Object class recognition and localization using sparse features with limited receptive fields [*]

Jim Mutch [†]
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA
jmutch@mit.edu

David G. Lowe
Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

Abstract

We investigate the role of sparsity and localized features in a biologically-inspired model of visual object classification. As in the model of Serre, Wolf, and Poggio, 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. Sparsity is increased by constraining the number of feature inputs, lateral inhibition, and feature selection. We also demonstrate the value of retaining some position and scale information above the intermediate feature level. Our final model is competitive with current computer vision algorithms on several standard datasets, including the Caltech 101 object categories and the UIUC car localization task. The results further the case for biologically-motivated approaches to object classification.

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. Recent work by Serre, Wolf, and Poggio [32] used a computational model based on our knowledge of visual cortex to obtain promising results on some of the standard classification datasets. Our paper builds on their approach by incorporating some additional biologically-motivated properties, specifically, sparsity and localized intermediate-level features. We show that these modifications further improve classification 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 [9, 17] (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 [25]. In our object classification model, one way we have found to increase sparsity is to use a lateral inhibition step 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 are found most effective. We show that each of these changes provides a significant boost in generalization performance.

While some current successful methods for object classification learn and apply quite precise geometric constraints on feature locations [8, 3], others ignore geometry and use a "bag of features" approach that ignores the locations of individual features [4, 26]. Intermediate approaches retain some coarsely-coded location information [1] or record the locations of features relative to the object center [20, 2]. According to models of object recognition in cortex [29], 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. At higher levels, spatial structure becomes implicitly encoded into the features themselves, which may overlap, while explicit spatial information is coded more coarsely. The question becomes one of identifying the level at which features have become complex enough that explicit spatial information can be discarded. We investigate retaining some degree of position

[*]This paper updates and extends an earlier presentation [24] of this research in CVPR 2006.

[†]The research described in this paper was carried out at the University of British Columbia.

and scale sensitivity up to the level of object detection, 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 categories [7]. Our results show that there are significant improvements to classification performance from each of the changes. Further tests on the UIUC car database [1] and the Graz-02 datasets [26] demonstrate that the resulting system can also perform well on object localization. Our results further strengthen the case for incorporating concepts from biological vision into the design of computer vision systems.

2. Models

The model[1] presented in this paper is a partial implementation of the "standard model" of object recognition in cortex (as summarized by [29]), which focuses on the object recognition 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 [28, 33] implies mainly feedforward processing. While full human-level classification performance is almost certain to require feedback, the feedforward case is the easiest to model and thus represents an appropriate starting point. Within this immediate recognition framework, recognition of object classes from different 3D viewpoints is thought to be based on the learning of multiple 2D representations, rather than a single 3D representation [27].

2.1. Previous models

Our model builds on that of Serre *et al.* [32], which in turn extends the "HMAX" model of Riesenhuber and Poggio [29]. These are the latest of a group of models which can be said to implement parts of the standard model, including neocognitrons [12] and convolutional networks [19]. 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 [15]).

- Simple ("S") layers apply local filters that 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

[1]Source code and related documentation for our model may be downloaded at http://www.mit.edu/∼jmutch/fhlib.html.

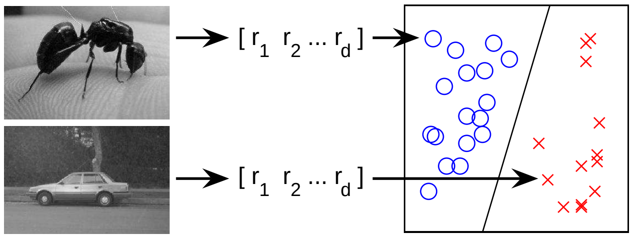

Figure 1. Overall form of our model. Images are reduced to feature vectors which are then classified by an SVM.

to account for the tuning and invariance properties [21] 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*, in which 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.* [32] introduced learning of intermediate-level shared features, made additional quantitative adjustments, and added a final SVM classifier to make the model useful for classification.

2.2. Our base model

We start with a "base" model which is similar to [32] and performs about as well. Nevertheless, it is an independent implementation, and we give its complete description here. Its differences from [32] 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 overall form of the model (shown in figure 1) is very simple. Images are reduced to feature vectors, which are then classified by an SVM. The dictionary of features is shared across all categories – all images "live" in the same feature space. The main focus of our work is on the feature computation stage.

Features are computed hierarchically in five layers: an initial image layer and four subsequent layers, each built from the previous by alternating template matching and max pooling operations. This process is illustrated in figure 2, and the following subsections describe each layer.

Note that features in all layers are computed at all positions and scales – interest point detectors are not used.

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).

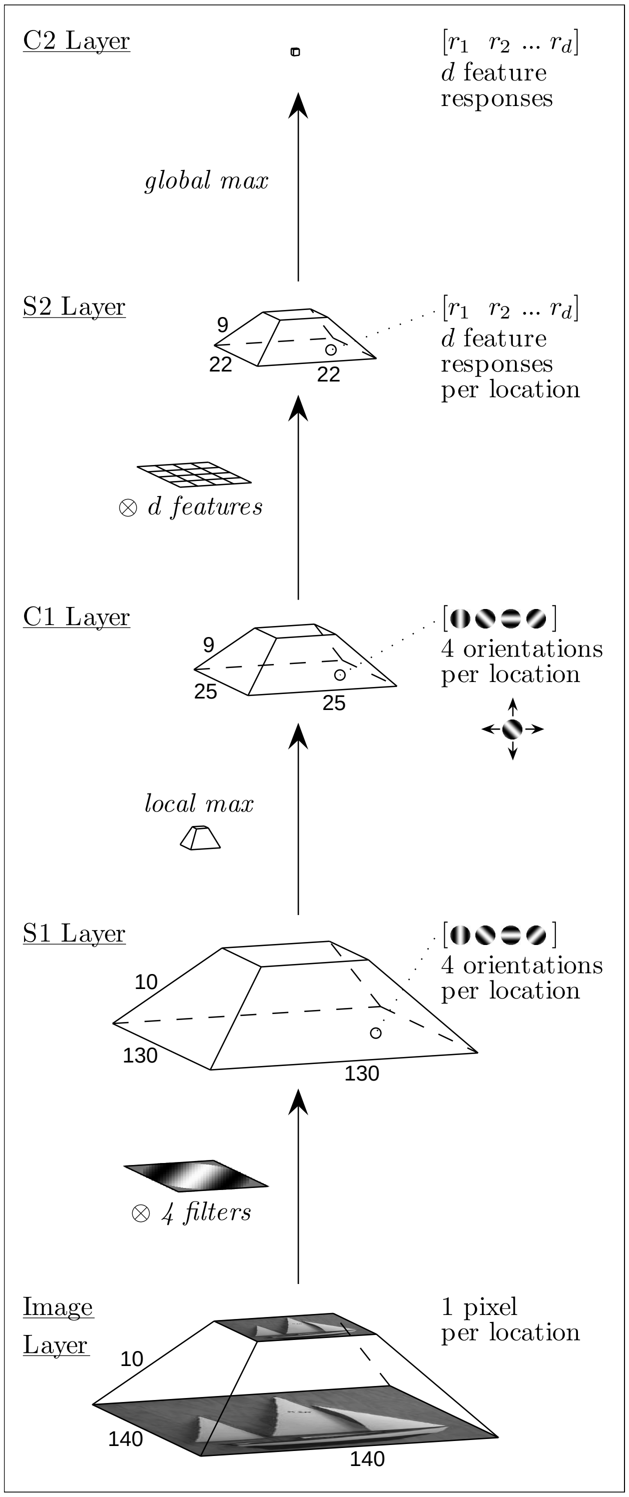

Figure 2. Feature computation in the base model. Each layer has units covering three spatial dimensions (x/y/scale), and at each 3D location, an additional dimension of *feature type*. The image layer has only one type (pixels), layers S1 and C1 have 4 types, and the upper layers have d (many) types per location. Each layer is computed from the previous by applying template matching or max pooling filters. Image size can vary and is shown for illustration.

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 [32] and uses 4 orientations. While 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 2). 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 are 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 [32] 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).

It should be noted that the filters produced by these parameters are quite clipped; in particular, the long axis of the Gabor filter does not diminish to zero before the boundary of the 11x11 array is reached. However, experiments showed that larger arrays failed to improve classification performance, and they were more expensive to compute.

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[2] 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 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 2. This layer provides a model for V1 complex cells.

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

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 [32].) 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 3). 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. Since we learn the prototype patches randomly from images containing background clutter, some will not actually represent the object of interest; others may simply 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 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 2. 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:

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

Both X and P have dimensionality $n \times n \times 4$, where $n \in \{4, 8, 12, 16\}$. As in [32], 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, i.e., we have a "bag of features".

SVM classifier. The C2 vectors are classified using an all-pairs linear SVM[3]. Data is "sphered" before classification: the mean and variance of each dimension are normalized to zero and one respectively.[4] 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 [32]. However, in [32]:

- 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

In this section we describe several changes which introduce sparsity and localized intermediate-level features into the model; these changes represent the main contributions of the paper. 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. Real neurons, however, are likely to be more selective among potential inputs. To increase sparsity among an S2 unit's inputs, 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 3; the resulting 4x4 prototype patch now contains only 16 C1 unit values, not 64. When computing responses to such "sparsified" 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

[3]We use the Statistical Pattern Recognition Toolbox for MATLAB [10].
[4]Suggested by T. Serre (personal communication).

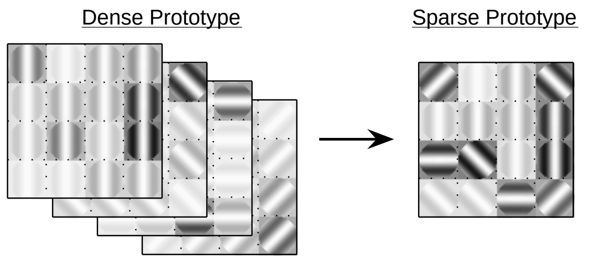

Figure 3. Dense vs. sparse S2 features. Dense S2 features in the base model are sensitive to all orientations of C1 units at each position. Sparse features are sensitive only to a particular orientation at each position. A 4x4 S2 feature for a 4-orientation model is shown here. Stronger C1 unit responses are shown as darker.

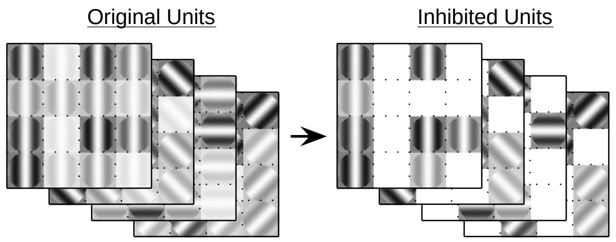

Figure 4. Inhibition in S1/C1. The weaker units (i.e., orientations) at each position are suppressed. A 4x4 patch of units (at a single scale) is shown here for a 4-orientation model. Stronger unit responses are shown as darker.

C1 unit representing its preferred orientation for that position. This makes the S2 unit less sensitive to local clutter, improving generalization.

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 $\pi/4$ [15].

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. This is illustrated in figure 4.

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 in C2. Above the S2 level, the base model becomes a "bag of features" [4], disregarding all geometry. The C2 layer simply takes the maximum response to each S2 feature over all positions and scales. This gives complete position and scale invariance, but S2 features are still too simple to eliminate binding problems:

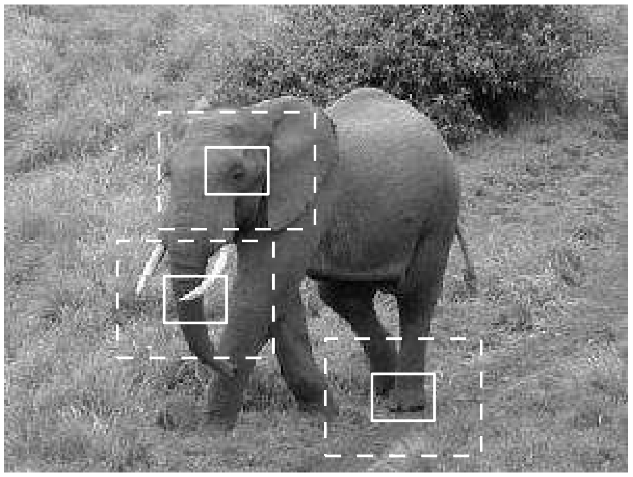

Figure 5. Limiting the position/scale invariance of C2 units. The solid boxes represent S2 features sampled from this training image. In test images, we will limit the search for the maximum response to each S2 feature to the positions represented by the corresponding dashed box. Scale invariance is similarly limited (although not shown here).

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 [30]. 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 is illustrated in figure 5.

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 central and dominant. 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 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 [22]. 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[5] 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.)

Our experiments show that dropping features (effectively forcing 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.* [7]. Most objects are centered and in the foreground, making it an excellent test of basic classification with a large number of categories. (The Caltech 101 has become the unofficial standard benchmark for this task.) Some sample images can be seen in figures 8 and 9.

First we ran our base model (described in section 2.2) on the full 102-category dataset. The results are shown in table 1 and are comparable to those of [32].

Classification scores for our model are averaged over 8 runs. For each run we:

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

Model	**15 training images/cat.**	**30 training images/cat.**
Our model (base)	*33*	*41*
Serre *et al.* [32]	35	42
Holub *et al.* [14]	37	43
Berg *et al.* [2]	45	
Grauman & Darrell [13]	50	58
Our model (final)	*51*	*56*
Lazebnik *et al.* [18]	56	65
Zhang *et al.* [35]	**59**	**66**

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

1. choose 15 or 30 training images at random from each category, placing 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.

Next we successively turned on the improvements described in section 2.3. Each has one or two free parameters. 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 these 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 6). 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 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. This process is illustrated in figures 6 and 7. The last parameter, number of features, was optimized for all 102 categories as a single

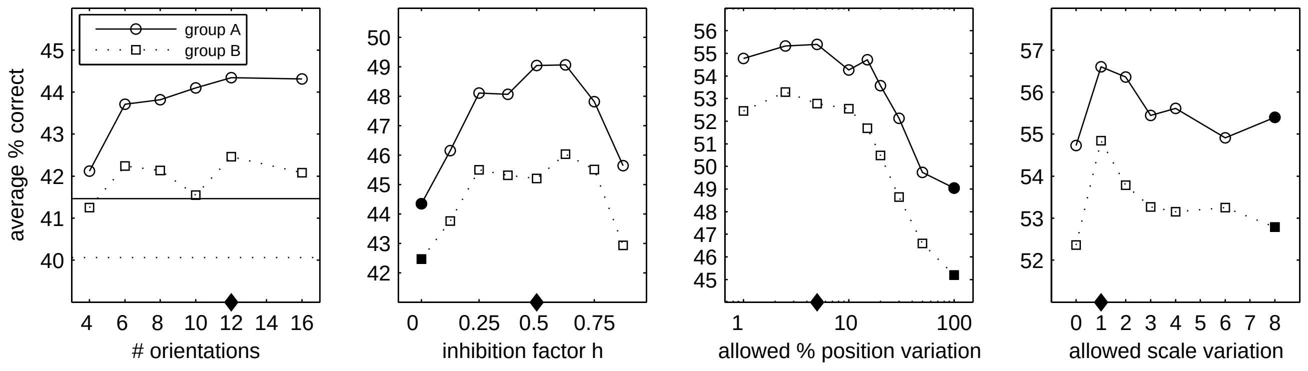

Figure 6. The results of parameter tuning for successive enhancements to the base model using the Caltech 101 dataset. Tests were run independently on two disjoint groups of 50 categories each. The horizontal lines in the leftmost graph show the performance of the base model (dense features, 4 orientations) 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. Each data point is the average of 8 independent runs, using 15 training images and up to 100 test images per category.

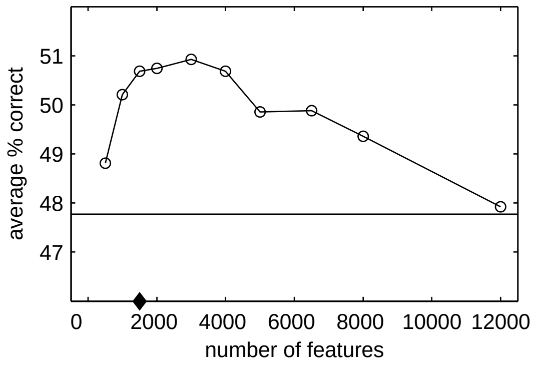

Figure 7. Results for the final model on all 102 categories using various numbers of features, selected from a pool of 12,000 features. The horizontal line represents the performance of the same model but with 4,075 randomly selected features and no feature selection. Each data point is the average of 4 runs with 15 training images and up to 100 test images per category.

Model version	**15 training images/cat.**	**30 training images/cat.**
Base	33	41
+ sparse S2 inputs	35 (+ 2)	45 (+ 4)
+ inhibited S1/C1 outputs	40 (+ 5)	49 (+ 4)
+ limited C2 invariance	48 (+ 8)	54 (+ 5)
+ feature selection	51 (+ 3)	56 (+ 2)

Table 2. The contribution of our successive modifications to the overall classification score, using all 102 categories. Each score is the average of 8 independent runs using all available test images. Scores shown are the average of the per-category classification rates.

group. 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.

The parameter values ultimately chosen were 12 orientations, $h = 0.5$, $t_p = \pm 5\%$, $t_s = \pm 1$ scale, 1500 features. Classification scores for the final model, which incorporates these parameters, are shown in table 1 along with those from other published studies. Our final results for 15 and 30 training images, using all 102 categories, are 51% and 56%. [6]

Table 2 shows the contribution to performance of each successive modification, using all 102 categories.

[6]When originally submitted for publication, these scores exceeded all previously published results for this dataset. Concurrent work by Lazebnik *et al.* [18] and Zhang *et al.* [35] scored higher. These approaches focus on improved SVM kernels, as does that of Grauman & Darrell [13]. A possible future project could involve replacing our simple classifier with one based on these ideas.

Figure 8 contains some examples of categories for which the system performed well, while figure 9 illustrates some difficult categories. In general, the harder categories are those having greater shape variability due to greater intra-category variation and nonrigidity. Interestingly, the frequency of occurence of background clutter in a category's images does not seem to be a significant factor. Note that performance is worst on the "background" category. This is not surprising, as our system does not currently have a special case for "none of the above". Background is treated as just another category, and the system attempts to learn it from at most 30 exemplars.

Table 3 shows the ten most common classification errors. Notably, most of these errors are not outrageous by human standards. The most common confusions are schooner *vs.* ketch (indistinguishable by non-expert humans) and lotus *vs.* water lily (similar flowers).

3.1. The selected features

Figure 10 shows the proportion of S2 features of each size (4x4, 8x8, etc.) that survived the feature selection process

Figure 8. Examples of Caltech 101 categories on which our system performed well.

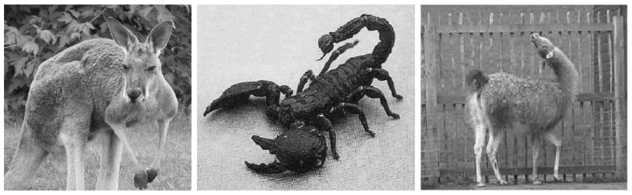
Figure 9. Examples of Caltech 101 categories which our system found more difficult.

Category	**Most Common Error**	**Frequency (%)**
schooner	ketch	19.32
lotus	water lilly	18.75
ketch	schooner	17.11
scorpion	ant	8.80
elephant	brontosaurus	8.46
crab	crocodile	7.85
crayfish	lobster	7.50
ibis	emu	7.50
lamp	flamingo	6.85
llama	kangaroo	6.51

Table 3. The ten most common errors on the Caltech 101 dataset, for the final model using 30 training images per category, averaged over 8 runs. Only categories having at least 30 remaining test images are included here.

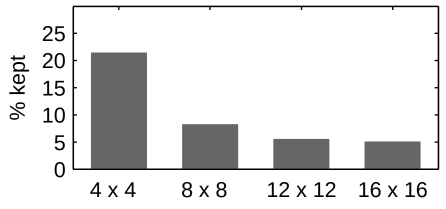

Figure 10. Percentage of each size of feature remaining after feature selection, using the final number of features (1500), averaged over 8 runs.

for the final model. Among these surviving features, the 4x4 size dominates, suggesting that this size generally yields the most informative features for this task [34].

Because S2 features are not directly made up of pixels, but rather C1 units, it is not possible to uniquely show what they "look like". However, it is possible to find the image patches in the test set to which a given feature responds most strongly. Figures 16 and 17 (end of paper) show two features from a particular run on the Caltech 101 dataset. According to the selection criteria, these features were ranked #1 and #101, respectively.

For most features, the highest-responding patches do not all come from one object category, although there are often a few commonly recurring categories. S2 features are still rather weak classifiers on their own.

4. Localization experiments (UIUC cars)

We ran our final model on the UIUC car dataset [1]. These experiments served two purposes.

- Our introduction of limited C2 invariance (section 2.3) sacrificed full invariance to object position and scale within the image; we wanted to see if we could recover it and at the same time perform object localization.
- We wanted to demonstrate that the model, and the parameters learned during the tuning process, could perform well on another dataset.

The UIUC car 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, all parameters were unchanged. The number of features was arbitrarily set to 500 and immediately yielded excellent results. We did not attempt to optimize system speed by reducing this number as we did in the multiclass experiments. As before, the features were selected from a group of randomly-sampled features eight times larger, 4000 in this case, and the selection process comprised 3 rounds. Features were compared in groups of at most 1000. See section 2.3 for details.

We trained the model using 500 positive and 500 negative training images; features were sampled from these same images.

For localization in these larger test images we added a sliding window. As in [1], the sliding window moves in steps of 5 pixels horizontally and 2 vertically. In the multiscale case this is done at every scale using these same step sizes. At larger scales there are fewer pixels, each representing more of the image, hence there are fewer window positions at larger scales.

Duplicate detections were consolidated using the neighborhood suppression algorithm from [1]. We increase the width of a "neighborhood" from 71 to 111 pixels to avoid merging adjacent cars.

Our results are shown in table 4 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.

Model	**Single-scale**	**Multiscale**
Agarwal *et al.* [1]	76.5	39.6
Leibe *et al.* [20]	97.5	
Fritz *et al.* [11]		87.8
Our model (final)	**99.94**	**90.6**

Table 4. Our results (recall at equal-error rates) for the UIUC car dataset along with those of previous studies. Scores for our model are the average of 8 independent runs. Scoring methods were those of [1].

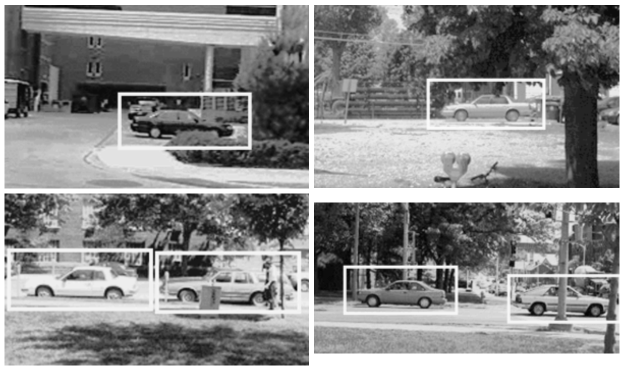

Figure 11. Some correct detections from one run on the single-scale UIUC car dataset.

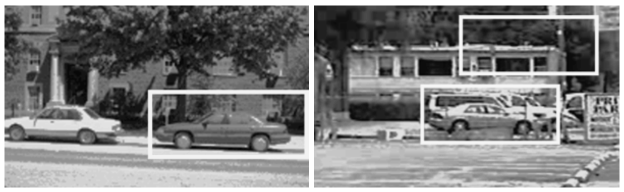

Figure 12. The only 2 errors (1 missed detection, 1 false positive) made in 8 runs on the single-scale UIUC car dataset.

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 12 shows the only errors from the 8^{th} run; figure 11 shows some correct single-scale detections.

For the multiscale tests, the scoring criteria include a scale tolerance (from [1]). Figures 13 and 14 show some correct detections and some errors on the multiscale set. Table 5 contains a breakdown of the types of errors made. Even in the multiscale case, outright false positives and missed detections are uncommon. Most of the errors are due to the following two reasons.

1. Two cars are detected correctly, but their bounding boxes overlap. This is more common in the multiscale case; see for example figure 14, bottom left. The neighborhood suppression algorithm eliminates one of them. Careful redesign of the suppression method could likely eliminate this type of error.

2. For certain instances of cars, the peak response, i.e., the highest-responding placement of the bounding box, occurs at a scale somewhat larger or smaller than that of the best bounding box. This is considered a missed detection (and a false positive) by the scoring algorithm [1].

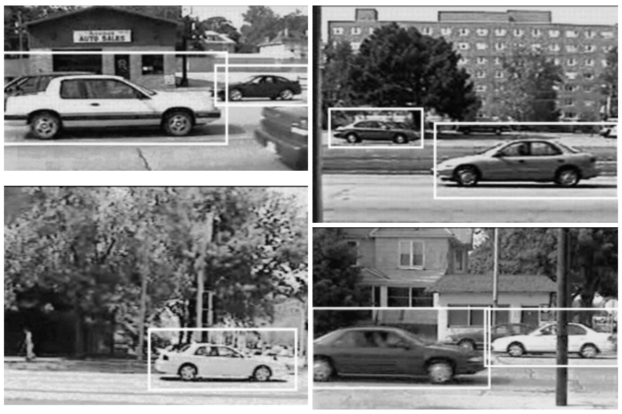

Figure 13. Some correct detections from one run on the multiscale UIUC car dataset.

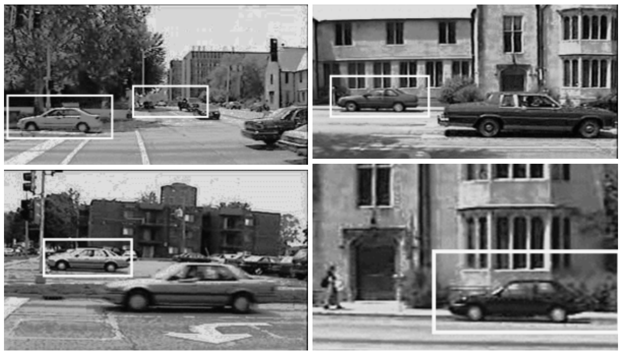

Figure 14. Examples of the kinds of errors made for one run on the multiscale UIUC car dataset. Top left: a simple false positive. Top right: a simple false negative. Bottom left: the second car is suppressed due to overlapping bounding boxes. Bottom right: the car is detected but the scale is slightly off.

Source of error	**Number of test images**
Simple false positive	1
Simple false negative	1
Suppression due to overlap	6
Detection at wrong scale	6

Table 5. Frequency of error types for one run on the multiscale UIUC car dataset.

5. Localization experiments (Graz-02)

Our final tests were conducted on the more difficult images of the Graz-02 [26] dataset; some example images may be seen in figure 15. Like the UIUC car dataset, Graz-02 was designed for the binary, single-category-*vs.*-background task, and the objects of interest are not necessarily central or dominant. It differs from the UIUC car dataset in the following ways.

1. There are three different positive categories: bikes, cars, and people. Nevertheless, the standard task is to distinguish *one* of these categories from the background category at a time, i.e., bikes *vs.* background, cars *vs.* background, and people *vs.* background.
2. The images are more difficult. There is a great deal of pose variation. Objects may be partially occluded and often appear in overlapping clusters.
3. While ground-truth location data is provided, it is generally not used for training, and there is no separate set of smaller training images.
4. The standard task is only to determine whether a test image contains an instance of the positive category or not. Its location within the image does not need to be reported.

Points (3) and (4) above make direct comparison with other studies difficult. Other work on this dataset has used pure bag-of-features models designed for the presence-or-absence task. Because we use localized features, we require training images in which the object is central and dominant, hence we do make use of ground-truth data in the training phase. And because we use a sliding window to identify objects in larger scenes, we end up solving the harder task of localization and then simply taking the peak response, throwing the location information away in order to compare to previous results.

Our positive training set was built from 50 randomly-selected square subimages containing a single object each.[7] Each such subimage was left-right reflected, resulting in a total of 100 positive examples. The negative training set consisted of 500 randomly-selected "background" subimages, equal in size to the average bounding box of the positive examples. Some training subimages can be seen in figure 15. The images from which the training subimages came were set aside, i.e., they were unavailable for testing. We learned a dictionary of 1,000 features (selected in 3 rounds from 8,000); all other parameters were again unchanged.

Given the many differences in task definition and the training sets, our results cannot be considered directly comparable to others. We used less training data (50 images compared to 300), but had the benefit of using ground-truth localization for training. With these caveats, our whole-image classification results were 80.5% for bikes, 70.1% for cars, and 81.7% for people. This is quite similar to the results of Opelt *et al.* [26], who obtained 77.8% for bikes, 70.5% for cars, and 81.2% for people. Recently, better results have been obtained by Moosmann *et al.* [23] for two of the categories (84.4% for bikes, 79.9% for cars), in part through the use of color information. Our reason for including these experiments is to test the application of our approach to more difficult problems with wide variation in viewpoint and object location, but a full comparison to other methods will require development of new data sets.

[7]This may have the side-effect of removing some of the easier images (those containing easily-separable single objects) from the test set.

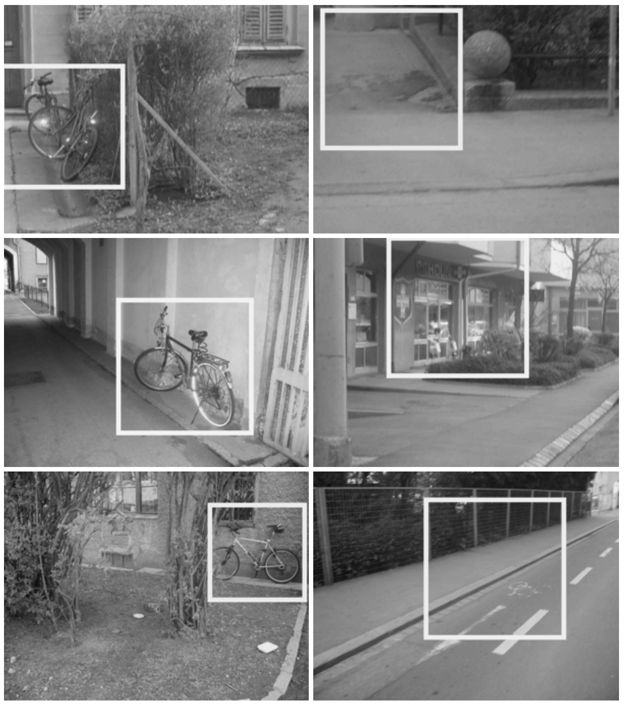
Figure 15. Some subimages used to train our Graz-02 "bikes" classifier.

6. 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 classification, strengthening the case for investigating biologically-motivated approaches to this problem. Even with our enhancements, the 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 speeds 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-level 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 still 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 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 [30]. 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.

Most of the performance improvements for our model were due to the feature computation stage. Other recent multiclass studies [18, 35] have done well using a more complex SVM classifier stage. From a pure performance point of view, the most immediately fruitful direction might be to try to combine these ideas into a single system. However, 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. We would like to be able to transform images into a feature space in which a simple classifier is good enough [5]. Even our existing classifier is not entirely plausible, as an all-pairs model does not scale well as the number of classes increases.

Another biologically implausible aspect of the current model is that it ignores the bandwidth limitations of single cells. On the timescale of immediate recognition, an actual neuron will have time to fire only a couple of spikes. Thus, it is more accurate to think of a single model unit as representing the synchronous activity of a population of cells having similar tuning [16].

The initial, feedforward mode of classification is the obvious first step towards emulating object classification in humans. A more recent model by Serre *et al.* [31] – having a slightly deeper feature hierarchy that better corresponds to known connectivity between areas in the ventral stream – has been able to match human performance levels for the classic animal/non-animal rapid classification task of Thorpe *et al.* [33]; however, large multiclass experiments have not yet been carried out. One might expect a deeper hierarchy having higher-order features or units explicitly tuned to different 2D views of an object to perform better on the more difficult datasets involving wide variation in pose.

Our work so far has focused mainly on the sparse structure of features. However, the process of learning these features from data in the current model is still quite crude – features are simply sampled at random and then discarded later if they do not prove useful. It is also unclear how well this method would extend to a model having a deeper hierarchy of features. More sophisticated methods will almost certainly be required. Previous hierarchical models (the neocognitron [12] and convolutional networks [19]) have investigated a number of bottom-up and top-down methods. Epshtein and Ullman [6] provide a principled, top-down approach for building feature hierarchies based on recursive decomposition of features into maximally informative sub-features. However, this technique would need to be extended to the multiclass, shared-feature case, and to the case in which the higher-level features are not pixel patches. It is likely that future work will incorporate elements of both bottom-up and top-down selection and learning of features.

Ultimately, the feedforward model should become the core of a larger system incorporating feedback, attention, and other top-down influences.

Acknowledgements

We thank the reviewers for several helpful comments. Funding for this research was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canadian Institute for Advanced Research (CIAR), and the University of British Columbia's University Graduate Fellowship program.

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. 1, 2, 8, 9

[2] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *CVPR*, June 2005. 1, 6

[3] G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. In *CVPR*, June 2005. 1

[4] G. Csurka, C. Dance, J. Willamowski, L. Fan, and C. Bray. Visual categorization with bags of keypoints. In *ECCV In-*

ternational Workshop on Statistical Learning in Computer Vision, Prague, 2004. 1, 5

[5] J. DiCarlo and D. Cox. Untangling invariant object recognition. *Trends in Cognitive Science*, 11:333–341, 2007. 11

[6] B. Epshtein and S. Ullman. Feature hierarchies for object classification. In *ICCV*, Beijing, 2005. 11

[7] 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. 2, 6

[8] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003. 1

[9] M. Figueiredo. Adaptive sparseness for supervised learning. *PAMI*, 25(9):1150–1159, September 2003. 1

[10] V. Franc and V. Hlavac. Statistical pattern recognition toolbox for Matlab. 4

[11] 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. 9

[12] 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. 2, 11

[13] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. Technical Report MIT-CSAIL-TR-2006-020, March 2006. 6, 7

[14] 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. 6

[15] D. Hubel and T. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, 148:574–591, 1959. 2, 5

[16] U. Knoblich, J. Bouvrie, and T. Poggio. Biophysical models of neural computation: Max and tuning circuits. Technical Report CBCL paper, April 2007. 11

[17] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *PAMI*, 27(6):957–968, 2005. 1

[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, June 2006. 6, 7, 11

[19] 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. 2, 11

[20] 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. 1, 9

[21] N. Logothetis, J. Pauls, and T. Poggio. Shape representation in the inferior temporal cortex of monkeys. *Current Biology*, 5:552–563, 1995. 2

[22] 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. 6

[23] F. Moosmann, B. Triggs, and F. Jurie. Randomized clustering forests for building fast and discriminative visual vocabularies. In *Neural Information Processing Systems (NIPS)*, November 2006. 10

[24] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, pages 11–18, New York, June 2006. 1

[25] 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. 1

[26] A. Opelt, A. Pinz, M.Fussenegger, and P.Auer. Generic object recognition with boosting. *PAMI*, 28(3), March 2006. 1, 2, 10

[27] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266, January 1990. 2

[28] M. Potter. Meaning in visual search. *Science*, 187:965–966, 1975. 2

[29] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999. 1, 2

[30] E. T. Rolls and G. Deco. *The Computational Neuroscience of Vision*. Oxford University Press, 2001. 5, 11

[31] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical Report CBCL Paper #259/AI Memo #2005-036, Massachusetts Institute of Technology, Cambridge, MA, October 2005. 11

[32] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, San Diego, June 2005. 1, 2, 3, 4, 6

[33] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996. 2, 11

[34] 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. 8

[35] H. Zhang, A. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, June 2006. 6, 7, 11

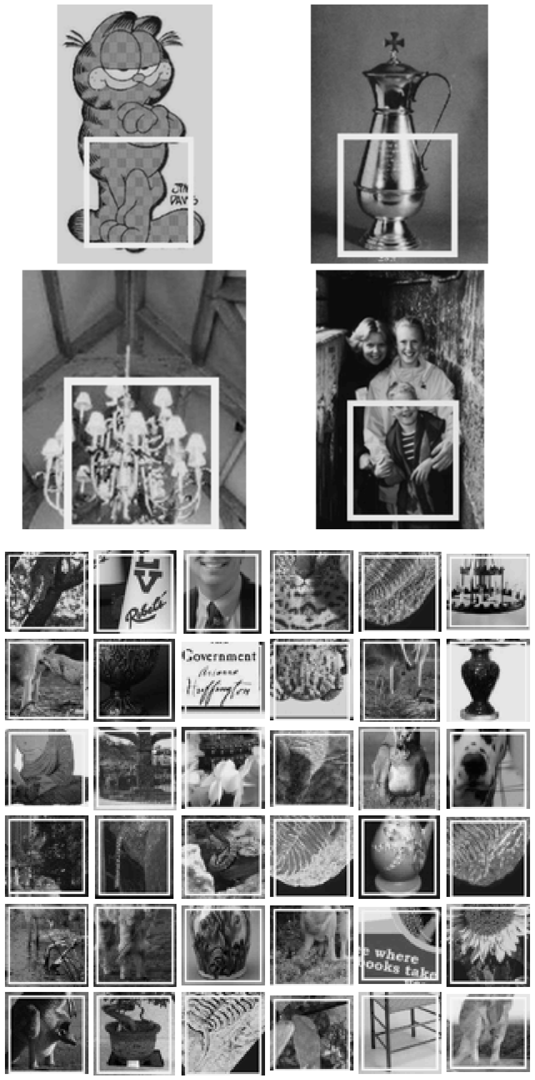
Figure 16. The 40 best image patches for feature #1, from one run of the final model on the Caltech 101 dataset. The four images at the top show the best-matching locations for the feature within the context of full images. To save space, for the next 36 matches we display only the matched patch.

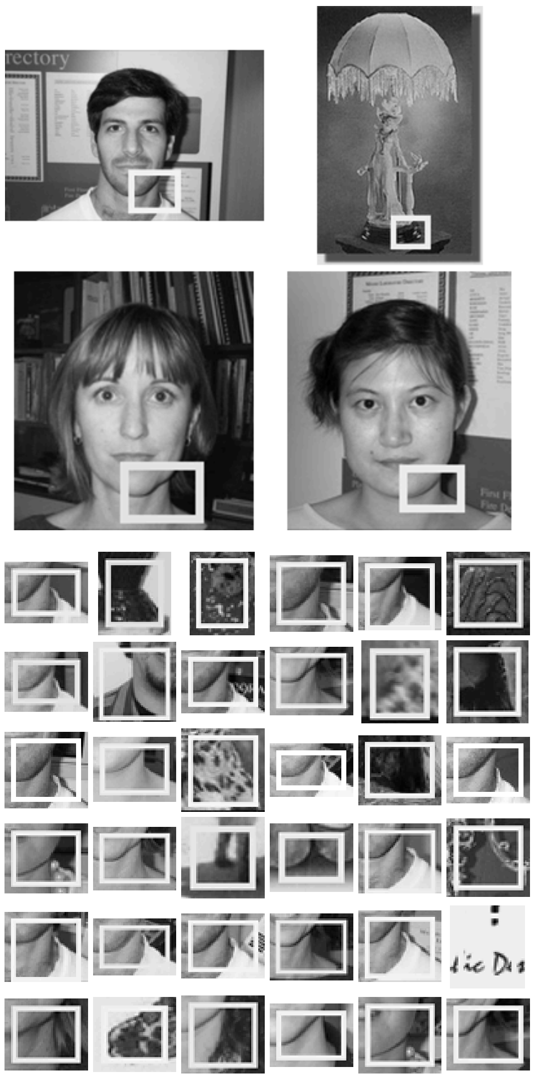
Figure 17. The 40 best image patches for feature #101 (using the same display format as figure 16).