convolution-based window scanning. We keep as close as possible to HOG for aggregating per-pixel sparse codes.

Let X be the sparse code computed at a pixel, whose dimension equals the dictionary size. For each non-zero entry x_i in X, we use soft binning (bilinear interpolation as in HOG, which we find is slightly better than hard binning) to assign its absolute value $|x_i|$ to one of the four spatially-surrounding cells. The result is a (semi-)dense feature vector F on each cell averaging codes in a 16x16 neighborhood, which we call *Histograms of Sparse Codes* (HSC). We normalize F with its $L2$ norm. Finally, we apply a power transform on each element of F

$$\bar{F} = F^\alpha \tag{2}$$

as is sometimes done in recognition settings [26]. The power transform makes the distribution of F's values more uniform and increases the discriminative power of F.

For general object detection in the PASCAL setting, we find that only using $|x_i|$ is not enough. Just as in the use of contrast-sensitive gradients in HOG, and the cosine and absolute cosine metrics in [9], we need signed values of x_i to differentiate white-on-black and black-on-white patterns. We add two half-wave rectified values before feeding them into bilinear aggregation. That is, each codeword i in the dictionary now has three values in the HSC:

$$[\ |x_i|, \ \max(x_i, 0), \ \max(-x_i, 0)\ ] \tag{3}$$

It is worth noting that there are very few ad-hoc design choices in these HSC features. We can do away with several engineering designs in HOG, such as 4-way normalization, truncation of gradient energy, and the asymmetry of horizontal/vertical directions from using 9 orientation bins. This illustrates the power of learning richer features on larger patches, which captures more information than gradients and has less need for manually designed transforms. Moreover, it is straightforward to change the settings, such as dictionary size, patch size or sparsity level, allowing the HSC features to adapt to the needs of different problems.

In Fig. 3 we visualize the HSC features using dominant codewords, and compare them to HOG. HSC features capture oriented edges using learned patterns, and can better localize them in each cell (the edges can be off-center). Moreover, HSC features can represent richer patterns such as corners (the girl's feet) or parallel lines (both horizontal and vertical in the negative image). While these two images may be confusing in the HOG space, an HSC-based model has no trouble telling them apart, as shown in the responses to a standard linear SVM trained on INRIA.

3.3. Supervised Training of Part Models

We intentionally only change the underlying local features and keep everything else identical in our own imple-

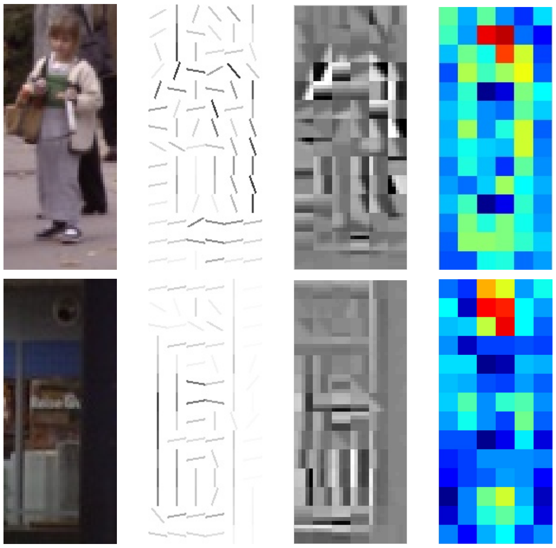

Figure 3: Visualizing HSC vs HOG: (a) image; (b) dominant orientation in HOG, weighted by gradient magnitude; (c) dominant codeword in HSC, weighted by histogram value; (d) per-cell responses of HSC features when multiplied with a linear SVM model trained on INRIA (colors are on the same scale).

mentation of the standard sliding window detection framework, following the DPM model of [13]. Let $p_i = (x_i, y_i, s_i)$ be the position and scale of part i, $p = \{p_i : i \in V\}$ be the placement of all parts, and let m be the mixture assignment of an image window I. The score of a deformable-part object detector $S(I, p, m)$ is

$$\sum_{i \in V} w_i^m \phi(x, p_i) + \sum_{ij \in E} w_{ij}^m \psi(p_i, p_j) + b_m \tag{4}$$

The graph $G = (V, E)$ specifies the connectivity of parts, a star graph connecting all parts to the root. $\phi(I, p_i)$ are the local features, where we exchange HOG for HSC. $\psi(p_i, p_j)$ is the deformation cost constraining part locations. The solution maximizing $S(I, p, m)$ can be computed using dynamic programming as in [13]. The computational cost is linear in the feature dimension of $\phi(I)$.

A lot of parameters need to be learned in the detector above, including appearance filters $\{w_i^m\}$, spring parameters $\{w_{ij}^m\}$, and mixtures biases $\{b_m\}$. The standard way of training the model is the latent SVM approach in [13]. The main challenge for learning is that many things are unknown about positive examples: part location, mixture assignment, and to some extent root location (due to imprecise bounding boxes). The learning procedure needs to iterate over training the model and assigning latent variables in the positive images, resulting in an elaborate and slow process, sometimes fragile due to the non-convex nature of the formulation. This poses a major challenge for training with appearance features that are more expressive, higher dimensional, and possibly redundant (as in our case).

We circumvent all the issues with non-convex learning by resorting to *supervised training*, assuming that everything is *known* about positive images, given by an external source. Injecting supervision has been a trend in detection, such as in poselets [5] or the HOG-based face detector [35]. For general object detection, it is difficult to obtain extensive human labels, and we instead use the state-of-the-art HOG-based detection system [14], where the outputs of their final detectors are used as "groundtruth". By fixing the latent variables in the part-based model, we make a fair and direct comparison of detection using HSC vs HOG features.

With latent variables fixed, learning the detection model can be defined as a convex quadratic program

$$\begin{aligned} &\operatorname*{argmin}_{\beta, \xi_n \geq 0} \quad \frac{1}{2}\beta \cdot \beta + C \sum_n \xi_n \\ \text{s.t.} \quad &\forall n \in \text{pos} \quad \beta \cdot \Phi(I_n, z_n) \geq 1 - \xi_n \\ &\forall n \in \text{neg}, \forall z \quad \beta \cdot \Phi(I_n, z) \leq -1 + \xi_n \end{aligned} \tag{5}$$

with slack penalties ξ_n. We use the dual-coordinate solver of [34], which in practice needs a single iteration over negative training images to converge. This allows us to train our supervised models much faster than the latent hard-negative mining approach of [14], making it feasible to work with high dimensional appearance features in part-based models.

3.4. Dimension Reduction using Learned Models

For root-only experiments on PASCAL, we use a dictionary of 100 codes over 5x5 patches, resulting in a 300-dimensional feature vector, an order of magnitude higher than HOG. We find it convenient to reduce the dimension down when training full part-based models. However, unsupervised dimension reduction, such as principal component analysis (PCA) on the data, tends not to work well for either gradient features or sparse codes. One way of doing proper dimension reduction in the SVM setting would be to consider joint optimization such as in the bilinear model of [25], but it requires an expensive iterative algorithm.

We find a simple way of doing supervised dimension reduction making use of models we have learned for the root-only case. Let us write each learned filter w_i^m as an $N \times n_f$ matrix W_i^m, where $N = n_x n_y$ (the number of spatial cells in a part filter) and n_f is the size of our HSC feature F. We wish to factor each filter into a low-rank representation:

$$W_i^m \approx C_i^m B \quad \text{where} \quad C_i^m \in \mathcal{R}^{N \times P}, B \in \mathcal{R}^{P \times n_F} \tag{6}$$

where B is analogous to a PCA-basis that projects F to a smaller dimension set $P \ll n_f$ and C_i^m is the appearance filter in this reduced space. We can simultaneously learn a good subspace for all filters of all classes by computing the SVD of the concatenated set of matrices, obtaining a universal projection matrix B that captures the essence (and removes the redundancy) in the HSC features. We integrate B into feature computation such that it is transparent to the rest of the system, making training (and testing) part-based models much faster without sacrificing much accuracy.

4. Experiments

We use both the INRIA Person Dataset [8] and the PASCAL2007 challenge dataset [12] for validating our Histograms-of-Sparse-Codes (HSC) features and extensively compare to HOG in identical settings. For INRIA, we use root-only models and evaluate the HSC settings such as dictionary size, sparsity level, patch size, and power transform. For PASCAL2007, we use both root-only and part-based models with supervised training, measure the improvements of HSC over HOG for the 20 classes, and compare to the state-of-the-art DPM system [14] which uses the same model but with additional tweaks (such as symmetry).

4.1. INRIA Person Dataset

The INRIA Person Dataset consists of 1208 positive training images (and their reflections) of standing people, cropped and normalized to 64x128, as well as 1218 negative images and 741 test images. This dataset is an ideal setting for studying local features and comparing to HOG, as it is what HOG was designed and optimized for, and training is straightforward (there is no need for mixture or latent positions for positive examples). The dual solver requires less than two passes over the negatives. The baseline average precision (AP) of our system using HOG is 80.2%.

Sparsity level and dictionary size. Do we need a sparsity level $K>1$? This is an intriguing question and illustrates the difference between reconstructing signals (what sparse coding techniques are designed for) and extracting meaningful structures for recognition. Fig. 4(a) shows the average precision on INRIA when we change the sparsity level along with the dictionary size using 5x5 patches. We observe that when the dictionary size is small, a patch cannot be well represented with a single codeword, and $K > 1$ (at least 2) seems to help. However, when the dictionary size grows and includes more structures in its codes, the $K = 1$ curve catches up, and performs very well. Therefore we use $K = 1$ in all the following experiments, which makes the HSC features behave indeed like histograms using a sparse code dictionary.

Patch size and dictionary size. Next we investigate whether our HSC features can capture richer structures using larger patches. Fig. 4(b) shows the average precision as we change both the patch size and the dictionary size. It is encouraging to see that indeed the average precision greatly increases as we use larger patches (along with larger dictionary size). While 3x3 codes barely show an edge over HOG, 5x5 and 7x7 codes work much better, and the trend continues beyond 200 codewords. 9x9 patches, however, may be too large for our setting and do not perform well.

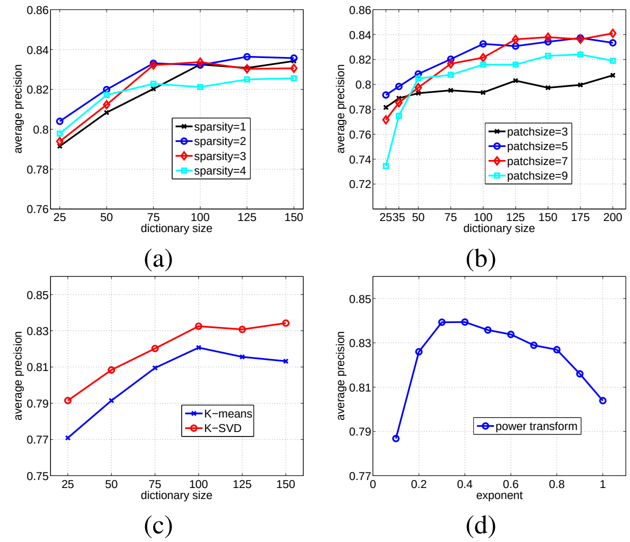

Figure 4: Investigating the use of sparse codes on INRIA. (a) Average precision (AP) of sparsity level vs dictionary size; sparsity=1 works well when the dictionary is large. (b) Patch size vs dictionary size; larger patches do code richer information but requires larger dictionaries. (c) Dictionary learning with K-SVD works better than K-means. (d) Power transform significantly improves the discriminative power of the sparse code histograms.

The ability to code and make use of larger patches shows the merits of our feature design and K-SVD learning comparing to the spherical k-medoids clustering in [9], which had considerable trouble with larger patches and observed decreases in accuracy going beyond the small size 3x3.

K-SVD vs K-means. With $K = 1$, one can also use K-means to learn a dictionary (after normalizing the magnitude of each patch). Fig. 4(c) compares the detection accuracy with K-SVD vs K-means dictionaries on 5x5 patches. K-SVD dictionaries have a clear advantage over K-means, probably because the reconstruction coefficient in sparse coding allows for a single codeword to model more appearances including the change of sign.

Power transform. Fig. 4(d) shows the use of power transform (Eq. 2) on the sparse features with varying exponent. Power transform does make a crucial difference, and an exponent around 0.3 performs the best, consistent with findings from other recognition contexts. We use 0.25.

Final results with root-only models. In Table 1 we show the average precision of our root-only models on the INRIA dataset comparing to the DPM system [14] (with parts, without context rescoring). We use dictionary size 100 for 3x3 patches, 150 for 5x5, and 300 for 7x7. Our results are competitive with the state of the art, while only

HOG	HSC_{3x3}	HSC_{5x5}	HSC_{7x7}	[14]
80.2%	80.7%	84.0%	84.9%	84.9%

Table 1: Average precision of HSC vs HOG (root-only) on the INRIA dataset. HSC-based detectors outperform HOG, especially with larger patch sizes, and are competitive with the state-of-the-art DPM system (with parts).

using a single root filter with no parts.

4.2. PASCAL2007 Benchmark

The PASCAL2007 dataset (comp3) includes 20 object classes in a total of 9963 images, widely used as the standard benchmark for general object detection. There are large variations across classes in terms of the consistency of shape, appearance, viewpoint or occlusion. We use the trainval positive images and the train negative images, and evaluate on the test images. For supervised training, we use the reported part locations of the voc-release4 system [14].

Final results with root-only models. We use a K-SVD dictionary of size 100 over 5x5 patches. With the expansion to to half-wave rectified codes, the feature dimension is 300. Our system does not handle the symmetry of filters explicitly, instead we flip the positive images and double the size of the training pool. We need 6 root filters to match a mixture of 3 filters from the DPM system.

Table 2(a) shows the average precision evaluation of our root-only models comparing the HSC features with HOG. The results are heartening: under identical settings, the HSC features improve AP by a large margin across the board, over 8% for many classes, and achieve a mean AP of **27.9%** over 21.4%. The improvement is also universal: HSC do better than HOG on 19 out of the 20 classes. Our results also outperform the state-of-the-art DPM system [14] with root-only models (fully trained on all trainval images.).

Fig. 6 shows some examples of the objects detected using HSC comparing to HOG. In general, we observe that HSC features help detect objects under challenging conditions and tend to avoid "silly" mistakes such as finding cats in a blue sky. Qualitatively, HSC-based detection produces results quite different from those of HOG, suggesting that there may be room for improvement by combining the strengths of both worlds.

Supervised dimension reduction. As described in Section 3.4, we learn a projection of HSC features to a lower dimension (universally applied to all cells) by utilizing models learned in the root-only case, and integrate it into feature extraction. Fig. 5 compares root-only models using model-based SVD with standard unsupervised SVD (computing SVD on the HSC features), on PASCAL as well as INRIA. For PASCAL, we select four classes (bus, cat, diningtable, motorbike). The results clearly show the advantage of our model-based dimension reduction. It is more effective on

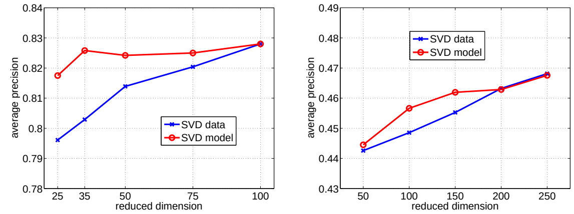

Figure 5: Comparing the effectiveness of dimension reduction: SVD-data is the standard way of unsupervised dimension reduction computing SVD on data; SVD-model computes SVD on learned root filters.

INRIA, suggesting that detecting person requires lower dimensional features than general objects. We use feature dimension 100 (reduced from 300) for our part-based models.

Final results with part-based models. Table 2 (b) shows the average precision of HSC vs HOG for part-based models. As in the root-only case, using HSC features leads to large improvements across the board, improving mAP from 30.1% to **35.2%**. It is also consistent: HSC improves over HOG on 19 classes, over 6% in many cases, and only does slightly worse on 1 classes (within margin of error). Here, HOG refers to our in-house implementation of our supervised part-based model (5). Our results also compare favorably to the state-of-the-art DPM system [14][1], improving 18 out of 20 classes. Not surprisingly, we see large improvements on the challenging classes that have a low baseline, such as bottle, cat, and diningtable.

Caching feature pyramids. To facilitate efficient training of multiple classes as in PASCAL, we precompute the feature pyramids and cache them. We find that it is sufficient to store each feature value in a single byte (comparing to 8 in a double), scaled to be between 0 and 1. The HSC features are within this range with a near uniform distribution after power transform. Single-byte caching not only makes the training process faster, but also suggests that there is high redundancy in the feature values and there likely exists much faster ways of computing them.

Running time. For a 300x300 image, single-scale HSC computation takes ~110ms on an Intel 3930k (single-core); for a typical scale pyramid of 40 levels, it takes ~4 seconds. This is slower than HOG (understandably) but manageable. Once features are computed, the computational cost is that of DPM, linearly scaling with feature dimension. For our PASCAL models with 6 mixture components and 8 parts, total test time is ~9 seconds. As for training, we only need to go through negative images once using the supervised approach, which takes about a day per class (single-core), comparable to that of DPM. A smaller set of negative images would speed up training without losing much accuracy.

[1]The average precisions of [14] are lower than reported on the authors' website, mainly because we exclude bounding box reprediction.

	aero	bike	bird	boat	bttl	bus	car	cat	chair	cow	table	dog	hors	mbik	prsn	plnt	shep	sofa	train	tv	$\overline{\text{avg}}$
HOG	20.5	47.7	9.2	11.3	18.3	35.4	40.8	4.0	12.2	23.4	11.2	2.6	41.0	30.3	21.0	6.6	11.8	16.0	31.5	32.5	21.4
HSC	26.0	50.9	7.5	14.6	24.0	46.0	48.6	12.8	17.6	30.1	16.8	11.3	44.1	41.3	31.3	10.6	20.6	28.3	35.5	40.7	27.9
Δ_{HSC}	+5.5	+3.2	-1.7	+3.3	+5.7	+10.6	+7.8	+8.8	+5.4	+6.7	+5.6	+8.7	+3.1	+11.0	+10.3	+4.0	+8.8	+12.3	+4.0	+8.2	+6.5
[14]	25.2	50.2	5.8	11.8	17.2	41.4	43.6	3.5	15.9	21.0	15.6	7.9	44.1	34.8	30.3	9.9	14.6	18.4	36.4	33.7	24.1

(a) Root-only models: HOG, HSC, their difference Δ_{HSC} (HSC-HOG); and DPM [14]

	aero	bike	bird	boat	bttl	bus	car	cat	chair	cow	table	dog	hors	mbik	prsn	plnt	shep	sofa	train	tv	$\overline{\text{avg}}$
HOG	30.3	56.4	9.7	15.6	23.2	49.1	51.1	14.9	19.6	21.6	19.6	10.7	56.0	47.3	40.0	12.8	16.7	27.9	41.0	39.5	30.1
HSC	32.7	59.4	12.4	14.8	30.4	52.1	55.2	23.6	22.8	27.1	35.7	15.4	58.5	51.4	41.3	14.1	24.6	38.4	47.3	47.3	35.2
Δ_{HSC}	+2.4	+3.0	+2.7	-0.8	+7.2	+3.0	+4.1	+8.7	+3.2	+5.5	+16.1	+4.7	+2.5	+4.1	+1.3	+1.3	+7.9	+10.5	+6.3	+7.8	+5.1
[14]	30.7	58.9	10.4	14.4	24.8	49.0	54.1	11.1	20.6	25.3	25.2	11.0	58.5	48.4	41.3	12.1	15.5	34.4	43.4	39.0	31.4

(a) Part-based models, with dimension reduction

Table 2: Results on the PASCAL2007 dataset. HSC and HOG results are from our supervised training system using identical settings and directly comparable. We achieve improvements over virtually all classes, in many cases by a large margin. (The results have been updated after a bug was discovered in the code – mAP is 1% higher than previously calculated).

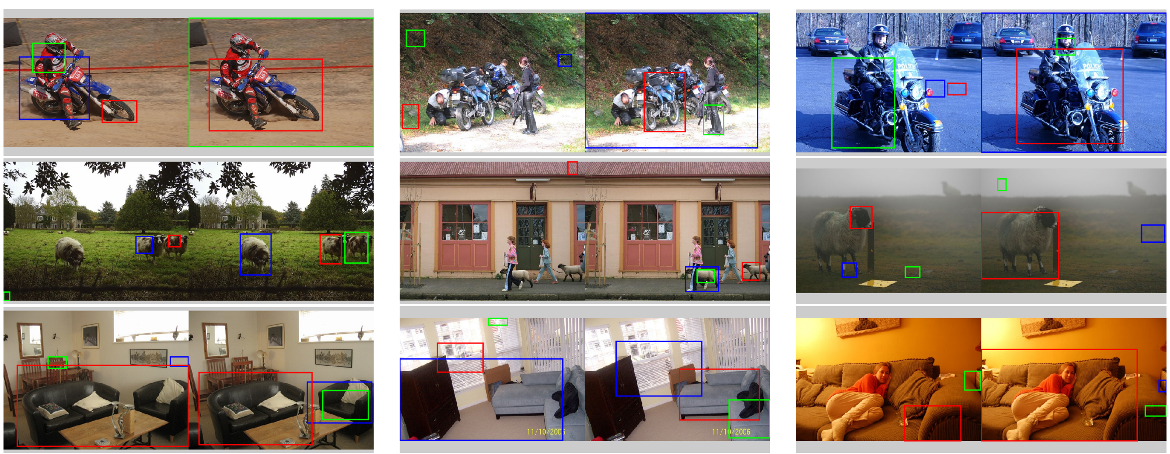

Figure 6: A few examples of HOG (left) vs HSC (right) based detection (root-only), showing top three candidates (in the order of red, green, blue). HSC behaves differently than HOG and tends to have different modes of success (and failure).

5. Discussions

In this work we demonstrated that dictionary based features, learned from data unsupervisedly, can replace and outperform the hand-crafted HOG features for general object detection. The detection problem is long thought to be a challenging case for feature learning, with millions of windows to consider. Through effective codebook learning, streamlined feature design and efficient training, we successfully showed how to build and use Histograms-of-Sparse-Codes (HSC) features in the spirit of HOG, which are capable of representing rich structures beyond gradients and lead to large improvements on virtually all classes on the PASCAL benchmark.

Our work is the first to clearly demonstrate the advantages of feature learning for general object detection, which come at a reasonable computational cost. Our studies show that large structures in large patches, when captured in a large dictionary, generally improve object detection, calling for future work on designing and learning even richer features. The sparse representation we use in the current HSC features are simple relative to what exits in the feature learning literature. There are a variety of more sophisticated schemes for coding, pooling and codebook learning that could potentially boost detection performance, and we believe this is a crucial direction toward solving the challenging detection problem under real-world conditions.

Acknowledgements: DR was funded by NSF Grant 0954083 and ONR-MURI Grant N00014-10-1-0933, and Intel Science and Technology Center - Visual Computing.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006. 1, 2

[2] H. Azizpour and I. Laptev. Object detection using strongly-supervised deformable part models. In *ECCV*, 2012. 2

[3] L. Bo, X. Ren, and D. Fox. Hierarchical Matching Pursuit for Image Classification: Architecture and Fast Algorithms.

In *Advances in Neural Information Processing Systems 24*, 2011. 2

[4] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. ACM, 2007. 2

[5] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, pages 1365–1372. IEEE, 2009. 2, 4

[6] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 2

[7] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. Wu, and A. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR)*, pages 440–445, 2011. 1

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I:886–893, 2005. 1, 2, 5

[9] M. Dikmen, D. Hoiem, and T. S. Huang. A data-driven method for feature transformation. In *CVPR*. IEEE, 2012. 1, 2, 3, 5

[10] S. Divvala, A. Efros, and M. Hebert. How important are deformable parts in the deformable parts model? In *ECCV Workshop on Parts and Attributes*, 2012. 2

[11] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *British Machine Vision Conference*, pages 1–11, 2009. 2

[12] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 2, 5

[13] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 32(9):1627–1645, 2010. 1, 2, 4

[14] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/. 2, 4, 5, 6, 7

[15] R. Girshick, P. Felzenszwalb, and D. McAllester. Object detection with grammar models. *Advances in Neural Information Processing Systems 24*, 2011. 2

[16] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 1, 2

[17] S. Hussain, W. Triggs, et al. Feature sets and dimensionality reduction for visual object detection. In *British Machine Vision Conference*, 2010. 2

[18] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In *Advances in Neural Information Processing Systems 23*, pages 1090–1098, 2010. 2

[19] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 2012. 1, 2

[20] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 17–32, 2004. 2

[21] T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, pages 89–96. IEEE, 2011. 1, 2

[22] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):448–461, 2010. 2

[23] D. Parikh and C. Zitnick. Finding the weakest link in person detectors. In *CVPR*, pages 1425–1432. IEEE, 2011. 1

[24] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. In *The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993. 2, 3

[25] H. Pirsiavash, D. Ramanan, and C. Fowlkes. Bilinear classifiers for visual recognition. *Advances in Neural Information Processing Systems 22*, 1(2), 2009. 2, 4

[26] X. Ren and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *Advances in Neural Information Processing Systems 25*, 2012. 1, 2, 3

[27] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient Implementation of the K-SVD Algorithm using Batch Orthogonal Matching Pursuit. Technical report, CS Technion, 2008. 3

[28] W. Schwartz, A. Kembhavi, D. Harwood, and L. Davis. Human detection using partial least squares analysis. In *ICCV*, pages 24–31. IEEE, 2009. 2

[29] H. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, and T. Darrell. Sparselet models for efficient multiclass object detection. In *ECCV*, 2012. 2

[30] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, pages 606–613. IEEE, 2009. 2

[31] S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. In *CVPR*, pages 1401–1408, 2011. 2

[32] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010. 1

[33] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009. 2

[34] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, pages 1385–1392. IEEE, 2011. 1, 4

[35] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, pages 2879–2886. IEEE, 2012. 1, 4

[36] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes. Do we need more training data or better models for object detection? In *BMVC*, 2012. 1