Multiclass Object Recognition Inspired by the Ventral Visual Pathway

by

James Vincent Mutch

B.Sc., The University of British Columbia, 1989

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

October 2006

© James Vincent Mutch, 2006
Abstract

We describe a biologically-inspired system for classifying objects in still images. Our system learns to identify the class (car, person, etc.) of a previously-unseen instance of an object. As the primate visual system still outperforms computer vision systems on this task by a wide margin, we base our work on a model of the ventral visual pathway, thought to be primarily responsible for object recognition in cortex.

Our model modifies that of Serre, Wolf, and Poggio, which hierarchically builds up feature selectivity and invariance to position and scale in a manner analogous to that of visual areas V1, V2, V4, and IT. As in that work, we first apply Gabor filters at all positions and scales; selectivity and 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 type of model in computer vision.
Contents

Abstract .................................................. ii

Contents .................................................. iii

List of Tables ............................................ vii

List of Figures ........................................... viii

Acknowledgments .......................................... x

1 Introduction ............................................. 1
  1.1 Motivation ........................................... 1
  1.2 Scope of Problem Addressed ......................... 3
     1.2.1 Format of Images ................................ 3
     1.2.2 Segmented vs. Unsegmented Training Images .... 4
     1.2.3 Categories ....................................... 4
     1.2.4 Context ......................................... 5
     1.2.5 Types of Classification Tasks ................... 5
  1.3 Contributions .......................................... 5
  1.4 Outline of Thesis .................................... 7
# Background

2.1 The Ventral Visual Pathway

2.1.1 V1 and Topographical Maps

2.1.2 Hierarchical Organization

2.1.3 Immediate Recognition and Feedforward Operation

2.2 Sparsity

2.3 Geometry and "Bags of Features"

# Previous Work

3.1 Other Biologically-Inspired Models

3.1.1 The Neocognitron

3.1.2 Convolutional Networks

3.1.3 "HMAX"

3.1.4 Serre, Wolf & Poggio Model

3.2 Relation to Recent Computer Vision Models

# Base Model

4.1 Model Overview

4.2 Feature Computation

4.2.1 Image Layer

4.2.2 Gabor Filter (S1) Layer

4.2.3 Local Invariance (C1) Layer

4.2.4 Intermediate Feature (S2) Layer

4.2.5 Global Invariance (C2) Layer

4.3 SVM Classifier

4.4 Differences from Serre et al.
List of Tables

6.1 Results for the Caltech 101 dataset along with those of previous studies 42
6.2 Contribution of successive modifications to the overall score 43
6.3 Per-category classification rates and most common errors for the Caltech 101 dataset 45
7.1 Results for the UIUC car dataset along with those of previous studies 48
7.2 Frequency of error types on the multiscale UIUC car dataset 50
9.1 Our results for the Graz-02 datasets 66
# List of Figures

1.1 Some challenges of visual object recognition ........................................... 2
1.2 Example images for the basic classification task ..................................... 6
1.3 Example images for the localization task .................................................. 6

2.1 The ventral visual pathway ......................................................................... 10

3.1 Architecture of the Neocognitron .............................................................. 15
3.2 Architecture of the LeNet-5 convolutional network .................................... 17
3.3 Architecture of the original “HMAX” model .............................................. 19

4.1 Overall form of our base model ................................................................. 24
4.2 Base model layers ..................................................................................... 25
4.3 An S2 feature (prototype patch) in the base model .................................... 28

5.1 Dense vs. sparse S2 features ..................................................................... 32
5.2 Inhibition in S1/C1 ................................................................................... 34
5.3 Limiting the position/scale invariance of C2 units ..................................... 35
5.4 Informative and uninformative features .................................................... 36
5.5 Using an SVM for feature weighting ......................................................... 36

6.1 Some images from the Caltech 101 dataset .............................................. 39
6.2 Results of parameter tuning using the Caltech 101 dataset
6.3 Results on the Caltech 101 dataset for various numbers of features
6.4 Some example images from easier categories
6.5 Some example images from difficult categories

7.1 Some correct detections on the single-scale UIUC car dataset
7.2 The only errors made on the single-scale UIUC car dataset
7.3 Some correct detections on the multiscale UIUC car dataset
7.4 Some errors made on the multiscale UIUC car dataset

8.1 Best image patches for feature #1
8.2 Best image patches for feature #2
8.3 Best image patches for feature #3
8.4 Best image patches for feature #101
8.5 Best image patches for feature #501
8.6 Best image patches for feature #1001
8.7 Feature sharing among categories
8.8 Sizes of features remaining after feature selection
8.9 Relative proportions of feature sizes by selection rank

9.1 Subimages used to train the Graz-02 "bikes" classifier
Acknowledgments

I am greatly indebted to my thesis supervisor David Lowe, without whom none of this would have happened. David has been both a brilliant collaborator and a kind and patient mentor, pointing me in all the right directions and expertly guiding me through many “firsts” – first paper submission, first conference talk, and many others. Thank you, David.

The UBC Computer Science Department as a whole has been an incredibly supportive environment. Special thanks to Rubén Morales-Menéndez, who helped me when I first came back to school, and to the 238 bullpen gang, for their friendship. Thanks to Robert Woodham for much helpful advice, and to Jim Little for being my second reader. Finally, thanks to David Kirkpatrick, for giving me my first book on AI all those years ago, and for encouraging me to come back.

I’m grateful for the lifelong support of my parents, who never failed to encourage my curiosity about the world.

To all the friends and family whom I’ve neglected during this thesis – thanks for your love and understanding.

Finally, my thanks and love to Rachel, who has borne the burden of my obsessiveness and shown me endless patience, understanding, and support.
1 Introduction

1.1 Motivation

Computer vision systems have become quite good at recognizing specific objects they have seen before. This is not an easy task, as it demands a difficult combination of specificity and invariance. The system must give very different responses to stimuli which are often superficially similar, such as two different faces. Yet it must give the same response to two different views of the same object, despite potentially large differences in viewpoint or illumination, presence of background clutter and partially occluding objects, and other factors (see figure 1.1). Nevertheless systems such as [23] are now capable of recognizing previously-seen objects – particularly rigid objects whose shapes do not change much – fairly accurately and at real-time speed, under a variety of conditions.

However, the generalization from object instances to object categories remains much more difficult for computers than it is for human vision. Computer vision systems that are good at recognizing specific, previously-seen objects tend to look for very distinctive low-level visual features having rigid geometric relationships. Various attempts to soften these constraints to span entire categories of objects while still retaining the necessary specificity have met with only partial success.
Figure 1.1: Some challenges of visual object recognition: multiple viewpoints, varying degrees of illumination, background clutter, and partial occlusion.
Given the still vastly superior performance of humans (and animals) on this task, it makes sense to look to the ever-increasing body of neuroscientific and psychophysical data for inspiration. In fact, recent work by Serre, Wolf, and Poggio [37] has shown that a computational model based on some of our current knowledge of visual cortex can be at least competitive with the best existing computer vision systems on some of the standard classification datasets. Our work builds on this model and improves its performance.

Our ultimate goal is to give computers human-level ability to learn to visually classify objects. The number of potential applications is virtually limitless, from image retrieval to allowing robots to interact more fully with their environments. The success (or failure) of vision systems inspired by our knowledge of the brain may also help add to that knowledge.

1.2 Scope of Problem Addressed

1.2.1 Format of Images

Most research in this area (including this study) focuses on the core problem of object classification in single 2D grayscale images (i.e., still “black and white” photographs).

A few object categories are almost impossible to distinguish without colour, such as lemons and limes. For other categories colour helps: most trees are green, a purple object is usually not a face, etc. But for many categories, such as cars, colour is just a distraction, and systems that incorporate it require larger amounts of training data in order to come to the conclusion that colour, for such categories, is not relevant. Moreover, humans do very well without colour cues. We thus focus
on the core problem of classification from shape and texture.

Recognition from motion cues is also an important component of real-world vision, and forms another area of active research, but it is outside the scope of this study.

1.2.2 Segmented vs. Unsegmented Training Images

Systems are generally trained on a set of labeled images containing exemplars of object categories; they are then expected to recognize and label new instances of those categories in other images.

Some computer vision systems can learn from completely unsegmented images, knowing only that some images contain an object of type X (but not where) and some do not. Others require training images having bounding boxes or precise outlines. We focus on learning from segmented images because it is more analogous to human visual learning. It is fairly clear that we learn about objects upon which our attention has already been focused by some other mechanism, possibly aided by motion, stereo, or colour cues. We wish to simply assume the existence of such a mechanism during training by using segmented images, or at least, images in which the object is central and dominant.

1.2.3 Categories

In reality, different object categories can have complex semantic relationships. These include parent-child "is-a" relationships (e.g., a robin is a bird) and whole-part "has-a" relationships (e.g., a cougar has a head). We ignore this for purposes of this study and treat categories as disjoint, unrelated sets, defined simply by the labels we give to our training images.
1.2.4 Context

Object classification in humans can be aided by context (the entire scene or other objects) or other prior beliefs. For example, street scenes prime us to see cars. A white blob next to a computer keyboard is probably a mouse. These kinds of top-down, scene-level effects are outside the scope of this study.

1.2.5 Types of Classification Tasks

The ultimate goal in object recognition is to be able to locate and classify every object in a scene. In this study we focus on two simpler tasks.

Basic Classification: What is this object? The answer can be one of many categories, but there is only one object, which does not have to be found in a larger image; see figure 1.2. It either dominates the image or is contained in an already-chosen attentional window. These experiments are described in chapter 6.

Localization: Where are all the objects of type X in this image, if any? Here the simplification is to reduce the number of categories to one, but the instance(s) of that category can be anywhere in the image; see figure 1.3. These experiments are described in chapter 7.

1.3 Contributions

The model presented in this paper is based on the “standard model” of object recognition in cortex [31] and builds on the “HMAX”-based model of Serre, Wolf, and Poggio [37]. We incorporate some additional biologically-motivated properties
Figure 1.2: Example images for the basic classification task. The goal is to determine the category of the object (airplane, elephant, etc.) which is centrally located and dominant.

Figure 1.3: Example images for the localization task. The goal is to locate all objects of a given single category. The white rectangles are the outputs of this task.
including sparsification of feature inputs, lateral inhibition, feature localization, and feature selection (see chapter 5). We show that these modifications further improve classification performance, strengthening our understanding of the computational constraints facing both biological and computer vision systems.

We test these modifications on the large Caltech dataset of images from 101 object classes (chapter 6). Our results show that there are significant improvements to classification performance from each of the changes. Further tests on the UIUC car database (chapter 7) demonstrate that the resulting system can also perform well on object localization. Our final system outperformed all previous studies involving these datasets, further strengthening the case for incorporating concepts from biological vision into the design of computer vision systems.

1.4 Outline of Thesis

The rest of the thesis is structured as follows.

- Chapter 2 reviews some of the ideas from biological and computer vision upon which our model is based.
- Chapter 3 discusses some of the previous biologically-motivated computational models for object classification.
- Chapter 4 describes our “base” model, which is essentially an abstraction of [37].
- Chapter 5 describes our enhancements to the base model including sparsification of feature inputs, inhibition, limited position/scale, invariance of intermediate-level features, and feature selection. These represent the main
contributions of this work.

- Chapter 6 describes the training and tuning of our model on the Caltech 101 dataset and presents the results for the basic classification task.

- Chapter 7 describes our localization experiments on the UIUC car dataset and presents the results.

- Chapter 8 provides some insight into the kinds of features that are being selected.

- Chapter 9 describes experiments on the Graz datasets, in which the images are somewhat more difficult.

- Chapter 10 summarizes the results and discusses possible future work.
2 Background

2.1 The Ventral Visual Pathway

Object classification in cortex is believed to be centered in the ventral visual pathway, running from primary visual cortex (V1) through areas V2, V4, and inferotemporal cortex (IT).

2.1.1 V1 and Topographical Maps

Among all the visual areas V1 has received the most study (although it is far from being fully understood [27]). Cells in V1 respond to very simple features (essentially oriented bars) at specific retinal positions and scales [15].

V1 is one of many cortical areas with a clear topographic mapping: its 2D layout corresponds roughly to 2D retinotopic position. The additional stimulus dimensions of scale and bar orientation are folded into this 2D layout so that a given bit of V1 cortex will contain cells responsive to a variety of scales and orientations at a specific retinotopic position. This packing of multiple stimulus dimensions into a 2D array is done in a manner that maintains continuity along stimulus dimensions while covering the entire stimulus space [38]. Note that V1 contains additional stimulus dimensions (colour, stereo, motion) that we ignore in this study.

Topographical maps are numerous in cortex: there are about 25 of them in
the visual system alone [5]. There are also topographical maps for body location in
the somatosensory system, frequency in the auditory system, and muscle groups in
the motor system. Continuous topographical maps minimize the amount of wiring
necessary for performing local computations. Neurons representing similar stimuli
are kept close together.

2.1.2 Hierarchical Organization

As we move through the ventral stream from V1 through V2, V4, and IT, we
encounter cells that are responsive to increasingly complex stimuli with increasing
invariance to position and scale. The first step seems to occur within V1 itself,
where the outputs of a number of "simple" cells responsive to the same feature (i.e.,
a particular orientation) are pooled by "complex" cells [15]. A resultant complex
cell is responsive to the same orientation as its inputs but has a larger receptive
field, i.e., a greater degree of position and scale invariance. Further steps combine
heterogeneous features to generate more complex features, and more pooling over
position and scale occurs. At the level of IT we encounter cells with a high degree
of position and scale invariance which are responsive to specific object views, as well
as viewpoint-invariant units.

Despite the increasing degree of spatial invariance at higher levels, combinatorical constraints on the number of complex features that could be represented rule out complete coverage of the potential stimulus space, as seems to occur in V1. This suggests an increasing role for ongoing learning at higher levels, and certainly at the level of IT.

This hierarchical arrangement of topographical maps is not unique to the visual system; similar hierarchies are present in the somatosensory, auditory, and motor systems [5].

2.1.3 Immediate Recognition and Feedforward Operation

Most forward projections between brain areas are matched by corresponding feedback connections, and the ventral stream is known to be modulated by other areas for reasons including attention and contextual priming. Nevertheless we seems to do quite well in an “immediate” recognition mode in which these effects are absent. Human subjects in rapid serial visual presentation (RSVP) experiments have been able to process images as rapidly as 8 per second [29]. EEG studies [39] show response times on the order of the latency of the ventral stream, suggesting a mainly feedforward mode of operation for this first stage of the basic classification task.

This constraint, more than any other, suggests that computational models of biological object classification might be workable despite the current limitations of our knowledge. Studies of the response properties of neurons at various levels in the hierarchy are growing more and more numerous. If we know what is being computed at each level, the feedforward constraint makes it easier to guess how it is being computed.
The feedforward constraint, however, does not apply to learning.

## 2.2 Sparsity

Representations in visual cortex are known to be *overcomplete*, i.e., data is represented using a much larger set of basis functions (neurons) than would be minimally necessary. For example, in macaque V1 there are 50 times as many output fibres as input fibres. However, if the decomposition of a signal into such an overcomplete code were done linearly, the components of the resulting vector would be correlated [27]. Some form of nonlinearity is required to *sparsify* them. Sparse vectors are vectors whose components are mostly zeros. In general, sparse codes are more metabolically efficient and more easily stored in associative memories [2]. Sparsity constraints have proven critical for explaining the response functions of V1 neurons in terms of the statistics of natural images [26].

A direct way to achieve sparsification is for cells to have nonlinear, highly peaked response functions. An alternative method is lateral inhibition, in which cells inhibit their less-active neighbours in a winner-take-all competition – the continuous topographical map organization of many cortical areas (section 2.1.1), in which cells encoding similar stimuli are kept close together, is ideal for this.

Furthermore, within machine learning, it has been found that increasing the sparsity of inputs [9, 17] (equivalent to reducing the capacity of the classifier) plays an important role in improving generalization performance.

Chapter 5 describes our efforts to incorporate these concepts into our classification model.
2.3 Geometry and “Bags of Features”

Some current successful computer vision systems for object classification learn and apply quite precise geometric constraints on feature locations [8, 4], while others ignore geometry and use a “bag of features” approach that ignores the locations of individual features [6].

However, in a hierarchical model, 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, this ceases to be a binary decision. The question becomes: at what level have features become complex enough that we can ignore their location?

If the features are too simple, we run the risk of “binding” problems: we are still vulnerable to false positives due to chance co-occurrence of features from different objects and/or background clutter. Conversely, the classification system would be unable to distinguish between an instance of a known category and a scrambled version of the same image. However, if the features are sufficiently complex and large enough to overlap, then random rearrangements of the image would destroy enough features to avoid the false positive.

In this study we investigated retaining some degree of position and scale sensitivity at a higher point in this hierarchy than the approach of [37], and show that this provides a significant improvement in final classification performance. Chapter 5 outlines our approach.
3 Previous Work

In this chapter we review other biologically-inspired models of object classification and discuss our approach in the context of some of the recent computer vision approaches.

3.1 Other Biologically-Inspired Models

This section reviews other biologically-inspired models, specifically, models of feedforward recognition in the ventral stream that hierarchically build up feature complexity and invariance.

3.1.1 The Neocognitron

The Neocognitron [12] was the first of this class of models. Generalizing from the "simple" and "complex" cells of Hubel & Wiesel [15], the Neocognitron starts with a 2D pixel layer and then computes alternating "S" and "C" layers (S1, C1, S2, C2, ...). "S" layers build up feature complexity and "C" layers build up position invariance. Its architecture is illustrated in figure 3.1.

Each Sn layer processes the previous layer and computes \( d_n \) feature maps; each map is the response to a particular type of local feature computed at all possible positions in the previous layer. In layer S1, local features are computed directly from
Figure 3.1: Architecture of the Neocognitron. Each layer consists of some number of square feature maps. The S1 layer's feature maps are computed from the image. Each feature map in a “C” layer is created by pooling units from one feature map in the previous “S” layer (or sometimes two in symmetry cases). However, “S” features in S2 or higher are computed by combining features from multiple maps in the previous layer. This is illustrated by the crossing connections from “C” to “S” layers. From [12].
pixels. In higher “S” layers, features are computed as local combinations of different types of cells (from different feature maps) in the previous “C” layer.

A C-cell’s value is a local weighted sum of a patch of S-cells of the same type (i.e., from the same feature map) in the previous layer. “C” layers also serve to reduce the total number of units by subsampling their input “S” layer.

At the top level, cells are complex enough to represent entire object categories, and are completely position invariant. Classification is performed by selecting the most active top-level cell.

Learning in the Neocognitron means learning what features to compute in the “S” layers. This is typically done in a bottom-up manner. The S1 layer is trained to find common or useful patterns in the pixel layer, then the S2 layer is trained to find patterns in the C1 layer, and so on. Clustering methods are often used.

The Neocognitron was invented for handwritten character recognition, but has been adapted for other 2D pattern classification tasks. It is not explicitly multiscale; patterns must be of a standard size.

3.1.2 Convolutional Networks

The Neocognitron was essentially the first convolutional network. In its regular, feedforward (post-learning) mode, its basic operation is convolution. An “S” layer is generated by convolving the previous layer with \( d \) local filters, while each feature map of a “C” layer is generated from the corresponding map in the previous “S” layer via convolution with a fixed local filter.

The term “convolutional network” now seems to refer to a network having this same basic structure, but with two major differences.
Training is top-down, via backpropagation [20].

The top-level features do not represent object categories. Instead, the vector of activations is fed into a classifier. This can be a standard, fully-connected multilayer neural network or some other classifier.

Convolutional networks such as LeNet-5 (figure 3.2) have been applied to commercial-level character recognition, speech recognition, and face/object recognition.

3.1.3 “HMAX”

In 1999 Riesenhuber & Poggio formulated the “standard model” of object recognition in cortex, essentially defining a class of models consistent with the following mostly agreed-upon facts regarding the ventral visual pathway (from [30]):

- “A hierarchical build-up of invariances first to position and scale and then to viewpoint and more complex transformations requiring the interpolation between several different object views;

- in parallel, an increasing size of the receptive fields;
• an increasing complexity of the optimal stimuli for the neurons;

• a basic feedforward processing of information (for "immediate" recognition tasks);

• plasticity and learning probably at all stages and certainly at the level of IT;

• learning specific to an individual object is not required for scale and position invariance (over a restricted range)."

They also created a quantitative model [31], later dubbed “HMAX”, which embodied some of these concepts. The basic HMAX model is not an end-to-end object classification system; it was designed to account for the tuning and invariance properties of neurons in IT cortex found by the experiments of Logothetis et al. [22].

The HMAX model is similar to the convolutional networks mentioned above in that it uses alternating “S” and “C” layers to build up feature complexity and invariance; see figure 3.3. Top-level view-trained units are learned vectors of activations of fully position/scale-invariant features. However, HMAX differs in the following ways:

• Rather than learning the low-level (S1) features, HMAX starts with statically defined features (Gaussian derivatives or Gabor filters) designed to mimic cells in V1 cortex.

• C-level pooling uses a MAX operation instead of a weighted sum; the output of a C-cell is that of its strongest input. This increases position and scale invariance without losing any feature specificity. Support for a MAX operation in at least some cells in visual cortex has been found in physiological studies [18].
Figure 3.3: Architecture of the original "HMAX" model. From [31].

- HMAX is explicitly multiscale. As in V1 cortex, the S1 layer applies filters at a range of scales. Higher C-units pool not only over local positions but also over nearby scales.

The original HMAX model had four fixed feature layers (S1, C1, S2, and C2) – none were learned. The relatively small number of simple features at the C2 level were insufficiently complex or distinct for object recognition tasks [35].
3.1.4 Serre, Wolf & Poggio Model

Serre et al. [37] modified the original HMAX model to make it useful for classification. The static S2 features were replaced by a much larger set of features sampled from training images, and the final vectors of C2 activations were fed into a support vector machine (SVM) classifier. The other layers remained mostly the same, although the exact S1 features and C1 pooling ranges were adjusted to more closely match physiological data [36]. The Serre model is described in chapter 4 by contrast with our base model.

This model achieved results comparable to the best non-biologically motivated approaches on the difficult Caltech 101 dataset.

3.2 Relation to Recent Computer Vision Models

The above-mentioned systems represent the most direct attempts to model object classification in the primate ventral stream. Nevertheless, many other computer vision systems draw analogies to aspects of biological vision, and all face the same challenges.

Virtually all current approaches to object classification start by extracting various kinds of local features from the image; object category representations are then learned and expressed in terms of the presence or activity level of these features. Common choices for features include fragments composed of raw pixels [41, 40] and SIFT descriptors [23]. They may be sparsely sampled at certain interest points where some local saliency condition is satisfied, or they may be densely computed at every point in the image. Recent work by Jurie & Triggs [16] suggests that sparse sampling at interest points is suboptimal for classification tasks, as too much
discriminative information is lost. Like other biologically-motivated approaches, our model uses the dense method.

Early object classification systems focused on a single object category at a time, learning a set of features that best distinguished that category from the background, i.e., from general clutter and all other object categories. As systems scale up towards the thousands of categories humans can recognize, having a separate set of features for each object category is clearly not feasible. Torralba et al. [40] employ a multiclass version of boosting to learn a set of shared features, and find that the number of features needed for a given level of performance scales roughly logarithmically with the number of categories. All the biologically-inspired models of object classification are shared-feature models.

The issue of feature selection is also explored in models such as that of Ullman et al. [41], which addresses the optimal complexity of features for classification tasks. Highly complex features may be extremely distinctive, but will not occur often enough to be useful; conversely, very simple features may occur frequently but are often not distinctive. For a single-class-vs.-background task, they found that features of "intermediate complexity" (around 10% of object size in their experiments) maximized mutual information. We address a similar issue for our model in chapter 8.

Higher up, at the level of category representations, there are a number of approaches. Constellation models such as that of Fergus et al. [8] encode explicit geometric relationships between object parts, while "bag of features" methods such as those of Csurka et al. [6] and Opelt et al. [28] represent objects as vectors of feature activations, discarding geometric relationships above the feature level. There are a number of approaches in between. The features of Agarwal et al. [1]
retain some coarsely-coded location information, while Leibe & Schiele [21] and Berg et al. [3] retain the locations of features relative to the object center. In a biologically-inspired feature hierarchy, spatial structure is gradually incorporated into the features themselves. In each “S” layer, there are loose geometric constraints on which features may be combined. Moving upward, as spatial information becomes implicitly encoded into more complex, overlapping features of varying sizes, explicit spatial information becomes more coarsely coded.
4 Base Model

This chapter describes the "base" version of our model. The base model is similar to [37] and performs about as well; nevertheless, it is an independent implementation, and we give its complete description here. Its differences from [37] will be listed briefly at the end of this chapter (section 4.4). Larger changes, representing the main contribution of this work, are described in chapter 5.

4.1 Model Overview

The overall form of the model (shown in figure 4.1) 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 model's biological plausibility lies in the feature computation stage.

4.2 Feature Computation

Features are computed in 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 4.2, and the following

\footnote{An abbreviated version of chapters 4-7 has been published as a conference paper [25].}
sections describe each layer. Note that features in all layers are computed at all positions and scales – interest point detectors are not used.

4.2.1 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).

4.2.2 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 [37] 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 4.2). 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 can be described by:

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

(4.1)
Figure 4.2: Base model layers. 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.
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 \( \theta \) varies between 0 and \( \pi \). The parameters \( \gamma \) (aspect ratio), \( \sigma \) (effective width), and \( \lambda \) (wavelength) are all taken from [37] 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:

\[
R(X, G) = \frac{\left| \sum X_i G_i \right|}{\sqrt{\sum X_i^2}} \quad (4.2)
\]

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. Nevertheless, experiments using much larger arrays did not show any effect on overall system performance.

### 4.2.3 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

\(^2\)Note that the max filter is itself a pyramid, so its size is 10x10 only at the lowest scale.
smaller in spatial extent and has the same number of feature types (orientations) as S1; see figure 4.2. This layer provides a model for V1 complex cells.

4.2.4 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 [37].) 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 figure 4.3). 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. 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
Figure 4.3: An S2 feature (prototype patch) in the base model. A 4x4 prototype patch is shown. Each prototype is sampled from the C1 layer of a training image at a random position and scale. For each position within the prototype, there are C1 values for each of the four orientations. Stronger C1 values are shown as darker.

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 4.2. 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:

\[
R(X, P) = \exp \left( -\frac{\|X - P\|^2}{2\sigma^2\alpha} \right) \tag{4.3}
\]

Both \( X \) and \( P \) have dimensionality \( n \times n \times 4 \), where \( n \in \{4, 8, 12, 16\} \). As in [37], the standard deviation \( \sigma \) is set to 1 in all experiments.

The parameter \( \alpha \) 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.

### 4.2.5 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”.

### 4.3 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.

### 4.4 Differences from Serre et al.

Our base model, as described above, performs only as well as that of Serre et al. in [37], despite several changes that one might expect to improve performance:

- We scale the smaller edge of each image to 140 pixels, as opposed to always scaling the height to 140. We thus avoid making tall "portrait" images very thin.

\(^3\)We use the Statistical Pattern Recognition Toolbox for Matlab [10].

\(^4\)Suggested by T. Serre (personal communication).
• Scales in our image pyramid differ multiplicatively. [37] does not use an image pyramid, but rather applies different-sized S1 filters to the full-scale image. However, these S1 filters differ additively in size. Hence the larger-scale C1 units in [37] have very little scale invariance because they pool S1 units computed using nearly identically sized filters.

• Our C1 subsampling ranges overlap in scale as well as in position. This should make the system more robust to small changes, but has no noticeable effect on the end result.

• We introduce the α parameter (section 4.2.4) to avoid favoring the smallest (4x4) S2 features. One reason this might not be helping is that it appears (in chapter 8) that 4x4 features are more suited to the task at hand anyway.

Our system also contains a simplification that results in no appreciable performance loss. The S1 filter parameters σ and λ in [37] change from scale to scale, in accordance with physiological studies [36]. We find using the same parameters for all scales makes little difference for our purposes.
This chapter describes four improvements to the base model, representing the main contribution of this work. The improvements are:

1. Sparsification of inputs to S2 units.
2. Inhibition of S1 and C1 unit outputs.
3. Limiting the position and scale invariance of C2 units.
4. Selecting the best S2 features.

Testing results for each modification are provided in chapter 6.

5.1 Sparser 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 their 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
Figure 5.1: Dense vs. sparse S2 features. Dense S2 prototypes in the base model are sensitive to all orientations of C1 units at each position. Sparse S2 prototypes 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.

In conjunction with this we increase the number of Gabor filter orientations in S1 and C1 from 4 to 12. Since we are 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].
5.2 Inhibited 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 process is illustrated in figure 5.2.

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.

This enhancement, together with the increased number of orientations (section 5.1), gives us a sparse, overcomplete code in S1 and C1.

5.3 Limited C2 Position/Scale Invariance

Above the S2 level, the base model becomes a “bag of features” [6], disregarding all geometry. The C2 layer simply takes 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 binding problems: we are still vulnerable to false positives due to chance co-occurrence of features from different objects and/or
Figure 5.2: Inhibition in S1/C1. Before inhibition, the circled unit in the prototype patch is getting some response to its desired orientation, despite the fact other orientations dominate. Inhibition increases the distance to prototype patches looking for non-dominant orientations.

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 [33]. 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 5.3.

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 localization of objects within complex scenes, as in the UIUC car database, we augment it with
Figure 5.3: 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).

a search for peak responses over object location using a sliding window.

5.4 Feature Selection

Our S2 features are prototype patches randomly selected from unsegmented training images. Many will be from the background (figure 5.4), 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 [24]. In fitting separating hyperplanes, the SVM is essentially doing feature weighting (see figure 5.5). Our all-pairs m-class linear SVM consists of \( m(m - 1)/2 \) binary SVMs.
Figure 5.4: Informative and uninformative features. An S2 feature sampled from the top right of this training image is not likely to be useful for classification.

Figure 5.5: Using an SVM for feature weighting. In this simple 2d binary SVM, feature 2 is clearly more useful in separating the classes than is feature 1.

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\textsuperscript{1} 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 setting their weights to zero rather than those assigned by the SVM) improves classification performance, and the resulting model is more economical to compute.

\textsuperscript{1}Depending on the desired number of features it may be necessary to drop less than half per round.
6 Multiclass Experiments - Tuning and Performance on the Caltech 101 Dataset

In this chapter we describe both the tuning of model parameters and the performance of the final model on the basic classification task.

Each modification described in chapter 5 has at least one free parameter (number of orientations, degree of inhibition, allowed position/scale variation, and number of features). Using subsets of the Caltech 101 dataset, we arrive at robust values for each of these parameters. We then evaluate the final model’s performance on the full dataset.

6.1 The Caltech 101 Dataset

The Caltech 101 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 this dataset ideal for testing basic classification on a large number of categories. (It has become the unofficial
standard benchmark for this task.) Some sample images are shown in figure 6.1.

6.2 Running the Model

To run our model for the basic classification task, the experimenter first defines a set of parameters including:

- the number of training images per category (15 or 30 for the Caltech 101),
- which modifications (chapter 5) are turned on, and
- parameter values for those modifications (number of orientations, degree of inhibition, etc.)

The system then:

1. chooses the desired number of training images at random from each category, placing remaining images in the test set,
2. learns features at random positions and scales from the training images (an equal number from each image),
3. builds C2 vectors for the training set,
4. trains the SVM (performing feature selection if that option is turned on),
5. builds C2 vectors for the test set, and
6. classifies the test images.

6.3 Parameter Tuning

The complete parameter space for the full set of modifications 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 6.2). 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.
Figure 6.2: 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 6.3: 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.
Table 6.1: Our results for the Caltech 101 dataset along with those of previous studies. Scores 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>15 training images/cat.</th>
<th>30 training images/cat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our model (base)</td>
<td>33</td>
<td>41</td>
</tr>
<tr>
<td>Serre et al. [37]</td>
<td>35</td>
<td>42</td>
</tr>
<tr>
<td>Holub et al. [14]</td>
<td>37</td>
<td>43</td>
</tr>
<tr>
<td>Berg et al. [3]</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>Grauman &amp; Darrell [13]</td>
<td>49.5</td>
<td>58.2</td>
</tr>
<tr>
<td>Our model (final)</td>
<td>51</td>
<td>56</td>
</tr>
</tbody>
</table>

The results of parameter tuning are shown in figures 6.2 and 6.3. The chosen parameters were 12 orientations, $h = 0.5$, $t_p = \pm 5\%$, $t_s = \pm 1$ scale, 1,500 features.

6.4 Multiclass Performance

Table 6.1 shows the performance of our base and final models and compares them with results of previous studies. Each result is the average of 8 independent runs. Our final results for 15 and 30 training images are 51% and 56%.

Note that subsequent work by other groups [19, 42] has exceeded this performance level. These studies use improved kernels for the SVM classifier (as does Grauman & Darrell [13]). It will be interesting to see whether these ideas can be successfully combined with our sparse image features to get further improvements.

Table 6.2 shows the contribution to performance of each successive modification.

Figure 6.4 contains some examples of categories for which the system performed well, while figure 6.5 illustrates some difficult categories. In general, the harder categories are those having greater shape variability due to greater intra-class variation and nonrigidity. Interestingly, the frequency of occurrence of background
Table 6.2: The contribution of our successive modifications to the overall classification score. 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>15 training images/cat.</th>
<th>30 training images/cat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>33</td>
<td>41</td>
</tr>
<tr>
<td>+ sparse S2 inputs</td>
<td>35 (+ 2)</td>
<td>45 (+ 4)</td>
</tr>
<tr>
<td>+ inhibited S1/C1 outputs</td>
<td>40 (+ 5)</td>
<td>49 (+ 4)</td>
</tr>
<tr>
<td>+ limited C2 invariance</td>
<td>48 (+ 8)</td>
<td>54 (+ 5)</td>
</tr>
<tr>
<td>+ feature selection</td>
<td>51 (+ 3)</td>
<td>56 (+ 2)</td>
</tr>
</tbody>
</table>

Table 6.3 shows the classification rate and most common error for each category. 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 (vaguely similar flowers).

Figure 6.4: Some example images from easier categories.

clutter in a category's images does not seem to be a significant factor.

Figure 6.5: Some example images from difficult categories.
The background category is the least recognized of all. 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 only 30 exemplars.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Category</th>
<th>Rate &amp; StDev</th>
<th>Most Common Error &amp; Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>car side</td>
<td>98.39</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Faces easy</td>
<td>97.87</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Motorbikes</td>
<td>96.81</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>minaret</td>
<td>94.02</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Faces</td>
<td>93.52</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>trilobite</td>
<td>89.96</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>airplanes</td>
<td>89.85</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>grand piano</td>
<td>86.23</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>yin yang</td>
<td>85.42</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>laptop</td>
<td>84.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>revolver</td>
<td>79.09</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>menoraah</td>
<td>77.63</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>ferry</td>
<td>69.59</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>dragonfly</td>
<td>69.41</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>euphonium</td>
<td>68.38</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>ketch</td>
<td>67.56</td>
<td>schooner 17.11</td>
</tr>
<tr>
<td>17</td>
<td>stop sign</td>
<td>66.54</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>soccer ball</td>
<td>66.54</td>
<td>nautilus 5.88</td>
</tr>
<tr>
<td>19</td>
<td>electric guitar</td>
<td>66.11</td>
<td>bass 5.83</td>
</tr>
<tr>
<td>20</td>
<td>schooner</td>
<td>65.15</td>
<td>ketch 19.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>watch</td>
<td>64.65</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>joshua tree</td>
<td>63.97</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>buddha</td>
<td>62.50</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>umbrella</td>
<td>61.94</td>
<td>lamp 5.28</td>
</tr>
<tr>
<td>25</td>
<td>Leopards</td>
<td>59.93</td>
<td>crocodile 5.44</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>ewer</td>
<td>58.18</td>
<td>strawberry 5.00</td>
</tr>
<tr>
<td>27</td>
<td>brain</td>
<td>58.09</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>dalmatian</td>
<td>53.72</td>
<td>garfield 5.07</td>
</tr>
<tr>
<td>29</td>
<td>helicopper</td>
<td>53.23</td>
<td>cannon 5.17</td>
</tr>
<tr>
<td>30</td>
<td>cougar face</td>
<td>52.88</td>
<td>flamingo head 5.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>chair</td>
<td>50.78</td>
<td>windsor chair 5.08</td>
</tr>
<tr>
<td>32</td>
<td>hawksbill</td>
<td>50.54</td>
<td>crab 5.71</td>
</tr>
<tr>
<td>33</td>
<td>chandelier</td>
<td>50.00</td>
<td>scissors 5.36</td>
</tr>
<tr>
<td>34</td>
<td>bonsai</td>
<td>49.74</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>lamp</td>
<td>49.60</td>
<td>flamingo 6.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>sunflower</td>
<td>49.55</td>
<td>starfish 5.68</td>
</tr>
<tr>
<td>37</td>
<td>butterfly</td>
<td>48.98</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>elephant</td>
<td>48.90</td>
<td>brontosaurus 8.46</td>
</tr>
<tr>
<td>39</td>
<td>crayfish</td>
<td>45.31</td>
<td>lobster 7.50</td>
</tr>
<tr>
<td>40</td>
<td>dolphin</td>
<td>42.14</td>
<td>bass 5.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>llama</td>
<td>39.84</td>
<td>kangaroo 6.51</td>
</tr>
<tr>
<td>42</td>
<td>kangaroo</td>
<td>38.84</td>
<td>sea horse 6.47</td>
</tr>
<tr>
<td>43</td>
<td>flamingo</td>
<td>38.18</td>
<td>emu 6.42</td>
</tr>
<tr>
<td>44</td>
<td>lotus</td>
<td>37.50</td>
<td>water lily 18.75</td>
</tr>
<tr>
<td>45</td>
<td>starfish</td>
<td>34.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>crab</td>
<td>30.81</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>ibis</td>
<td>28.25</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>scorpion</td>
<td>25.00</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>BACKGROUND</td>
<td>12.17</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3: Per-category classification rates and most common errors for 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 shown. The most common error is shown only if it occurs at least 5% of the time.
7 Localization Experiments
(UIUC Car Dataset)

This chapter describes our experiments on the UIUC car dataset [1], in which we used our final, tuned model to tackle the single-class localization problem. These experiments served two purposes.

- Our introduction of limited C2 invariance (section 5.3) sacrificed full invariance to object position and scale within the image; we wanted to see if we could recover it.

- We wanted to demonstrate that the model, and the parameters learned during the tuning process, worked equally well on another dataset.

7.1 The UIUC Car 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.
7.2 Model Parameters

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 5.4 for details.

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

7.3 Sliding Window

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, although at larger scales there are fewer pixels, each representing more of the image. Hence there are fewer window positions at larger scales.

For efficiency reasons, levels S1, C1, and S2 are computed once for the entire image. Then a C2 vector is computed for each position of the sliding window. When computing the C2 vector, the scope of the C2 layer’s MAX operation is limited to the sliding window, and feature position and scale ranges (section 5.3) are considered relative to the window frame.

Duplicate detections were consolidated using the neighborhood suppression algorithm from [1]. We increase the width of a “neighborhood” from 71 to 111 pixels.
<table>
<thead>
<tr>
<th>Model</th>
<th>Single-scale</th>
<th>Multiscale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agarwal et al. [1]</td>
<td>76.5</td>
<td>39.6</td>
</tr>
<tr>
<td>Leibe et al. [21]</td>
<td>97.5</td>
<td></td>
</tr>
<tr>
<td>Fritz et al. [11]</td>
<td>87.8</td>
<td></td>
</tr>
<tr>
<td>Our model</td>
<td><strong>99.94</strong></td>
<td><strong>90.6</strong></td>
</tr>
</tbody>
</table>

Table 7.1: 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].

to avoid merging adjacent cars.

### 7.4 Results

Our results are shown in table 7.1 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 7.2 shows the only errors from the 8th run; figure 7.1 shows some correct single-scale detections.

For the multiscale tests, the sliding window also searches through scale, and the scoring criteria include a scale tolerance (from [1]). Figures 7.3 and 7.4 show some correct detections and some errors on the multiscale set. Table 7.4 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
Figure 7.1: Some correct detections from one run on the single-scale UIUC car dataset.

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 7.4, bottom left. The neighbourhood suppression algorithm eliminates one of them. Since the detector itself is the main focus of this work, this kind of error is not a great concern.

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

Figure 7.3: Some correct detections from one run on the multiscale UIUC car dataset.

<table>
<thead>
<tr>
<th>Source of error</th>
<th>Number of test images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple false positive</td>
<td>1</td>
</tr>
<tr>
<td>Simple false negative</td>
<td>1</td>
</tr>
<tr>
<td>Suppression due to overlap</td>
<td>6</td>
</tr>
<tr>
<td>Detection at wrong scale</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 7.2: Frequency of error types for one run on the multiscale UIUC car dataset.
Figure 7.4: 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.
8 Analysis of Features

In this chapter we take a closer look at the kinds of features that are being selected.

8.1 The Value of S2 Features

Most of this model's biological plausibility is in the way the S2 features are being computed: starting with Gabor filters and then building up invariance and complexity, using lateral inhibition along the way, etc. This raises an important question. How much of the model's performance is due to these particular features, and how much is due to the rest of the model – i.e., the large number of features, the powerful SVM classifier, and the feature selection step?

To test this, we trained a "stub" version of the model on the Caltech 101 dataset. In the stub version, layer S2 is computed directly from the image layer; see figure 4.2. Feature learning is performed by sampling patches of pixels. When computing responses to an image, S2 prototypes (now just simple image patches) are compared to candidate patches using normalized cross-correlation.

Note that without the intervening C1 layer, which was performing subsampling, the number of units in S2 would be much larger than in the full model. To keep the number comparable, we tried two different approaches.

1. Subsampling the image down to the size of the full model's C1 layer.
2. Increasing the size of prototype patches and moving them across the image in larger steps.

The best classification rate for the stub model was 37% (for 30 training images per category), down from 56% for the full model. This is strong evidence that the S2 features in the full model really are doing something useful.

8.2 Visualizing Features

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 8.1 through 8.6 show exactly this for a particular run on the Caltech 101 dataset.

We ranked the 1,500 features which survived the feature selection step (section 5.4) by their average weight across all binary SVMs – the same criterion by which selection was performed. Figure 8.1 shows the 40 patches to which feature #1 responds most strongly. Feature #1 is the feature with the highest average weight – the most informative feature overall under our selection scheme. Figures 8.2 through 8.6 show the strongest 40 patches for some other features.

For most features, these 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.
Figure 8.1: Best image patches for feature #1, from one run on the Caltech 101 dataset. The top left patch represents the highest response.
Figure 8.2: Best image patches for feature #2, from one run on the Caltech 101 dataset. The top left patch represents the highest response. Note that this feature seems to be responding to a strong edge introduced into this category by artificial rotation of the images – an unfortunate flaw of this dataset.
Figure 8.3: Best image patches for feature #3, from one run on the Caltech 101 dataset. The top left patch represents the highest response.
Figure 8.4: Best image patches for feature #101, from one run on the Caltech 101 dataset. The top left patch represents the highest response.
Figure 8.5: Best image patches for feature #501, from one run on the Caltech 101 dataset. The top left patch represents the highest response.
Figure 8.6: Best image patches for feature #1001, from one run on the Caltech 101 dataset. The top left patch represents the highest response.
8.3 Feature Sharing Across Categories

For any given feature, there will be one category for which it "votes" the most strongly. In fact it is possible to sort all the categories – in the case of the Caltech 101, from 1 to 102 – in order of how strongly the particular feature votes for them. In figure 8.7 we show, for a few chosen features from one run, how rapidly each feature's influence tails off from the 1st to 102nd category. This gives us a feel for how much features are being shared among categories.

Most features seem fairly strong for a significant number of categories (at least 20). Notably, the features ranked as most important overall by the feature selection process (section 5.4) tend to be features that vote very strongly for one or two categories over all the others. Features 1-6 are shown in the top half of figure 8.7; the general trend holds for the top 100 or so features.

8.4 Utility of Different Feature Sizes

Recall that S2 features come in various sizes: 4x4, 8x8, 12x12, and 16x16. It turns out that 4x4 features are significantly favoured by the feature selection process, i.e., they are the most informative features for this task [41]. This can be seen in figure 8.8, which shows the percentage of each feature size remaining after feature selection. (Note that we do not show absolute numbers because smaller features are also somewhat preferentially favoured in the original sampling process due to edge effects.)

Over 20% of the 4x4 features originally sampled survived the feature selection phase, as compared to just over 5% of the 16x16 features.

An interesting exception to this occurs in the first 100 or so features – the
Figure 8.7: Feature sharing among categories, for a few features from one run on the Caltech 101 dataset. The y axis for each subplot represents the weight with which the feature votes for each category (averaged across all binary SVMs involving the category; negative weights count as zero). Categories (the x axis) are sorted by this value to show the rate of dropoff. The leftmost category for each subplot is the category for which the feature votes most strongly. \( w \) represents each feature's overall weight (the area under the curve), relative to that of feature #1.
Figure 8.8: Percentage of each size of feature remaining after feature selection.

features ranked as most informative overall by the feature selection process. As shown in figure 8.9, larger features are much more common in the top 100.
Figure 8.9: Relative proportions of feature sizes by selection rank. 4x4 features dominate except among the very top-ranked features. This is best seen in the top graph.
9 Other Experiments (Graz Datasets)

This chapter describes our experiments on the more difficult images of the Graz-02 datasets.

9.1 The Graz-02 Datasets

We ran some additional tests on the Graz-02 datasets [28] (bikes, cars, and people). Each image for a given category contains one or more instances of that category only. In other words, “bikes” images contain only bikes, “cars” images contain only cars, etc. There is also a “background” category whose images do not contain instances of any of these categories.

As in other studies utilizing these datasets, we perform only single-category-vs-background tests. For the “bikes” category, the task is simply to tell whether or not a given image contains a bicycle. Unlike the UIUC tests, the particular location within the image is not important.

While the task may be simpler, the individual category datasets are harder than those of the Caltech 101 or UIUC datasets. They contain greater pose variability, and there are more instances of partial occlusion.
9.2 Training the Model

Our model (with localized C2 features; see section 5.3) needs to be trained on centred images. We make use of the annotations, provided with the Graz-02 data, to randomly select 50 square subimages containing a single object each. The images they came from are set aside, i.e., they are unavailable for testing. We then create a left-right flipped copy of each positive subimage, giving us a total of 100 positive subimages for training.

Next, 500 negative (background) subimages are taken from randomly-selected images in the "background" category. Each subimage is extracted from a randomly-chosen bounding box, equal in size to the average bounding box size of the positive training examples. See figure 9.1.

The feature dictionary contains 1,000 features (selected in 3 rounds from 8,000 features). All other parameters are again unchanged.

9.3 Testing the Model

All remaining images (from the positive category and the background category) form the test set. For each test image, we slide a square window through all positions and scales. Images are classified as positive or negative based on the peak detector response over all window locations.

9.4 Results

Our results for the single-category-vs-background test for each of the three categories are shown in table 9.1. As was the case in [28], we do not do as well on these more difficult images, and our scores are somewhat lower than those of [28].
Figure 9.1: Subimages used to train the Graz-02 "bikes" classifier.

<table>
<thead>
<tr>
<th>Dataset (vs. background)</th>
<th>Our score</th>
<th>Opelt et al. [28]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bikes</td>
<td>72</td>
<td>78</td>
</tr>
<tr>
<td>Cars</td>
<td>64</td>
<td>71</td>
</tr>
<tr>
<td>People</td>
<td>81</td>
<td>81</td>
</tr>
</tbody>
</table>

Table 9.1: Our results for the Graz-02 datasets.
Part of the discrepancy may be due to the fact we are actually solving the more difficult problem of localization, but then throwing the location away. We are also using fewer training instances than [28], and our method of selecting subimages for training may also be skewing things somewhat. It is possible that by taking many of the easily-separable single-object subimages for training, we are leaving ourselves with a slightly harder test set. Finally, it is possible that a wider range of C2 position/scale tolerance would be appropriate for this more varied dataset. These results should thus be viewed as preliminary.

In any case, this dataset shows there is still room for improvement on difficult single-class-vs.-background tasks.
10 Discussion and Future Work

10.1 Summary

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 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 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 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 [33]. 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.

10.2 Future Work

Most of the performance improvements for our model were due to the feature computation stage. Other recent multiclass studies [19, 42] have done well by improving the 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.

Our ultimate goal is to emulate the process of object classification by humans.
The initial, feedforward mode of classification is the obvious first step. A recent, updated model by Serre et al. [34] – 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 in the classic animal/non-animal task of Thorpe et al. [39]. This model has yet to be tested on a large multiclass problem like the Caltech 101. Even with a perfect model of human feedforward recognition, it is not clear what level of performance we could expect. More psychophysical study of the boundary between what humans can do in the initial feedforward stage and what requires recurrent processing (serial attention, integration of gist, context, and top-down expectation) is needed.

The feedforward model is probably still far from complete. Features in the various layers can probably be modeled more accurately, and the feature learning stage is still very crude. Within-layer interactions such as contour integration are absent. Higher-order features or view-tuned units might improve performance under wide variations in viewpoint.

Nevertheless, there will come a point at which it is appropriate to begin introducing back-projections into the model from higher levels in the ventral stream and also from other brain areas, visual and non-visual. This growth in model complexity will have to be managed with great care, as there are many recurrent connections in the brain, connecting many areas. This parallels the experience of many AI researchers, who have found that its very difficult to solve any one problem in isolation.
Bibliography


[38] Nicholas V. Swindale. How many maps are there in visual cortex? Cerebral Cortex, 10(7):633–643, July 2000.


