Multiclass Object Recognition with Sparse, Localized Features

Jim Mutch and David G. Lowe
Department of Computer Science
University of British Columbia
#201 - 2366 Main Mall
Vancouver, B.C., Canada, V6T 1Z4
{mutch,lowe}@cs.ubc.ca

Abstract

We apply a biologically inspired model of visual object recognition to the multiclass object categorization problem. Our model modifies that of Serre, Wolf, and Poggio. As in that work, we first apply Gabor filters at all positions and scales; feature complexity and position/scale invariance are then built up by alternating template matching and max pooling operations. We refine the approach in several biologically plausible ways, using simple versions of sparsification and lateral inhibition. We demonstrate the value of retaining some position and scale information above the intermediate feature level. Using feature selection we arrive at a model that performs better with fewer features. Our final model is tested on the Caltech 101 object categories and the UIUC car localization task, in both cases achieving state-of-the-art performance. The results strengthen the case for using this class of model in computer vision.

1. Introduction

The problem of recognizing multiple object classes in natural images has proven to be a difficult challenge for computer vision. Given the vastly superior performance of human vision on this task, it is reasonable to look to biology for inspiration. In fact, recent work by Serre, Wolf, and Poggio [23] has shown that a computational model based on our knowledge of visual cortex can be competitive with the best existing computer vision systems on some of the standard recognition datasets. Our paper builds on their approach by incorporating some additional biologically-motivated properties, including sparsification of features, lateral inhibition, and feature localization. We show that these modifications further improve recognition performance, strengthening our understanding of the computational constraints facing both biological and computer vision systems.

Within machine learning, it has been found that increasing the sparsity of basis functions [7, 14] (equivalent to reducing the capacity of the classifier) plays an important role in improving generalization performance. Similarly, within computational neuroscience, it has been found that adding a sparsity constraint is critical for learning biologically plausible models from the statistics of natural images [20]. For object class recognition, one way we have found to increase sparsity is to use a lateral inhibition model that eliminates weaker responses that disagree with the locally dominant ones. We further enhance this approach by matching only the dominant orientation at each position within a feature rather than comparing all orientation responses. We also increase sparsity during final classification by discarding features with low weights and using only those that have been found most effective. We show that each of these changes provides a significant boost in generalization performance.

While some current successful methods for object class recognition learn and apply quite precise geometric constraints on feature locations [6, 3], others ignore geometry and use a "bag of features" approach that ignores the locations of individual features [4]. According to models of object recognition in cortex [21], the brain uses a hierarchical approach, in which simple, low-level features having high position and scale specificity are pooled and combined into more complex, higher-level features having greater location invariance. We investigate retaining some degree of position and scale sensitivity at a higher point in this hierarchy than the approach of [23], and show that this provides a significant improvement in final classification performance.

We test these improvements on the large Caltech dataset of images from 101 object classes [5]. Our results show that there are significant improvements to classification performance from each of the changes. Further tests on the UIUC car database [1] demonstrate that the resulting system can also perform well on object detection and localization. Our results further strengthen the case for incorporating concepts from biological vision into the design of computer vision systems.

Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)
0-7695-2597-0/06 $20.00 © 2006 IEEE

2. Models

The model presented in this paper is based on the "standard model" of object recognition in cortex (as summarized by [21]), which focuses on the capabilities of the ventral visual pathway in an "immediate recognition" mode, independent of attention or other top-down effects. The rapid performance of the human visual system in this mode implies mainly feedforward processing, making it the easiest to model.

2.1. Previous models

Our model builds on that of Serre et al. [23], which in turn extends the "HMAX" model of Riesenhuber and Poggio [21]. These are the latest of a group of models which can be said to implement parts of the standard model, including convolutional networks [16] and Neocognitrons [10]. All start with an image layer of grayscale pixels and successively compute higher layers, alternating "S" and "C" layers (named by analogy with the V1 simple and complex cells discovered by Hubel and Wiesel [13]).

- Simple ("S") layers use convolution with local filters to compute higher-order features by combining different types of units in the previous layer.
- Complex ("C") layers increase invariance by pooling units of the same type in the previous layer over limited ranges. At the same time, the number of units is reduced by subsampling.

Recent models have moved towards greater quantitative fidelity to the ventral stream. HMAX was designed to account for the tuning and invariance properties [18] of neurons in IT cortex. Rather than attempting to learn its bottom-level ("S1") features, HMAX uses hardwired filters designed to emulate V1 simple cells. Subsequent "C" layers are computed using a hard MAX – a C unit's output is the maximum value of its afferent S units. This increases feature invariance while maintaining specificity. HMAX is also explicitly multiscale: its bottom-level filters are computed at all scales, and subsequent C units pool over both position and scale.

Serre et al. [23] introduced learning of intermediate-level shared features, made additional quantitative adjustments, and added a final (non-biologically motivated) SVM classifier to make the model useful for classification.

2.2. Our base model

Our base model is similar to [23] and performs about as well. Nevertheless, it is an independent implementation, and we give its complete description here. Its differences from [23] will be listed briefly at the end of this section. Larger changes, representing the main contribution of this paper, are described in section 2.3.

The model consists of five layers: an initial image layer and four subsequent layers, each layer built from the previous by alternating template matching and max pooling operations. It is shown graphically in figure 1, and the following subsections describe each layer.

Image layer. We convert the image to grayscale and scale the shorter edge to 140 pixels while maintaining the aspect ratio. Next we create an image pyramid of 10 scales, each a factor of $2^{1/4}$ smaller than the last (using bicubic interpolation).

Gabor filter (S1) layer. The S1 layer is computed from the image layer by centering 2D Gabor filters with a full range of orientations at each possible position and scale. Our base model follows [23] and uses 4 orientations. Where the image layer is a 3D pyramid of pixels, the S1 layer is a 4D structure, having the same 3D pyramid shape, but with multiple oriented units at each position and scale (see figure 1). Each unit represents the activation of a particular Gabor filter centered at that position/scale. This layer corresponds to V1 simple cells.

The Gabor filters are 11x11 in size, and can be described by:

$$G(x,y) = \exp\left(-\frac{(X^2+\gamma^2Y^2)}{2\sigma^2}\right)\cos\left(\frac{2\pi}{\lambda}X\right) \qquad (1)$$

where $X = x\cos\theta - y\sin\theta$ and $Y = x\sin\theta + y\cos\theta$. x and y vary between -5 and 5, and θ varies between 0 and π. The parameters γ (aspect ratio), σ (effective width), and λ (wavelength) are all taken from [23] and are set to 0.3, 4.5, and 5.6 respectively. Finally, the components of each filter are normalized so that their mean is 0 and the sum of their squares is 1. We use the same size filters for all scales (applying them to scaled versions of the image). The response of a patch of pixels X to a particular S1 filter G is given by:

$$R(X,G) = \left|\frac{\sum X_i G_i}{\sqrt{\sum X_i^2}}\right| \qquad (2)$$

Local invariance (C1) layer. This layer pools nearby S1 units (of the same orientation) to create position and scale invariance over larger local regions, and as a result can also subsample S1 to reduce the number of units. For each orientation, the S1 pyramid is convolved with a 3D max filter, 10x10 units across in position[1] and 2 units deep in scale. A C1 unit's value is simply the value of the maximum S1 unit (of that orientation) that falls within the max filter. To achieve subsampling, the max filter is moved around the S1 pyramid in steps of 5 in position (but only 1 in scale), giving a sampling overlap factor of 2 in

[1] Note that the max filter is itself a pyramid, so its size is 10x10 only at the lowest scale.

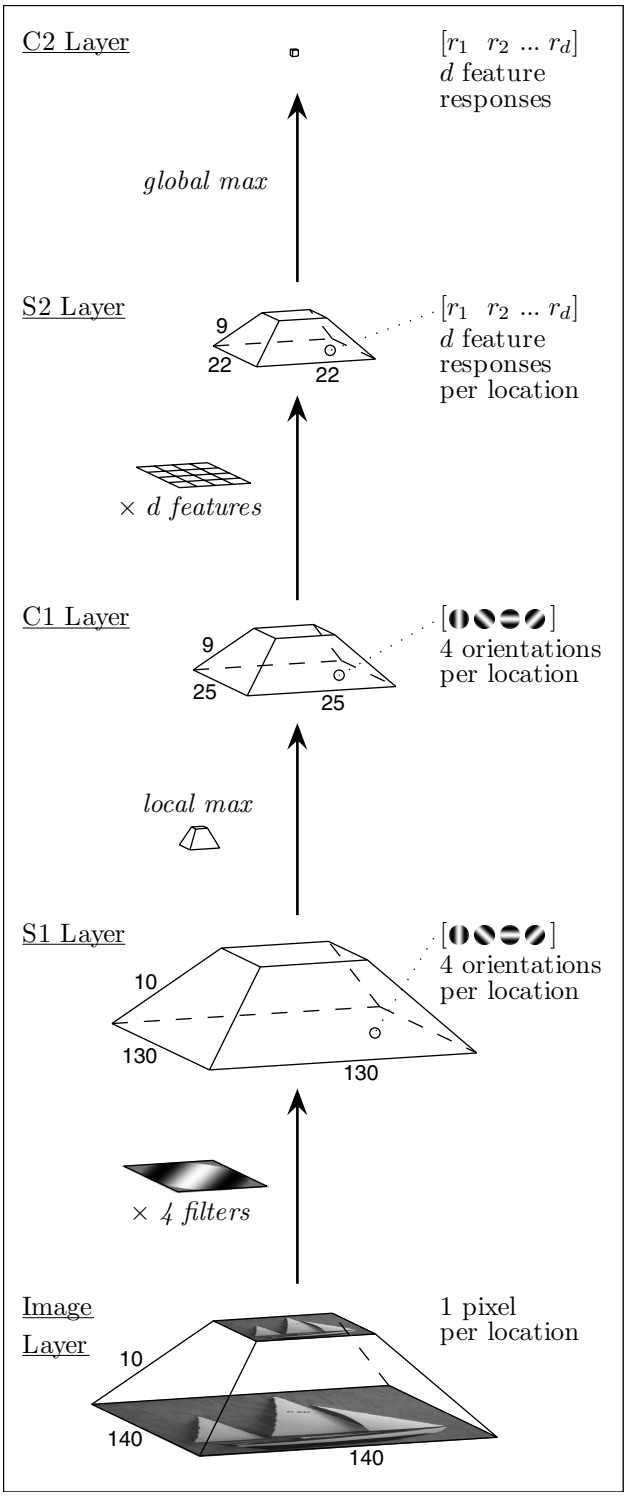

Figure 1. Overview of the base model. Each layer has units covering three spatial dimensions (x/y/scale), and at each 3D location, an additional dimension of *feature type*. The image layer has only one type (pixels), layers S1 and C1 have 4 types, and the upper layers have d (many) types per location. Each layer is computed from the previous via convolution with template matching or max pooling filters. Image size can vary and is shown for illustration.

both position and scale. Due to the pyramidal structure of S1, we are able to use the same size filter for all scales. The resulting C1 layer is smaller in spatial extent and has the same number of feature types (orientations) as S1; see figure 1. This layer provides a model for V1 complex cells.

Intermediate feature (S2) layer. At every position and scale in the C1 layer, we perform template matches between the patch of C1 units centered at that position/scale and each of d prototype patches. These prototype patches represent the intermediate-level features of the model.

The prototypes themselves are randomly sampled from the C1 layers of the training images in an initial feature-learning stage. (For the Caltech 101 dataset, we use d = 4,075 for comparison with [23].) Prototype patches are like fuzzy templates, consisting of a grid of simpler features that are all slightly position and scale invariant.

During the feature learning stage, sampling is performed by centering a patch of size 4x4, 8x8, 12x12, or 16x16 (x 1 scale) at a random position and scale in the C1 layer of a random training image. The values of all C1 units within the patch are read out and stored as a prototype. For a 4x4 patch, this means 16 different positions, but for each position, there are units representing each of 4 orientations (see the "dense" prototype in figure 2). Thus a 4x4 patch actually contains 4x4x4 = 64 C1 unit values.

Preliminary tests seemed to confirm that multiple feature sizes worked somewhat better than any single size. Smaller (4x4) features can be seen as encoding shape, while larger features are probably more useful for texture. Since we learn the prototype patches randomly from unsegmented images, many will not actually represent the object of interest, and others may not be useful for the classification task. The weighting of features is left for the later SVM step. It should be noted that while each S2 prototype is learned by sampling from a specific image of a single category, the resulting dictionary of features is shared, i.e. all features are used by all categories.

During normal operation (after feature learning) each of these prototypes can be seen as just another convolution filter which is run over C1. We generate an S2 pyramid with roughly the same number of positions/scales as C1, but having d types of units at each position/scale, each representing the response of the corresponding C1 patch to a specific prototype patch; see figure 1. The S2 layer is intended to correspond to cortical area V4 or posterior IT.

The response of a patch of C1 units X to a particular S2 feature/prototype P, of size $n \times n$, is given by a Gaussian radial basis function:

$$R(X, P) = \exp\left(-\frac{\|X - P\|^2}{2\sigma^2\alpha}\right) \qquad (3)$$

Both X and P have dimensionality $n \times n \times 4$, where $n \in$

$\{4, 8, 12, 16\}$. As in [23], the standard deviation σ is set to 1 in all experiments.

The parameter α is a normalizing factor for different patch sizes. For larger patches $n \in \{8, 12, 16\}$ we are computing distances in a higher dimensional space; for the distance to be small, there are more dimensions that have to match. We reduce the weight of these extra dimensions by using $\alpha = (n/4)^2$, which is the ratio of the dimension of P to the dimension of the smallest patch size.

Global invariance (C2) layer. Finally we create a d-dimensional vector, each element of which is the maximum response (anywhere in the image) to one of the model's d prototype patches. At this point, all position and scale information has been removed.

SVM classifier. The C2 vectors are classified using an all-pairs linear SVM[2]. Data is "sphered" before classification: the mean and variance of each dimension are normalized to zero and one respectively.[3] Test images are assigned to categories using the majority-voting method.

Differences from Serre et al. Our base model, as described above, performs about as well as that of Serre et al. in [23]. However, in [23]:

- image *height* is always scaled to 140,
- a pyramid approach is not used (different sized filters are applied to the full-scale image),
- the S1 parameters σ and λ change from scale to scale,
- S1 filters differ in size additively,
- C1 subsampling ranges do not overlap in scale, and
- S2 has no α parameter.

2.3. Improvements

We have developed and tested a number of improvements to the base model. Each of these is described below. Testing results for each modification are provided in section 3.

Sparsify S2 inputs. In the base model, an S2 unit computes its response using all the possible inputs in its corresponding C1 patch. Specifically, at each position in the patch, it is looking at the response to every orientation of Gabor filter and comparing it to its prototype. Based on the principle that features should be as sparse as possible, we reduce the number of inputs to an S2 feature to one per C1 position. In the feature learning phase, we remember the identity and magnitude of the *dominant* orientation (maximally responding C1 unit) at each of the $n \times n$ positions in the patch. This is illustrated in figure 2; a 4x4 prototype patch now contains only 16 C1 unit values, not 64. When computing responses

[2]We use the Statistical Pattern Recognition Toolbox for Matlab [8].

[3]Suggested by T. Serre (personal communication).

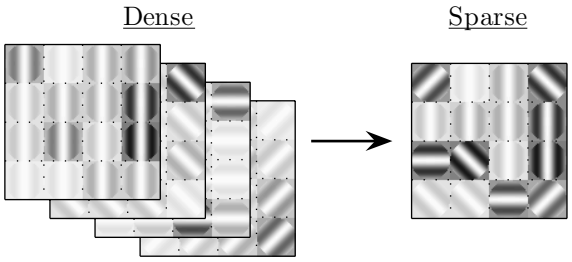

Figure 2. Dense vs. sparse S2 features. Dense S2 features in the base model are sensitive to all orientations of C1 units at each position. Sparse features are sensitive only to a particular orientation at each position. A 4x4 S2 feature for a 4-orientation model is shown here. Stronger C1 unit responses are shown as darker.

to S2 features, equation 3 is still used, but with a lower dimensionality: for each position in the patch, the S2 feature only cares about the value of the C1 unit representing its preferred orientation for that position.

In conjunction with this we increase the number of Gabor filter orientations in S1 and C1 from 4 to 12. Since we're now looking at particular orientations, rather than combinations of responses to all orientations, it becomes more important to represent orientation accurately. Cells in visual cortex also have much finer gradations of orientation than $\pi/4$ [13].

Inhibit S1/C1 outputs. Our second modification is similar – we again ignore non-dominant orientations, but here we focus not on pruning S2 feature inputs but on suppressing S1 and C1 unit *outputs*. In cortex, lateral inhibition refers to units suppressing their less-active neighbors. We adopt a simple version of this between S1/C1 units encoding different orientations at the same position and scale. Essentially these units are competing to describe the dominant orientation at their location.

We define a global parameter h, the *inhibition level*, which can be set between 0 and 1 and represents the fraction of the response range that gets suppressed. At each location, we compute the minimum and maximum responses, R_{min} and R_{max}, over all orientations. Any unit having $R < R_{min} + h(R_{max} - R_{min})$ has its response set to zero.

As a result, if a given S2 unit is looking for a response to a vertical filter (for example) in a certain position, but there is a significantly stronger horizontal edge in that rough position, the S2 unit will be penalized.

Limit position/scale invariance of S2 features. Like many "bag of features" models [4], the base model disregards all geometry above the level of S2 units. It simply uses the maximum response to each S2 feature at any position or scale. This gives complete position and scale invariance, but S2 features are still too simple to eliminate bind-

ing problems: we are still vulnerable to false positives due to chance co-occurrence of features from different objects and/or background clutter.

We wanted to investigate the option of retaining some geometric information above the S2 level. In fact, neurons in V4 and IT do not exhibit full invariance and are known to have receptive fields limited to only a portion of the visual field and range of scales [22]. To model this, we simply restrict the region of the visual field in which a given S2 feature can be found, relative to its location in the image from which it was originally sampled, to $\pm t_p\%$ of image size and $\pm t_s$ scales, where t_p and t_s are global parameters.

This approach assumes the system is "attending" close to the center of the object. This is appropriate for datasets such as the Caltech 101, in which most objects of interest are at similar positions and scales within the image. For the more general detection of objects within complex scenes, as in the UIUC car database, we augment it with a search for peak responses over object location using a sliding window.

Select features that are highly weighted by the SVM. Our S2 features are prototype patches randomly selected from unsegmented training images. Many will be from the background, and others will have varying degrees of usefulness for the classification task. We wanted to find out how many features were actually needed, and whether cutting out less-useful features would improve performance, as we might expect from machine learning results on the value of sparsity.

We use a simple feature selection technique based on SVM normals [19]. In fitting separating hyperplanes, the SVM is essentially doing feature weighting. Our all-pairs m-class linear SVM consists of $m(m-1)/2$ binary SVMs. Each fits a separating hyperplane between two sets of points in d dimensions, in which points represent images and each dimension is the response to a different S2 feature. The d components of the (unit length) normal vector to this hyperplane can be interpreted as feature weights; the higher the k^{th} component (in absolute value), the more important feature k is in separating the two classes.

To perform feature selection, we simply drop features with low weight. Since the same features are shared by all the binary SVMs, we do this based on a feature's average weight over all binary SVMs. Starting with a pool of 12,000 features, we conduct a multi-round "tournament". In each round, the SVM is trained, then at most[4] half the features are dropped. The number of rounds depends on the desired final number of features d. (For performance reasons, earlier rounds are carried out using multiple SVMs, each containing at most 3,000 features.)

[4]Depending on the desired number of features it may be necessary to drop less than half per round.

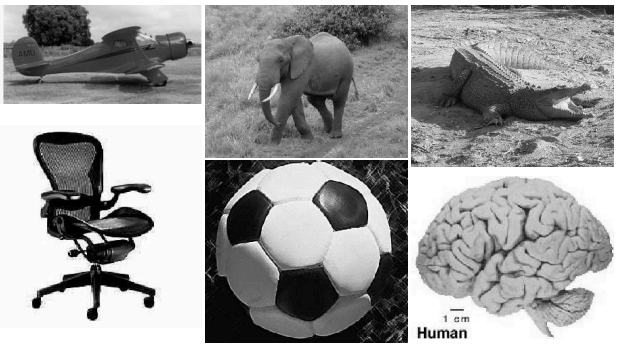

Figure 3. Some images from the Caltech 101 dataset.

Model	15 training images/cat.	30 training images/cat.
Our model (base)	33	41
Serre et al. [23]	35	42
Holub et al. [12]	37	43
Berg et al. [2]	45	
Grauman & Darrell [11]	49.5	**58.2**
Our model (final)	**51**	56

Table 1. Published classification results for the Caltech 101 dataset. Results for our model are the average of 8 independent runs. Scores shown are the average of the per-category classification rates.

Our experiments show that dropping features (effectively setting their weights to zero rather than those assigned by the SVM) improves classification performance, and the resulting model is more economical to compute.

3. Multiclass experiments (Caltech 101)

The Caltech 101 dataset contains 9,197 images comprising 101 different object categories, plus a background category, collected via Google image search by Fei-Fei et al. [5]. Most objects are centered and in the foreground, in a stereotypical pose. Some sample images are shown in figure 3.

First we ran our base model (described in section 2.2) on the entire set. The results are shown in table 1 for both 15 and 30 training images per category.

Each result is the average of 8 runs. For each run we:

1. choose 15 or 30 training images at random from each category, placing all remaining images in the test set,
2. learn features at random positions and scales from the training images (an equal number from each image),
3. build C2 vectors for the training set,
4. train the SVM (performing feature selection if that option is turned on),
5. build C2 vectors for the test set and classify the test images.

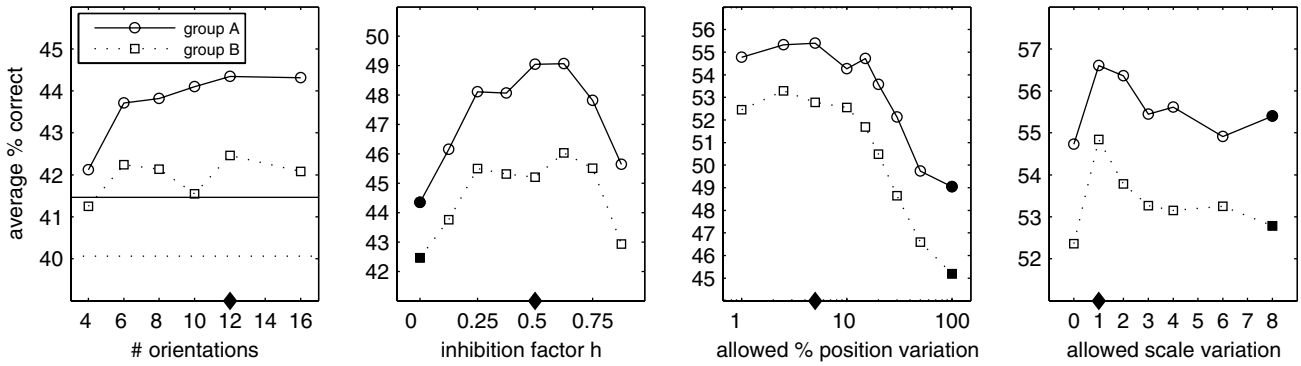

Figure 4. The results of parameter tuning for various enhancements to the base model using the Caltech 101 dataset. Each data point is the average of 8 independent runs, using 15 training images and up to 100 test images per category. Tests were run independently on two disjoint groups of 50 categories each. The horizontal lines in the leftmost graph show the performance of the base model (dense features, 4 orientations) on the two groups. Tuning is cumulative: the parameter value chosen in each graph is marked by a solid diamond on the x-axis. The results for this parameter value become the starting points (shown as solid data points) for the next graph.

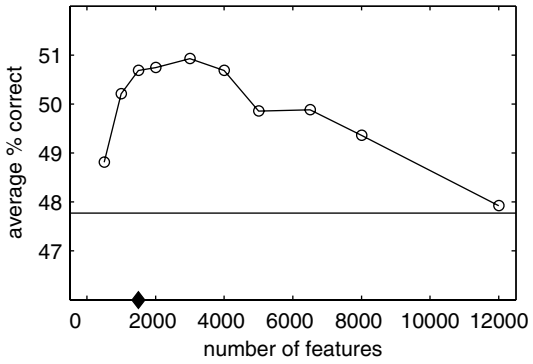

Figure 5. Results for the final model on the entire Caltech 101 dataset for various numbers of features, selected from a pool of 12,000. Each data point is the average of 4 runs with 15 training images and up to 100 test images per category. The horizontal line represents the performance of the same model but with 4,075 randomly selected features and no feature selection.

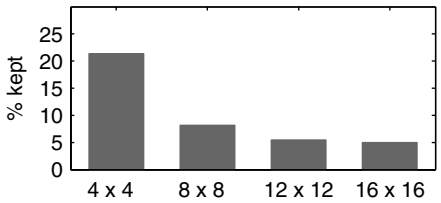

Figure 6. Percentage of each size of feature remaining after feature selection, using the final number of features (1500).

Next we successively turned on the improvements described in section 2.3. Each has one or two free parameters that must be tuned. The complete parameter space is too large to search exhaustively, hence we chose an order and optimized each parameter separately before moving to the next. First we turned on S2 input sparsification and found a good number of orientations, then we fixed that number and moved on to find a good inhibition level, etc. Our goal was to find parameter values that could be used for any dataset, so we wanted to guard against the possibility of tuning parameters to unknown properties specific to the Caltech 101. This large dataset has enough variety to make this unlikely; nevertheless, we ran tests independently on two disjoint subsets of the categories and chose parameter values that fell in the middle of the good range for both groups (see figure 4). The fact that such values were easy to find increases our confidence in the generality of the chosen values. The two groups were constructed as follows:

1. remove the easy faces and background categories,
2. sort the remaining 100 categories by number of images, then
3. place odd numbered categories into group A and even into group B.

The final parameter, number of features, was optimized for all 102 categories. Since models with fewer features can be computed more quickly, we chose the smallest number of features that still gave results close to the best. Among these surviving features, the 4x4 size dominates (figure 6), suggesting that this size yields the most informative features for this task [24].

The results of parameter tuning are shown in figures 4 and 5. Note that we limited the number of test images per category to 100 to save time. The chosen parameters were 12 orientations, h = 0.5, t_p = $\pm$5%, t_s = $\pm$1 scale, 1500 features.

Finally, we computed results for our final model, using both 15 and 30 training images and all remaining test images per category. Again, each result is the average of 8 independent runs. The results are summarized in table 1, along with those from other published studies. Our final results for 15 and 30 training images are 51% and 56%.

Model	Single-scale	Multiscale
Agarwal et al. [1]	76.5	39.6
Leibe et al. [17]	97.5	
Fritz et al. [9]		87.8
Our model	**99.94**	**90.6**

Table 2. Detection/localization results (recall at equal-error rates) for the UIUC car dataset. The results for our model are the average of 8 independent runs. Scoring methods were those of [1].

According to advance copies of upcoming papers provided by the authors [15, 25], some further improved results for the Caltech 101 dataset will be published soon. These use improved kernels for the SVM classifier (as does Grauman & Darrell [11]). It will be interesting to see whether these ideas can be successfully combined with our sparse image features to get further improvements.

4. Detection / localization experiments (UIUC car dataset)

We ran our final, tuned model on the UIUC car dataset [1]. This dataset consists of small (100x40) training images of cars and background, and larger test images in which there is at least one car to be found. There are two sets of test images: a single-scale set in which the cars to be detected are roughly the same size (100x40 pixels) as those in the training images, and a multi-scale set.

Other than the number of features, which we set to 500 (selected from 4000 in 3 rounds, comparing features in groups of at most 1000), all parameters were unchanged. For localization in these larger images we added a sliding window. Duplicate detections were consolidated using the neighborhood suppression algorithm from [1].

We trained the model using 500 positive and 500 negative training images; features were sampled from these same images. As in [1], the sliding window moves in steps of 5 pixels horizontally and 2 vertically. We increase the width of a "neighborhood" from 71 to 111 pixels to avoid merging adjacent cars.

Our results are shown in table 2 along with those of other studies. Our recall at equal-error rates (recall = precision) is 99.94% for the single-scale test set and 90.6% for the multiscale set, averaged over 8 runs. Scores were computed using the scoring programs provided with the UIUC data.

In our single-scale tests, 7 of 8 runs scored a perfect 100% – all 200 cars in 170 images were detected with no false positives. To be considered correct, the detected position must lie inside an ellipse centered at the true position, having horizontal and vertical axes of 25 and 10 pixels respectively. Repeated detections of the same object count as false positives. Figure 8 shows the only errors from the 8^{th} run; figure 7 shows some correct single-scale detections.

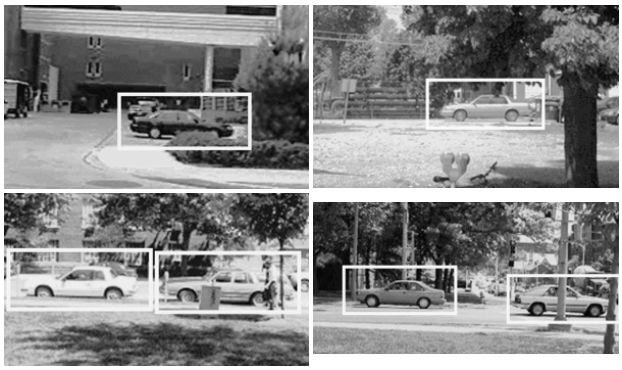

Figure 7. Some correct detections from one run on the single-scale UIUC car dataset.

Figure 8. The only 2 errors (1 missed detection, 1 false positive) made in 8 runs on the single-scale UIUC car dataset.

For the multiscale tests, the sliding window also searches through scale, and the scoring criteria include a scale tolerance (from [1]).

5. Discussion and future work

In this study we have shown that a biologically-based model can compete with other state-of-the-art approaches to object categorization, strengthening the case for investigating biologically-motivated approaches to object recognition. Even with our enhancements, this model is still relatively simple.

The system implemented here is not real-time; it takes several seconds to process and classify an image on a 2GHz Intel Pentium server. Hardware advances will reduce this to immediate recognition within a few years. Biologically motivated algorithms also have the advantage of being susceptible to massive parallelization. Localization in larger images takes longer; in both cases the bulk of the time is spent building feature vectors.

We have found increasing sparsity to be a fruitful approach to improving generalization performance. Our methods for increasing sparsity have all been motivated by approaches that appear to be incorporated in biological vision, although we have made no attempt to model biological data in full detail. Given that both biological and computer vision systems face the same computational constraints arising from the data, we would expect computer vision research to benefit from the use of similar basis functions for

describing images. Our experiments show that both lateral inhibition and the use of sparsified intermediate features contribute to generalization performance.

We have also examined the issue of feature localization in biologically based models. While very precise geometric constraints may not be useful for broad object categories, there is a substantial loss of useful information in completely ignoring feature location as in bag-of-features models. We have shown a considerable increase in performance by using intermediate features that are localized to small regions of an image relative to an object coordinate frame. When an object may appear at any position or scale in a cluttered image, it is necessary to search over all potential reference frames to combine appropriately localized features. In biological vision this attentional search appears to be driven by a complex range of saliency measures [22]. For our computer implementation, we can simply search over a densely sampled set of possible reference frames and evaluate each one. This has the advantage of not only improving classification performance but also providing quite accurate localization of each object. The strong performance shown on the UIUC car localization task indicates the potential for further work in this area.

As we do not wish to stray too far from what is clearly a valuable source of inspiration, we lean towards future enhancements that are biologically realistic. A likely first step would be to attempt to model intermediate level features (above V1) more accurately, possibly adding higher-order features or view-tuned units. In addition, there would likely be some benefit to clustering intermediate features to favor those that occur most frequently in the training set.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *PAMI*, 26(11):1475–1490, November 2004.

[2] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *CVPR*, June 2005.

[3] G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. In *CVPR*, June 2005.

[4] G. Csurka, C. Dance, J. Willamowski, L. Fan, and C. Bray. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague, 2004.

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *CVPR Workshop on Generative-Model Based Vision*, 2004.

[6] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[7] M. Figueiredo. Adaptive sparseness for supervised learning. *PAMI*, 25(9):1150–1159, September 2003.

[8] V. Franc and V. Hlavac. Statistical pattern recognition toolbox for Matlab.

[9] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminative models for object category detection. In *ICCV*, pages 1363–1370, Beijing, China, October 2005.

[10] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, April 1980.

[11] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. Technical Report MIT-CSAIL-TR-2006-020, March 2006.

[12] A. Holub, M. Welling, and P. Perona. Exploiting unlabelled data for hybrid object classification. In *NIPS Workshop on Inter-Class Transfer*, Whistler, B.C., December 2005.

[13] D. Hubel and T. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, 148:574–591, 1959.

[14] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *PAMI*, 27(6):957–968, 2005.

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

[16] 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.

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

[18] N. Logothetis, J. Pauls, and T. Poggio. Shape representation in the inferior temporal cortex of monkeys. *Current Biology*, 5:552–563, 1995.

[19] D. Mladenic, J. Brank, M. Grobelnik, and N. Milic-Frayling. Feature selection using linear classifier weights: Interaction with classification models. In *The 27th Annual International ACM SIGIR Conference (SIGIR 2004)*, pages 234–241, Sheffield, UK, July 2004.

[20] B. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[21] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

[22] E. T. Rolls and G. Deco. *The Computational Neuroscience of Vision*. Oxford University Press, 2001.

[23] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, San Diego, June 2005.

[24] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 5(7):682–687, 2002.

[25] H. Zhang, A. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, June 2006.