UNIFIED DETECTION AND RECOGNITION FOR READING TEXT IN SCENE IMAGES

A Dissertation Presented

by

JEROD J. WEINMAN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2008

Computer Science

UNIFIED DETECTION AND RECOGNITION FOR READING TEXT IN SCENE IMAGES

A Dissertation Presented

by

JEROD J. WEINMAN

Approved as to style and content by:

Allen R. Hanson, Co-chair

Erik G. Learned-Miller, Co-chair

Andrew McCallum, Member

Keith Rayner, Member

Andy G. Barto, Department Chair
Computer Science

To my mother, Gloria Reneé, the first teacher and scientist I knew, and my father James Lauren, the first funding agency I was fortunate enough to have.

ACKNOWLEDGEMENTS

As soon as one begins to reflect upon how many people, in some way or another, have had some influence on so many personal and professional matters, it becomes apparent how little of any achievement can be claimed as one's own but must surely be attributed to the influential efforts of others. So it is with this thesis. It is impossible to name everyone who has played a role in making this work a reality, yet surely there are many to whom I owe a specific debt of gratitude.

First of all, none of this would have been possible without my parents, James and Gloria Weinman. Their constant love and support have blessed me for a lifetime. Although my mother went to be with the Lord ten years ago this week, her influence has indelibly marked every fiber of my being and every page of this thesis. She modeled for me a passion for her work, compassion for her students, great generosity for all, and a magnificent faith in the Lord. Truly, she lived up to her name and brought great glory to God.

There have been over 100 teachers in my formal education. I am indebted to many fine instructors on the faculty of the Alliance Public Schools. In particular, I must thank my high school English teachers Edward and Valerie Foy. Their lessons were simple, yet profound, and their gift to me of the craft of writing has had more impact than any other course. In my four years at Rose-Hulman Institute of Technology, I learned more about a great many things than I ever expected. The culture gave me great appreciation for the sense of community, the teachers modeled habits of thought that continue to be fruitful, and my fellow students sparked a sense of creativity like no other I have experienced. I am indebted to Professor Claude W. Anderson for challenging me in course after course. Never before had so much work been so much fun. His actions demonstrated to me how deeply he cares for his students, and he was my first model of an integrated life of following Christ in academia. To Professor Gary J. Sherman I am grateful for lessons in abstraction and for teaching me to see the beauty in mathematical patterns and visualization. His many mantras about flarns and clarps and going somewhere, doing something, and coming back, as well as the advice to "get concrete," have been immensely valuable. Finally, I want to thank Professor Roger G. Lautzenheiser for teaching an elective course on the mathematics of image processing that got this whole thing started.

My years at UMass have been enriched by many, beginning with Professor James Allan and S. Chandu Ravela. Their organization of and participation in an NSF Research Experience for Undergraduates brought me to UMass and sealed my interest in graduate study. I am grateful to my longtime advisor Professor Allen R. Hanson for giving me the opportunity to work in a historied research lab. He always provided me "more than enough rope," but consistently demanded clarity and never failed

to provide thoughtful critiques—born from his decades of experience—of my often wandering ideas. Professor Erik G. Learned-Miller's arrival at UMass could not have been timed better. I am immensely grateful for his insistence on doing work I could take pride in. His scientific and career mentorship have been truly valuable, and I will surely miss our frequent intellectual sparring and sharpening of each other's ideas. Professor Edward M. Riseman also had a hand in shaping ideas early in my graduate career. To Professor David Jensen I am indebted for a great deal of practical career advice. His personal mentorship helped me navigate the waters of late-stage graduate work, and his insightful course on research methods offered me lessons I hope I will hold for an entire career.

Many others at UMass have contributed to the efforts of this thesis in ways large and small, direct and indirect. I wish to thank: Charles Sutton for countless theoretical and technical discussions on topics often more detailed than he had probably bargained for; Aron Culotta for oodles of technical references; Trevor Strohman for many exciting, deep, and unexpectedly long conversations about nearly every topic under the sun; Vidit Jain for being a reliable sounding board and mathematical assistant; Andy Fast and Anne-Marie Strohman for sharpening manuscript prose; Piyanuch (Pla) Silapachote and Marwan A. Mattar for collaboration on the VIDI project and acquiring data used in this thesis; Professor Joseph Horowitz, George Bissias, Richard Weiss, David Walker Duhon, and Chris Pal for opportunities to collaborate and share ideas on interesting projects; Sarah Osentoski for giggling, commiseration, and Nebraska football games; and, all the tea totallers, coffee czars, bagel barons, donut dukes, and graduate representatives that make the UMass Computer Science Department the very special community that it is.

This work also would not have been possible without the support of several funding agencies. Various portions have been supported by the National Science Foundation under the NSF grants IIS-0100851, IIS-0326249 and IIS-0546666, and the National Institutes of Health under NIH grant 1R21EY018398-01. Additional support was provided by Central Intelligence Agency and the National Security Agency.

Time outside the department has been at least as important as my time inside. The people of the First Baptist Church of Amherst have truly been a family to me. I have been blessed time and again by them. To the many generations of the "somethings" growth group I humbly owe my thanks. You have fed me spiritually, sustained me emotionally, and nourished me physically. From Valley View, to Sugarloaf, to Blingo, to the Fireside, I can hardly think of a single thing that has meant so much over six years as Tuesday nights. And to my wonderful bride Jennifer Nicole I am so thankful for constant encouragement and an inspirational devotion.

Finally, the greatest acknowledgement is for a debt redeemed by Jesus Christ. By His grace and mercy can I strive to do all things to the glory of God.

יהוה פקח עור — 146:8 תהילים

ἓν οἶδα ὅτι τυφλὸς ὢν ἄρτι βλέπω. — Κατὰ Ιωάννην 9:25

ABSTRACT

UNIFIED DETECTION AND RECOGNITION FOR READING TEXT IN SCENE IMAGES

MAY 2008

JEROD J. WEINMAN

B.Sc., ROSE-HULMAN INSTITUTE OF TECHNOLOGY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Allen R. Hanson and Professor Erik G. Learned-Miller

Although an automated reader for the blind first appeared nearly two-hundred years ago, computers can currently "read" document text about as well as a seven-year-old. Scene text recognition brings many new challenges. A central limitation of current approaches is a feed-forward, bottom-up, pipelined architecture that isolates the many tasks and information involved in reading. The result is a system that commits errors from which it cannot recover and has components that lack access to relevant information.

We propose a system for scene text reading that in its design, training, and operation is more integrated. First, we present a simple contextual model for text detection that is ignorant of any recognition. Through the use of special features and data context, this model performs well on the detection task, but limitations remain due to the lack of interpretation. We then introduce a recognition model that integrates several information sources, including font consistency and a lexicon, and compare it to approaches using pipelined architectures with similar information. Next we examine a more unified detection and recognition framework where features are selected based on the joint task of detection and recognition, rather than each task individually. This approach yields better results with fewer features. Finally, we demonstrate a model that incorporates segmentation and recognition at both the character and word levels. Text with difficult layouts and low resolution are more accurately recognized by this integrated approach. By more tightly coupling several aspects of detection and recognition, we hope to establish a new unified way of approaching the problem that

will lead to improved performance. We would like computers to become accomplished grammar-school level readers.

CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The first attempt to build machines that could read occurred nearly two-hundred years ago [102]. Since then, computers have barely reached the reading level accuracy of a second-grade child [99]. Many facilities are involved in human reading [97], and similarly, there has been a vast amount of research on computational methods for text recognition. The bulk of text recognition research over the last fifty years has focused on scanned documents and faxes, but interest in indexing text in video and scene images has grown rapidly in the last decade. Detecting and recognizing text in scenes involves some of the same issues as document processing, but there are also several new problems to face. Recent research in these areas has shown some promise, but there is still much work to be done.

The problem of developing a computational model for reading is broad, encompassing many facets of information. There are any number of typefaces, and they may be encountered at many legible sizes. Text is usually printed on a surface, and humans do not generally require that the surface be fronto-parallel to their eyes in order to read it. Additionally, lighting conditions may vary widely and designers use many colors for text and background. All of these factors conspire to make the robust reading problem very challenging.

Many processes are undoubtedly at work when humans read. Reading text in arbitrary images likely goes hand in hand with scene interpretation and object recognition. Humans might (subconsciously) ask: *Where do I expect to see text? Does text at a particular size and location contradict a sensible scene interpretation?* Recognizing these factors will allow us to take some necessary steps toward a unified view of scene interpretation, even if such propositions are not directly modeled.

One of the central limitations of many current approaches is that they operate primarily in a feed-forward, bottom-up, pipelined architecture that isolates the many tasks and information involved in reading. The result is a system that commits errors from which it cannot recover and has components that lack access to relevant information.

We propose a system that in its design, training, and operation is more integrated. First, we present a contextual model for text detection (Chapter 3). Although we improve detection results by taking a broader view of the input scene than the typical sliding window approach, the model is still illiterate. That is, it can find what it thinks *looks* like text, but is unable to interpret it. Next, we focus on the recognition task, introducing a model that integrates several information sources, including

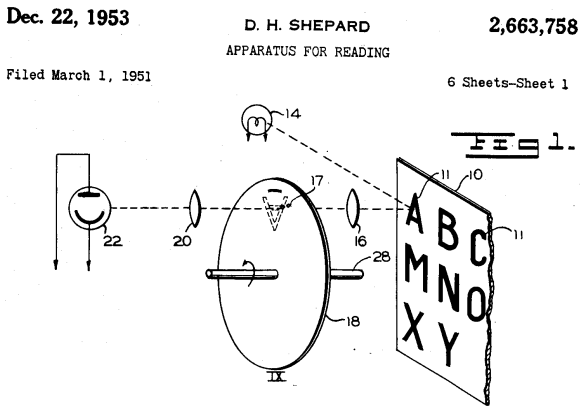


Figure 1.1. Figure from the patent for the first practical OCR system [105].

font consistency, linguistic properties, and a lexicon (Chapter 4). This unified reading model outperforms approaches using similar information in the more traditional pipelined architecture.

Detection and recognition are not processes that happen in isolation, but that instead require feedback and communication for optimal performance. Toward this end, we introduce a template-based local feature model that can be utilized for detection or recognition (Chapter 5). We achieve better results with fewer features by selecting features and training the model on the joint task of detection *and* recognition, rather than for each sub-task independently. Finally, we unify character and word segmentation with recognition, integrating these processes in a single model that can evaluate and compare interpretations with maximal awareness (Chapter 6).

Next we describe the problem of scene text reading and contrast it with the probem of document processing. In the remainder of this chapter we review some of the broad literature relevant to the tasks of document processing, text detection, and character recognition. Chapter 2 will then review the common, underlying framework for the probability models used in the rest of the thesis.

1.1 Scene Text

There is a long history of research and development of automatic readers. The first practical system was a patent filed in 1951 by David H. Shepard [105]. This device (see Figure 1.1) converted typewritten documents into punch cards in a magazine subscription department. Of course, hardware devices and software techniques have improved a great deal in the fifty intervening years.

Several differences exist between reading text in documents and in scene images. A few examples are shown in Figure 1.2. The first primary difference is in the problem of locating the text to be recognized. In a standard one or two column document, almost no detection must be done. Lines of text are easy to identify and simple heuristics are usually sufficient to prepare the input for recognition. In more complex documents such as newspapers and magazines, there is an added difficulty of distin-

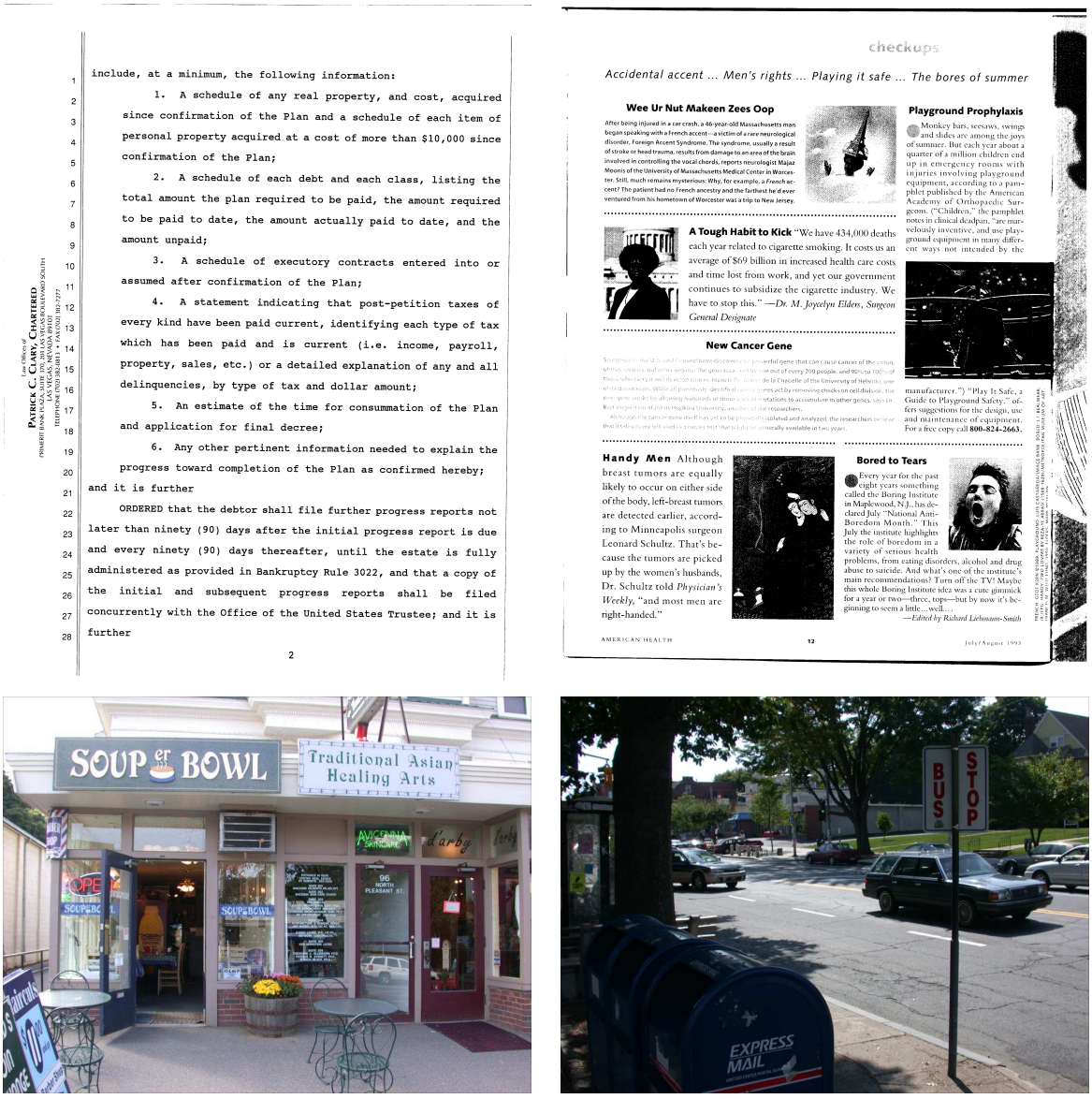


Figure 1.2. Images for document page reading (top) and scene text reading (bottom).

guishing between text and image regions. However, these are often separate and aided by several intentional cues such as high contrast in the text and strong rectangular boundaries in images. In addition, text almost always appears in canonical horizontal or vertical orientations, and many techniques have been developed to estimate the global rotation necessary to horizontally "level" text lines in the image plane. Recent developments have also been made in the analysis of pages using camera-based acquisition techniques. More complex world transforms must be considered in these cases, but often an explicit model of the page can be used to rectify the text and return it to a planar appearance. Uneven lighting can be problematic in these situations, but the general binary nature of text on a page still makes local processing feasible for providing good binarizations.

The problem of finding text in an arbitrary image of a scene can be radically more complex. First, the contents of the input image are generally more varied. From urban structures to more natural subjects like trees, the variety of potential image contents is vast, and it occupies all of the input image. Text regions in scene images need not be well-bounded the way they usually are in documents. Furthermore, text in scenes is often only a few words in one place. There are no large paragraphs or long lines to analyze. Although text is generally designed to be readable, there are often adverse effects of the imaging conditions that can make it difficult to identify. Distance from the camera can make text small and low resolution, without much detail. Specularities can mix text regions with a reflected image. Perspective distortion can produce text with a varying font size or orientation. Moreover, unlike document processing, there is no global page model whose transform parameters can be estimated. Because text may be printed on bricks, wood, or complex backgrounds, the simple binarization and text zoning algorithms of document processing will be insufficient. Small amounts of text can appear anywhere, at any size, with any world orientation, and on any surface. All of these problems tend to make text location in scene images generally more challenging than document text detection and pre-processing.

It is also important to contrast the nature of recognizing text in documents and scene images. As mentioned above, a region of text in a scene often involves only one to five words. This makes several techniques that might be used by document recognition systems less applicable. For instance, powerful language models that rely on complete sentences or longer phrases may not be useful. Also, techniques for identifying the font face of a text sample to aid recognition may not be robust with only a handful of characters. In addition to the small sample problem, the sheer variety of fonts that may be present in signs and scenes is often much greater than that found in most documents. Text recognition algorithms for scene images must handle a great number of fonts, many of which will be entirely novel for the system. Perspective distortions and complex backgrounds, as described above, will also require recognition algorithms to be robust to character "deformations" and non-binary input. In low resolution situations, it will not be reasonable to binarize text regions before recognition. This is in contrast to most document recognition systems, which often rely on binary images to at least perform word segmentation prior to recognition, if not a heuristic over-segmentation of binarized characters.

Input Output

Figure 1.3. Example scene text reading result.

In conclusion, several things make reading scene text fundamentally different from document reading. Most of these make the problem much harder. This thesis develops methods for overcoming many of these difficulties. While others have published some solutions for these problems, we examine the shortcomings of the assumptions made in prior work. Our central contribution is removing as many of these assumptions as possible, allowing several aspects of detection and recognition—even detection and recognition themselves—to be more unified. An example result using the methods of this thesis is shown in Figure 1.3. The rest of this chapter focuses on prior work in both document and scene text processing.

1.2 Text Detection

Unless characters are expected to appear at pre-defined page locations , as in forms processing, the text must somehow be located. Text detection in document processing is often treated as a very straightforward process. Typically, this involves a search for lines of text in a binarized image. Other approaches include processing and classifying connected components [4]. A cogent survey of the document image analysis field as represented by publications in the Transactions on Pattern Analysis and Machine Intelligence is given by Nagy [86]. Detecting text in scenes or low resolution video is more difficult, yet these problems have experienced increased prominence recently with contests sponsored by the International Conference on Document Analysis and Recognition in 2003 [72] and 2005 [73]. The two competitions received a total of nine entries. On the task of word block detection, the best system received an F-score of 0.62 with a recall of 0.67 and precision of 0.62.[1] In Chapter 3, we will present a model that on different, but similarly difficult, data receives an F-score of 0.72 (recall 0.67 and precision 0.79) on the broader (and admittedly easier) text block detection task.

[1]Recall is equivalent to detection rate, while precision is the fraction of true positives among predicted positives. F-score is the harmonic mean of recall and precision.

Table 1.1. Difficulties of text localization in various media.

Issue	Graphic Art	Video	Scene Image
Low Resolution	No	Yes	Sometimes
Font Variety	Yes	Yes	Yes
Low Contrast	No	No	Sometimes
Non-Standard Orientation	Yes	Rarely	Yes
Perspective Distortion	Sometimes	Rarely	Yes

In Chapter 5 we present a model and training algorithm that more closely couples the text detection with the eventual recognition.

There are a wide variety of techniques for detecting text in video, still images of scenes, or complex documents such as graphic art. These different input modalities differ slightly, but the difficulties are shared among them in various ways, as shown in Table 1.1. Most of the approaches can broadly be characterized as edge-/gradient-, color-, or texture-based segmentation and localization methods.

1.2.1 Edge- and Gradient-Based Text Detection

Relatively early work on text segmentation by Jain and Bhattacharjee [53] applied a bank of Gabor filters, followed by clustering to classify pixels. Similarly, Wu et al. [132] use a non-linear function of multiscale Gaussian derivatives to identify regions of high energy, which typically correspond to text. These features are also subsequently clustered. The same approach is also used by Thillou et al. [113]. Garcia and Apostolidis [36] use edge detectors and morphological operations to remove noise and fill in dense edgel areas. Gao et al. [34, 35] have a pipelined approach involving edge detection, adaptive search, color modeling, and layout analysis. For superimposed horizontal text in video, the technique of Wolf et al. [131] uses local integrations of horizontal derivatives followed by morphological processing. The results are tracked through frames, integrated for higher resolution, and binarized to improve performance with standard OCR software. Ezaki et al. [29] have a dual-mode approach. For small characters (under thirty pixels), the method combines the output from morphological operations, binarizes, and finds connected components in horizontally oriented regions. For larger characters, edges are calculated and a color-based method is also used to hypothesize character regions. Shen and Coughlan [104] compose edges into end-stops and pairs defining strokes, which may then be grouped into vertical and horizontal features.

Figure 1.4 illustrates one potential problem with edge-based methods. When text is small, blurry, and of low contrast, edge detectors may not find all of the vertical strokes perceived in higher resolution text, which these methods rely heavily on. If there is sufficient image detail to recognize text, then there ought to be enough to detect it. The method we propose in Chapter 5 uses the same image gradient-based features for both detection and recognition.

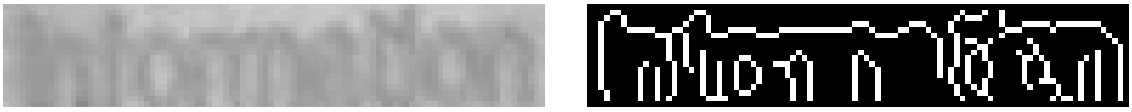


Figure 1.4. Small text in an image (x-height of 14 pixels) and its corresponding Canny edge map.

1.2.2 Color-Based Text Detection

Gao et al. [34, 35] use color as one step in their framework, assuming that character regions can be segmented with a Gaussian mixture model into foreground and background components. Alternately, Zhang and Chang [137] segment an input image using the mean-shift algorithm and then label each region as text or non-text with a Markov field. Attempts to binarize images without regard for character identity are prone to poor performance when images have low resolution and/or low contrast (see Figure 1.5 on page 12). Instead, our approach throughout this thesis will be to detect and recognize characters based directly on the image, rather than an intermediate, uninformed binarization.

1.2.3 Texture-Based Text Detection

Clark and Mirmehdi [24] use four simple texture features as input to a neural network for text detection. These include local grayscale variance, edge density, neighborhood histogram differences (a type of texture gradient), and a normalized edge direction symmetry measure. Li et al. [67] employ a neural network trained on the mean and second- and third-order central moments of wavelet coefficients to independently classify blocks of pixels. Chen and Yuille [21] use a combination of feature sets in a cascaded AdaBoost classifier to detect text of all sizes in images. By using features fixed to a particular window size, they are able to train a single classifier and run it at several window sizes to detect text at many scales. The features are a combination of intensity and gradient statistics, histograms motivated by the bimodal nature of text regions, and linked edges. Yamaguchi and Maruyama [135] employ a hierarchical classification of local regions for text detection. The first stage tests the region intensity histogram for bimodality. The second stage uses an SVM trained on sparse Haar wavelet coefficients.

Our models closely follow the texture-based approach, adding a variety of statistics of the gradient-based features to get regional texture cues. The main contribution is in how we use these features for the overall reading task.

1.2.4 Layout Analysis

Many of the text detection references above perform some form of layout analysis. Most use heuristics to find strictly horizontal or horizontal and vertical linear text strings [132, 36, 34, 29]. Thillou et al. [113, 31] correct for perspective distortion by following a coarse line analysis with vanishing point estimation and perspective

rectification, as do Gao et al. [34]. Still, both of these methods rely on hand-coded rules.

In contrast, Zhang and Chang [137] use triplet cliques on image segments to flexibly handle the linearity of text in a probabilistic model with learned parameters. Similarly, Shen and Coughlan [104] classify vertical and horizontal stroke features with learnable compatibilities in a probabilistic model.

Probabilistic systems for layout analysis are more flexible and require less manual parameter tuning than heuristic methods. As such, they should be able to directly handle text that is not strictly horizontal or vertical, if they are trained on such data. What these two approaches lack is feedback between the feature computation (i.e., segmentation [137], stroke detection [104]) and the classification of the features as text or non-text. Thus, top-down interpretation cannot influence the lower-level features. To address this issue, we propose joint feature selection for detection and recognition in Chapter 6 .

1.2.5 Summary

Many of the approaches to text detection and segmentation above may be classified as texture-based, since they use various features and statistics of intensities and gradients[53, 132, 24, 67, 135, 113]. Some make use of edge features or morphological operator outputs [36, 34, 35]. Others combine some of these features [131], e.g., at different stages in a pipelined classifier [21] or to handle different sizes of text [29]. By contrast, some others methods are region-based [137] or use higher-level features as input to a classifier [104].

With a few exceptions, most of these systems are divorced from the recognition process. As such, many do only a rough initial or windowed detection [53, 24, 21, 135], while others may track text regions across video frames [67, 131]. Some use heuristics for combining text regions and/or pruning non-text regions [132, 36, 34, 35, 113], while very few have learned or learnable layout analyses [137, 104].

Prior to recognition, many of the systems binarize text regions [132, 36, 131, 21, 137, 113], while one isolates characters (via binarization) but subsequently classifies features of the grayscale image [35].

By learning the spatial properties of text to improve detection and bridging the gap between detection and recognition during training, the methods we present in this thesis overcome many of the limitations present in earlier approaches to text detection.

Next we discuss some of the relevant approaches to recognizing text in document processing and more recent camera- and video-based systems.

1.3 Text Recognition

A great deal of prior knowledge can be brought to bear on the text recognition problem, from the essentials like the general appearance of characters, to more complex factors such as language and lexicon awareness. Many of these have been employed in one form or another, but typically they are not fully integrated. In this

section, we review some of the approaches to recognizing text in both documents and camera- or video-based systems.

1.3.1 Incorporating Language

Many have realized the advantage in crafting computer reading systems that utilize prior language knowledge to improve recognition. Work on processing misspellings goes back several decades [38]. The earliest use of uniting character confusion likelihoods with a lexicon constraint is by Bledsoe and Browning [11] in 1959. Device-specific character confusion likelihoods are combined with word unigram probabilities to find the most likely dictionary word given the OCR output. One major drawback is that the computational load is linear in the size of the lexicon. Language information may also be used by looking at smaller, more local pieces of information. Riseman and Hanson [100] use binary positional trigrams to flag and correct spelling or recognition errors. Later, Jones et al. [55] combined these ideas by using device-specific character confusions with bigram models to find the most likely string as an OCR post-processor.

Rather than model character confusions [11, 55], which are limited by both their device- and font-specific nature, it seems more natural to model prediction uncertainty. For example, when an `l` (ell) and a `1` (one) cannot be adequately distinguished in isolation, probabilistic methods can assign them both roughly the same high probability. Having equal weighting given the image, other sources of information can help resolve the ambiguity. This approach is presently more common.

At the word level, Zhang and Chang [138] use a Parzen window model with Gaussian kernels to model character conditional densities. Recognition is then conditioned on a prior probability over words, thus restricting output to lexicon entries. "Back-off" to character recognition is facilitated by a non-word prior probability. In this mode, only the character appearance is used; no bigrams or other linguistic properties are modeled.

At a higher level, Hull [50] introduced a Markov model incorporating part of speech (POS) information for word transitions. This uses a hard threshold on probabilities to restrict the set of hypotheses considered for each word. Given the word hypotheses and their corresponding parts of speech, the observation probability of each POS is generated for each "time" step; the transition probabilities are learned from a corpus. The most likely POS sequence is output, which is used to further restrict the set of word hypotheses but does not necessarily make a unique prediction. This higher level information can be quite useful for recognition under adverse condition.

A more integrated, mostly script-generic OCR system is presented by Bazzi et al. [5]. Preprocessing involves finding lines, and features are computed on small slices of a line that has been normalized to minimize the effects of font size (up to 14 pixel columns for a character). An HMM for each character traverses these slices. Operating in a closed-vocabulary mode, the tokens are lexicon words, and the transitions are based on word n-gram models. Since the state space is very large (the lexicon size), the Viterbi algorithm is approximated by a multi-pass search. In an open-vocabulary mode, the tokens are characters, and the transitions are based on character n-gram

models; this performs worse than closed-vocabulary. A hybrid method that uses character-based recognition with some higher level constraints from a word lexicon and a word unigram language model performs better than the character-based (open vocabulary) model but not as well as the word-based model, although it is free of the closed vocabulary assumption. In Chapters 4 and 6 we will present hybrid models that outperform both strictly open- and closed-vocabulary modes in empirical evaluations.

Kornai [59] discusses many of the issues with the common approaches to language modeling in recognition. Many issues are perennial (e.g., combinatorial explosion and pattern complexity), but several have seen progress (e.g., lack of joint optimization, closed world assumption, inappropriate setting of the alphabet, lack of semantic checking, etc.). We address the issues of joint optimization and the closed world assumption, among others, in this thesis.

1.3.2 Robust Recognition

As computers and methods have become more powerful, applications involving reading text from scenes have become a reality. Some of the earliest work in this area realized the importance of combining recognition with segmentation. In the work of Ohya et al. [87] scene images are segmented and candidate character regions are detected by observing gray-level differences between adjacent regions. Character pattern candidates are obtained by linking detected regions according to their nearby positions and similar gray levels. For recognition, the character pattern candidates are compared to a training data set. A relaxational approach to determine character patterns updates the comparisons by evaluating the interactions between categories of patterns, based on topological similarity. This important early work flexibly modeled alternative interpretations in a framework akin to probability. However, early stages of the algorithm still rely on binarization, which can be problematic (c.f., §1.2 and Fig. 1.5 on page 12).

Given a character segmentation and binarization, McQueen and Mann [80] place a grid on binary character images and count the lines (polarity changes) along horizontal, vertical, and diagonal directions. These features are compared to a set of training data, classifying them by the nearest-neighbor rule. The results are then post-processed in an ad hoc fashion with trigrams and a lexicon. After perspective rectification and binarization, Thillou et al. [113] isolate characters via connected components analysis and pass the results to a neural network for recognition. Their results are also post-processed by applying an n-gram model to the n-best character list.

Rather than binarize an image for recognition, Wang and Pavlidis [122] advocate extraction of features from the grayscale image to avoid information loss. The recognition system of Chen et al. [23] follows this lead. Although they use a Gaussian mixture model to binarize text regions, the purpose is only to isolate characters. Once found, a character is placed on a grid and Gabor filters are applied to a grayscale normalized image. A Fisher linear discriminant trained on 4,000 characters in six fonts is then used for classification.

Kusachi et al. [62] have a multi-resolution coarse-to-fine image scan for characters that is free of a separate detection stage. Feature vectors are calculated by sampling edge directions on a grid and then subjecting them to a PCA decomposition. A database of training examples is stored in the PCA format. Features are calculated at every location, and candidates are iteratively pruned if they do not sufficiently match the query's high-value eigenvectors. This approach, applied to Japanese kanji, is unique in completely unifying detection and recognition. However, it still relies on features derived from binarized images.

One of the most robust and widely-deployed systems for digit recognition is the convolutional network of LeCun [65]. Unlike other neural network approaches to classification, the convolutional network uses highly shared features that account for the strong spatial information involved in character recognition. It is related to the Fukushima Neocognitron [33], a model with hierarchical processing for invariant recognition based on successive stages of local template matching and spatial pooling. Rather than use the typical three-layer neural network architecture, a convolutional network is arranged with an initial set of image filter kernels, followed by a weighted averaging and downsampling operation. These outputs are then fed to another layer of convolution kernels for further averaging and downsampling. These final feature maps are finally fed to a fully connected classification layer. All of the kernel coefficients and weighted averaging parameters are learned from training data. Thus, the primary layer extracts basis features from an image (such as ink detectors and edge orientations), and the downsampling operation lessens their spatial specificity for robustness to distortion. The second layer extracts more complicated features of the initial filter outputs (perhaps intersections or loops) and is followed by more spatial pooling.

The convolutional network remains one of the best methods for digit recognition [108]. Recently it has been expanded to general character recognition for camera-based document processing by Jacobs et al. [52]. They combine LeCun's original dynamic programming method for simultaneous (one-dimensional) segmentation and recognition [64] with a constraint that word candidates (paths through the Viterbi trellis) come from a lexicon, made more efficient by a trie format [74].

A related framework by Belongie et al. is character recognition with shape context features [8]. A shape context is a local feature descriptor that bins edge detector outputs (possibly with orientation information) over a log-polar spatial area. They estimate an aligning transform of shape contexts between a prototype and a query, using the nearest neighbor rule for recognition. Although it is among the best on the MNIST data set for digit recognition, the approach has several key limitations. First, since it is a nearest neighbor approach, recognition is generally linear in the number of exemplars. Second, each exemplar must undergo a match process quadratic or cubic in the number of shape context descriptors it and the query contain. The matching complexity combined with the limitations of stored exemplar-type classifiers make this approach unattractive for general scene text recognition. However, the core idea of using features based on local image properties has been important for general discrimination techniques beyond matching.

Figure 1.5. Uninformed segmentation can make recognition difficult. TOP: Input image. LEFT: Probability of foreground under a Gaussian mixture model ($k = 3$, HSV color space). RIGHT: Binarization by threshold ($p\left(\mathsf{foreground}|k = 3, \mathsf{image}\right) \geq \frac{1}{2}$).

One other closely related work is that of Tu et al. [118]. Here, discriminative models are used as bottom-up proposal distributions in a large generative model. Discriminative text detectors are used, and then a generative model must be able to explain the pixel data as a contour describing a known character. They claim the advantage of the generative model is that it ensures consistent regions. However, spatially coupled discriminative models can overcome this [61, 124]. While their bottom-up detectors seem to do a reasonable job of detecting text to subsequently be explained, the generative models for characters are independent of one another. In other words, there is no coupling of information based on where a character should appear (i.e., next to other characters), nor how it should appear (similar to nearby characters). The models in this thesis will demonstrate the utility of such information for interpreting scene text.

We can broadly categorize the systems for robust character recognition along the following dimensions:

- Binarization [87, 80, 113] versus grayscale feature extraction [23, 62, 52, 118]
- Separate detection/segmentation [80, 113, 23] versus integrated detection/segmentation and recognition [87, 62, 52, 118]
- Language independent [87, 23, 62, 118] versus language post-processing [80, 113] versus integrated language models [52].

Although some OCR systems have a fully integrated language model [5, 14, 52], the systems thus far designed for reading scene text either use no language information [87, 22, 62] or, at best, post-processing [80, 113]. Moreover, as the most difficult scene text suffers from the difficulties listed in Table 1.1 on page 6, binarization and segmentation will become increasingly difficult without regard for interpretation (see Figure 1.5). The effectiveness of integrating language with simultaneous segmentation and recognition on grayscale—not binary—images was demonstrated in the camera-based OCR system of Jacobs et al. [52]. Research on noisy and degraded documents has recognized this fact for some time. Indeed, this may explain some of the success of text detection and recognition systems that use commercial OCR systems (e.g., [132, 131, 21]).

1.3.3 Adaptive Recognition

Several methods have been proposed to adapt the recognition process to a particular input. These include adaptive classifiers that change their character models to match the fonts present in a document, using character or word similarity to constrain classification to respect equivalences, and recognition-free approaches based on cryptogram solvers.

1.3.3.1 Document- and Font-Specific Recognition

Perhaps the simplest approach to adaptive recognition is to first identify the font in use and then use a classifier specific to that font. It is likely that many commercial OCR systems employ this strategy for clean scans of documents. Shi and Pavlidis [106] use font recognition and contextual processing to improve recognition. Font information—whether fixed width and/or italic—is extracted from global page properties and by recognizing short words. Contextual processing is done by post-processing spell correction, but also by measuring the distance of an illegal word to each word in the lexicon on the basis of a system-dependent confusions (much like Jones et al. [55]). Bapst and Ingold [3] demonstrate that typographic information can improve document image analysis, with fewer complex recognition heuristics. They examine font recognition with an *a priori* font set as the basis, the resulting monofont recognition problem, and advantages in word segmentation.

The other, more general approach is to *learn* font-specific character models during recognition. A system proposed in the mid 1960's by Nagy, Shelton, and Baird [84, 2] involves a heuristic self-corrective character recognition algorithm that adapts to the typeface of the document by retraining the classifier on the test data with the labels previously assigned by the classifier and iterating until the labels go unchanged. Kopec [58] updates character models from a document transcription. An aligned template estimation algorithm maximizes a character template likelihood, subject to some disjointness constraints. Glyph origins are located and labeled using some initial set of templates. Rather than use a transcription, Edwards and Forsyth [28] improve character models by bootstrapping a weak classifier from a small amount of training data. By identifying words—which are more discriminative than individual characters—with high confidence, their constituent characters may be extracted and added to character model training data.

A related approach is to parameterize font styles. For instance, Mathis and Breuel [78] examine the problem of test set "styles" (e.g., font properties) that are not present in the training set. They use training data to learn prior probabilities over font properties, which may then be marginalized in a Bayesian fashion during recognition. Veeramachaneni and Nagy [120] formulate the recognition problem as one where characters have a class-conditional parametric (i.e., normal Gaussian) distribution and use the typical MAP prediction rule. Latent font styles are introduced, and the EM algorithm is used on the unlabeled test data to acquire probabilities for the styles.

1.3.3.2 Character and Word Similarity

Closely related to font awareness and adaptivity is the application of character similarity and dissimilarity. Two tokens that look the same should often be given the same label, and two that look very different should conversely be given different labels.

The first application of this idea is due to Hong and Hull [46], who cluster word images according to image similarity. The results of OCR output from individual words are combined in a voting scheme to yield a single identity for an entire cluster. Manmatha et al. [76, 96] cluster handwritten historical document word images and create a document index by manually labeling a few clusters. This outperforms HMM recognition on the same data and trades full automation for a more accurate system with a reduced amount of manual input.

At the character level, Hobby and Ho [45] cluster character images by their similarity. The bitmaps are then averaged for improved readability of noisy fax images. Using the averages as input to OCR improves recognition rates, while voting on cluster labels does not give a statistically significant improvement. Breuel [15, 16] uses a neural network trained on pairs of characters that are labeled as being the same or different. The network is then used as a scoring function to cluster digits with similar appearances by simulated-annealing [15]. (No method is reported for determining the correspondence between cluster labels and actual classes.) Alternatively, the network may be used as a scoring function for a nearest neighbor classifier [16].

These methods capitalize on the idea of similarity; that characters and words of similar appearance should be given the same label. However, they suffer from the drawback that there is no feedback between the labeling and clustering process. Hobby and Ho [45] ameliorate this somewhat by purging outliers from a cluster and matching them to other clusters where possible. These processes all solve the clustering and recognition problems in separate stages, making it impossible to recover from errors in the clustering stage. In Chapter 4 we present a model that incorporates character (dis)similarity with other sources of information simultaneously, outperforming the pipelined approaches detailed above.

1.3.3.3 Cryptogram Approaches

At the extreme end of the adaptability spectrum is the hypothesis that no appearance model whatsoever is necessary, but that recognition can occur using only language. This is the idea behind cryptogram puzzles, where a short message is encrypted with a substitution cipher. These typically may be solved by frequency analysis and recognizing letter patterns in words. Examples include Nagy et al. [85], Ho and Nagy [44], Lee [66], and Huang [48].

1.3.4 Summary

There are many levels and sources of information that may be employed for text recognition. Language models may consist of lexicons [11, 106, 5, 138, 52, 28], n-grams [100, 55, 5, 14, 80, 113], and parts of speech [50], but often these act as post-processors

rather than integrated systems. In addition to language, identifying character boundaries is necessary for recognition. This involves handling the foreground/background problem either by binarization [87, 80, 113] or by operating directly on a grayscale image [23, 62, 52]. Inter-character boundaries must also be segmented, which may be a separate process [80, 113, 23] or integrated with recognition [87, 62, 52]. Both of of these problems become more difficult as the input degrades. Using local font and style information is also clearly helpful for improving recognition performance, but typically this is done as a pre-processing step.

We have highlighted some historically significant or the most immediately relevant work. More references to further camera-based systems may be found in a recent review by Liang et al. [69]. The model we present in Chapter 4 shows how language information, robust recognition features, and adaptive strategies can be united.

1.4 Joint Detection and Recognition

The path from sensed image to text cognition in humans is complex and still somewhat unknown. Nearly all approaches to computer-based reading have treated the detection and recognition components of the reading process as isolated stages [70, 133, 134, 22, 20, 21]. At best, these approaches use recognition to filter out suspect nonsense detections or find the best character segmentation after a binarization process.

Two works to date that we are aware of have a more integrated detection and recognition process. The first is that of Ohya, et al. [87], first described in Section 1.3.2. Here, candidate segmented regions are dynamically combined to be interpreted as characters, if possible. The "detection" occurs only if the joined regions form a recognizable character. In a unified model by Tu et al. [118], discriminative text detectors are used as (bottom-up) proposal distributions for a (top-down) generative recognition model. This generative model aims to explain the raw grayscale pixels as characters, among other classes. Some works bypass the detection stage all together by adding a non-character "background" class to the set of categories a classifier must distinguish among. This is the approach taken by Kusachi et al. [62], also reviewed in Section 1.3.2. By using a sliding-window technique, a character classifier then also performs detection. The problem with this method is that it greatly increases the number of hypotheses the initial classifier must consider. Kusachi et al. ameliorate this somewhat by using a coarse-to-fine matching strategy to quickly prune non-character regions. Unfortunately, their use of a non-parametric nearest neighbor classifier minimizes the benefit since every input must be compared to the database of sample characters at least once.

To summarize, these methods are all unified in their approach to processing at test time. Ohya, et al. [87] has no learning component, though it does use a flexible relaxation labeling strategy. Tu et al. [118] learn their generative and discriminative models independently for each class (background, characters, faces, etc.). Although the Bayesian model doesn't explicitly contain a detection step, one is implicitly used to improve the approximate inference strategy. Kusachi et al. [62] simply dispense

with the detection stage altogether. One important facet being overlooked by all of these methods is the opportunity to improve testing performance by unifying the training of detection and recognition stages.

Later in the thesis we will examine how to tie detection and recognition together during learning as well as the operation of the system. This will rely on sharing intermediate levels of the classifier—i.e., features. In the remainder of this section, we review some related feature selection strategies as they apply to computer vision tasks and some recent work related to sharing features for different tasks.

1.4.1 Feature Selection

Several general frameworks exist for selecting features. The two most basic are greedy *forward* and *backward* schemes. Forward schemes incrementally add features to a model based on some criterion of feature utility. Examples of this include work by Viola and Jones [121], who use single-feature linear classifiers as weak learners in a boosting framework, adding features with the lowest weighted error to the ensemble. A similar forward method by Berger et al. [9] involves adding only those candidates that most increase the likelihood of a probabilistic model. Backward schemes, by contrast, selectively prune features from a model. The ℓ_1 or Laplacian prior [128] for neural networks, maximum entropy models, logistic regression, etc. belongs to this category. In this scheme, features are effectively eliminated from a model during training by fixing their corresponding weights to zero. Many other variants for selecting a subset of features are possible; see Blum and Langley [12] for a more thorough review.

Feature types and selection strategies for visual tasks have varied widely. The Viola and Jones object detector [121] employs outputs of simple image difference features, which are similar to wavelets. There are many possible filters and only a few are discriminative, so a selection process is required primarily for computational efficiency. Other methods use image fragments or patches as feature descriptors. These patches may be taken directly from the image [119, 1] or an intermediate wavelet-based representation [103, 83].

1.4.2 Feature Sharing

One of the most prominent examples of shared features in the general machine learning literature is Caruana's work on multitask learning [19]. The central idea is to use a shared representation while learning tasks in parallel.

In particular, the high-dimensional patch-based image features mentioned above are often densely sampled and vector quantized to create a discrete codebook representation. Winn et al. [130] iteratively merge code words that do not contribute to discrimination overall, while Jurie et al. [56] create a clustering algorithm designed to cover the space with code words for better discrimination. Bernstein and Amit [10] cluster patches in a generative model for characters. Alternatively, LeCun et al. [65] learn (rather than select for) a discriminative intermediate feature representation. However, all of these methods are focused on one type of recognition, either catego-

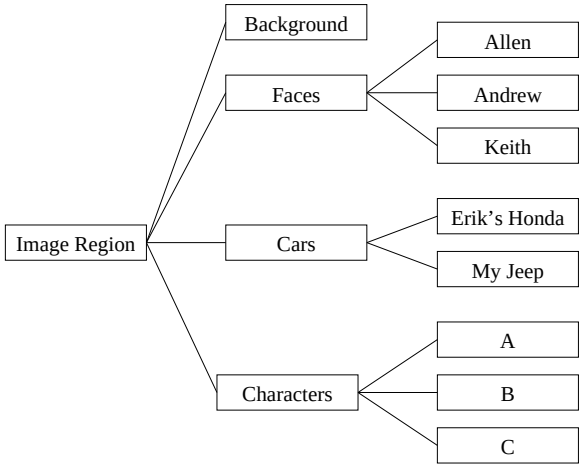


Figure 1.6. An example object class hierarchy for images. Categorization, or detection, is classifying instances with labels from the center column, while identification, or recognition, consist of giving instances labels from the right column.

rization (detection) or identification (recognition). Since all derived classifiers have the same codebook of patches, they use a shared representation.

Torralba et al. [114] have also applied this idea to vision tasks by showing that jointly selecting features for detecting several object categories reduces the requisite number of features and generalizes better than selecting features for detectors independently. Bar Hillel and Weinshall [43] demonstrated that learning category level models first and using that (fixed) representation for more specific recognition is better than learning recognition models directly.

Only recently has work appeared that extends the idea of learning and feature sharing across a task hierarchy. The earlier work of Torralba et al. [114] simply shared features for multiple object category detection tasks. However, they give one example in their latest work of performing shared feature selection for face detection and mood categorization (a more generic form of recognition)—hierarchically related tasks [116]. The technique is based on boosting, which learns a one-versus-all binary classifier for each class of interest. Their selection method requires an explicit search of the (sub)classes that will share a feature. Exponential in complexity, they approximate the search with a greedy best-first forward selection strategy.

1.4.3 Summary

Methods for text detection and recognition have either been learned and operated independently [70, 133, 134, 22, 20, 21] or else primarily approached from a recognition-only point of view (with detection the logical implication of a recognition) [87, 62, 118]. The latter method can be more computationally expensive because there are many more classes to consider. Feature selection strategies can reduce the computational burden of a classifier [119, 1, 121, 83], while learning shared representations for multiple recognition tasks [65, 114, 56, 130, 10, 43, 116] can improve gen-

eralization by sharing training data. Two works have examined learning hierarchical recognition tasks like detection (or categorization) and recognition (subclass identification). One fixes the representation (features) learned at the category level [43] for subclass recognition, while the other jointly selects features for both levels [116].

1.5 Conclusions

There are many computational aspects of the entire process of reading the text present in an image. All the features used for text detection are likely useful, but their performance is bounded by the extent that they overlap with non-text in feature space. False positives and missed detections are evidence of this, and indeed all bottom-up architectures for recognition of any sort are vulnerable to the problem. The top-down influence of recognition will be required both to eliminate the detection of texture that appears to be characters (imagine a white picket fence as a string of ones), as well as finding more instances of text that don't conform to typical cases (such as a lone letter or an nonlinear baseline).

The theme of this chapter is that by segregating the computational processes of reading, opportunities for cooperation and information sharing have been missed. This thesis demonstrates that these missed opportunities translate into performance losses.

CHAPTER 2

DISCRIMINATIVE MARKOV FIELDS

A digital image is a projection of high dimensional, high resolution reality into a lower dimensional and lower resolution space. The basic purpose of computer vision, manifested in various ways, is to recover some unknown aspects of that reality from the comparatively small amount of data in images. In a few cases, geometry and laws of the physical world guide this inference process in a deductive fashion. In most cases however, images contain insufficient information to uniquely determine truths about reality. Therefore, we are forced to reason from incomplete information about propositions concerning the real world. For this reason, computer vision and probability theory are inextricably linked.

The underlying computational mechanism for the contributions made in this thesis is the discriminative Markov field. This powerful tool models dependencies between unknowns, learns from training data, and gives interpretable results in the form of probabilities. Introduced by Lafferty et al. [63], these undirected, discriminatively trained graphical models are experiencing wide popularity in a number of fields for their great flexibility and effectiveness.

In this chapter, we review the necessary general background for the specific models we present throughout the thesis. First we introduce the basic notation and building blocks for discriminative Markov fields. Then we discuss some of the general issues with learning parameters from data. Finally, we briefly present some methods for using the models to make predictions. In short, we cover:

- What is the basic model? (§2.1)
- How can we learn it? (§2.2)
- How can we use it? (§2.3)

2.1 Basic Model Formulation

2.1.1 Probability for Prediction

As mentioned above, computer vision involves making inferences from images. Any time we must reason from incomplete information, the inferences are always conditioned on some amount of background knowledge, prior information, or assumptions. We shall denote these as I throughout and condition all formal probabilities upon them.

Throughout this thesis, we will want to infer things about a set of unknowns, denoted as $\mathbf{y}$, given some observation (usually an image), denoted as $\mathbf{x}$. In the simplest sense, this involves assigning a probability distribution to the set of possible values for $\mathbf{y}$, conditioned on the observation and our prior information I: $p(\mathbf{y} \mid \mathbf{x}, I)$. This distribution represents a state of knowledge about $\mathbf{y}$. Typically, assigning such a probability is either very complex or requires many (probably unwarranted) assumptions. Therefore, we introduce a parameter or model space $\boldsymbol{\Theta}$ that describes a range of alternative possibilities for relating $\mathbf{x}$ to $\mathbf{y}$. If we have further evidence in the form of training data $\mathcal{D}$, we may then employ the laws of probability to calculate the predictive distribution

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}, I) = \int_{\boldsymbol{\Theta}} p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I)\, p(\boldsymbol{\theta} \mid \mathcal{D}, I)\, \mathrm{d}\boldsymbol{\theta}. \tag{2.1}$$

Note we have assumed that (i) given a prediction model $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, the training data $\mathcal{D}$ do not reveal anything additional about $\mathbf{y}$, and (ii) given the training data $\mathcal{D}$, an additional image $\mathbf{x}$ does not give any information about the prediction model $\boldsymbol{\theta}$.

For now, we will focus on two probabilities in (2.1), the parameter-conditional predictive distribution $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I)$ and the parameter posterior $p(\boldsymbol{\theta} \mid \mathcal{D}, I)$. Acquiring and using the full predictive distribution are discussed in the next two sections.

2.1.2 Model Structure

One general way to represent the relationship between $\mathbf{x}$ and $\mathbf{y}$ is by capturing any local dependencies among the unknowns, which may also depend on the observation. Here, we motivate and give the general form for such a model.

Formally, we decompose $\mathbf{y}$ into a number of individual unknowns y_i, indexed by $i \in V$, that take values from some set $y_i \in Y_i$. Frequently the set of labels is the same for all y_i, but this need not be so. If we want to refer to some subset of the unknowns, $C \subset V$, we write $\mathbf{y}_C$. The space of all $\mathbf{y}_C$ is thus the Cartesian product of the individual label spaces,

$$Y_C \equiv \bigotimes_{i \in C} Y_i,\ C \subseteq V. \tag{2.2}$$

Let Ω be the domain for the observations $\mathbf{x}$.

If observing one or more of the y_i were to change our state of knowledge about some of the other unknowns, then we say these are logically dependent. Such dependencies are frequently found in image understanding applications. For instance, knowing one region contains text increases our belief that similar neighboring regions are also text. These influences can be modeled by local functions that represent the compatibility of labelings $\mathbf{y}_C$ given to small subsets C of the unknowns. Compatibility functions have the general form

$$U(\mathbf{y}_C, \mathbf{x}) : Y_C \times \Omega \to \mathbb{R}. \tag{2.3}$$

If we have a family of sets $\mathcal{C} = \{C \mid C \subseteq V\}$ with corresponding compatibility functions indexed by $C \in \mathcal{C}$, then we may write the conditional probability as an exponential sum over all the compatibilities

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I) \equiv \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{C \in \mathcal{C}} U_C(\mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta}_C)\right\}, \tag{2.4}$$

where $\boldsymbol{\theta}_C$ represents the parameters for a particular compatibility function indexed by C, which collectively represent the parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_C\}_C$ of the probability distribution. The normalization constant

$$Z(\mathbf{x}) \equiv \sum_{\mathbf{y} \in Y_V} \exp\left\{\sum_{C \in \mathcal{C}} U_C(\mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta}_C)\right\} \tag{2.5}$$

ensures that (2.4) is a proper probability distribution.

As an aside, we note that it is often natural to think of having more than one *type* of function associated with a particular set of unknowns C. For instance, one function on two neighboring unknown characters in a sequence could relate the compatibility of the bigram, while another could relate the compatibility of the same neighboring characters' cases (e.g., lower versus upper). Since the exponent in (2.4) is additive, such redundant functions U_C can be reduced to the simpler form presented here.

Models like (2.4) have a long history in statistical physics, where the compatibility functions U_C represent the energy of a particular configuration in a physical system. Hence, we will also refer to a compatibility function as an energy.

The probability could alternatively be written as a product of exponentiated compatibilities. Thus, the joint probability over $\mathbf{y}$ factors into a product of local functions, typically called "factors." Such exponential distributions are related to Markov fields by the Hammersley-Clifford Theorem [40]. Each unknown y_i and the entire observation $\mathbf{x}$ may be viewed as a node in a bipartite graph. The compatibility functions (or factors) are also introduced as nodes and connected by edges to their arguments.

A simple example of a so-called factor graph is shown in Figure 2.1. Here, the $\mathbf{y}$ represent unknown labels for characters. The joint probability is controlled by two types of factors, which are illustrated by square nodes in the graph. One is a function of a single character and the image, which influences the probability based on appearance. The other is a function of two neighboring characters and models local properties of the labels alone, like bigrams. Together, these factors contribute to the overall probability. Typically, the same function is replicated several times in a probability or factor graph, but with different arguments. For parsimony, we simply use the entire observation $\mathbf{x}$ as an argument to the compatibility functions, but generally when a function is replicated, the parameters $\boldsymbol{\theta}_C$ are the same for all C, and an appropriate portion of $\mathbf{x}$ is used within the function. In this example, the same discriminant is used for each character, but its input is drawn from different portions of the image. The bigram function does not use the image at all and is thus truly replicated as the same function with different arguments.

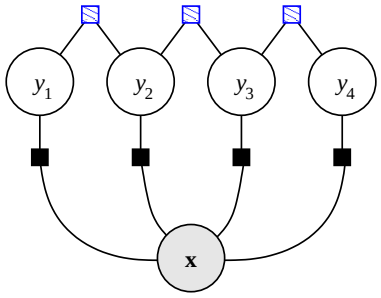


Figure 2.1. A simple factor graph that could be used for character recognition based on the appearance of characters and bigrams. Solid (black) factors represent the compatibility between a character label y_i and an associated portion of the image $\mathbf{x}$. Hatched (blue) factors can model the bigram statistics between neighboring y_i.

These types of models have a long history in computer vision. For instance, see books by Li [68] and Winkler [129] for detailed treatments of more traditional models. Traditionally, exponential/Markov field probability models were generative, i.e., they represented $p(\mathbf{y}, \mathbf{x} \mid \boldsymbol{\theta}, I)$ rather than the discriminative/conditional probability $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I)$. The discriminative form has become popular recently in many applications, such as region labeling [61, 41, 124], object detection [115], and object recognition [95]. We will employ them throughout this thesis for all aspects of the reading task, from detection to recognition. For an excellent tutorial on further applications of factor graphs, see Kschischang et al. [60].

2.2 Model Training

In the last section we discussed the parameter-conditional model $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I)$ that is part of the integrand used to calculate the predictive distribution $p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}, I)$. In this section, we formulate how the predictive distribution is calculated.

Unfortunately, the integral (2.1) is generally intractable, and thus requires approximation. One method might be to use Markov Chain Monte Carlo to sample from the parameter posterior $p(\boldsymbol{\theta} \mid \mathcal{D}, I)$ and approximate the predictive distribution with model averaging. Other alternatives include expectation propagation (EP) [81, 94], which takes advantage of the parameter posterior being a product of simple terms (which we will describe in more detail later in this section), and variational Bayesian (VB) methods [51, 6], which force a factorization of the parameter posterior. Both of these methods attempt to optimize the Kullback-Liebler (KL) divergence between the actual and approximate posteriors. This divergence not being symmetric, the difference between EP and VB is in the ordering of the arguments.

It is not yet clear whether such improved approximations are necessary or worthwhile for this particular Bayesian integral. Therefore, throughout this work we resort to the much simpler point approximation. The standard approach is to find the most likely model

$$\widehat{\boldsymbol{\theta}} \equiv \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} p\left(\boldsymbol{\theta} \mid \mathcal{D}, I\right) \tag{2.6}$$

and use the point approximation for the parameter posterior

$$p\left(\boldsymbol{\theta} \mid \mathcal{D}, I\right) \approx \delta\left(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\right) \tag{2.7}$$

so that the integral (2.1) becomes

$$p\left(\mathbf{y} \mid \mathbf{x}, \mathcal{D}, I\right) \approx p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I\right). \tag{2.8}$$

The question then becomes how to find the best model $\widehat{\boldsymbol{\theta}}$ given the training data.

First, we must explicitly write the form of the parameter posterior. With a set of training labels $\mathcal{D}_{\mathbf{y}} \equiv \left\{\mathbf{y}^{(k)}\right\}_k$ that are conditionally independent given a model $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and the corresponding training images $\mathcal{D}_{\mathbf{x}} \equiv \left\{\mathbf{x}^{(k)}\right\}_k$, we may use Bayes' rule to write the parameter posterior

$$p\left(\boldsymbol{\theta} \mid \mathcal{D}, I\right) = \frac{p\left(\mathcal{D}_{\mathbf{y}} \mid \mathcal{D}_{\mathbf{x}}, \boldsymbol{\theta}, I\right)}{p\left(\mathcal{D}_{\mathbf{y}} \mid I\right)} p\left(\mathcal{D}_{\mathbf{x}}, \boldsymbol{\theta} \mid I\right), \tag{2.9}$$

where $\mathcal{D} = (\mathcal{D}_{\mathbf{y}}, \mathcal{D}_{\mathbf{x}})$. The models of $\boldsymbol{\Theta}$ parameterize conditional distributions of $\mathbf{y}$ given $\mathbf{x}$. Thus, we have formulated the problem such that a set of images and the discriminative parameters are independent on background information I, so that $p\left(\mathcal{D}_{\mathbf{x}}, \boldsymbol{\theta} \mid I\right) = p\left(\mathcal{D}_{\mathbf{x}} \mid I\right) p\left(\boldsymbol{\theta} \mid I\right)$. Since we must eventually optimize the parameter posterior for $\boldsymbol{\theta}$, we need only be concerned with the terms that directly depend on $\boldsymbol{\theta}$ and can safely ignore both $p\left(\mathcal{D}_{\mathbf{x}} \mid I\right)$ and $p\left(\mathcal{D}_{\mathbf{y}} \mid I\right)$. Using the conditional independence of the training instances to factor the likelihood, we thus write

$$\begin{aligned} p\left(\boldsymbol{\theta} \mid \mathcal{D}, I\right) &\propto p\left(\mathcal{D}_{\mathbf{y}} \mid \mathcal{D}_{\mathbf{x}}, \boldsymbol{\theta}, I\right) p\left(\boldsymbol{\theta} \mid I\right) && (2.10) \\ &= \prod_k p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \boldsymbol{\theta}, I\right) p\left(\boldsymbol{\theta} \mid I\right). && (2.11) \end{aligned}$$

At this point, we note that the likelihood terms $p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \boldsymbol{\theta}, I\right)$ have the same form as the parameter-conditional predictive distribution (2.4),

The optimization (2.6) dictates we find the best model. Toward this end, we create an objective function from the logarithm (an optimum preserving strictly monotonic function) of the parameter posterior form in (2.11),

$$\begin{aligned} \mathcal{O}\left(\boldsymbol{\theta}; \mathcal{D}\right) &\equiv \mathcal{P}\left(\boldsymbol{\theta}\right) + \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{D}\right) && (2.12) \\ \mathcal{P}\left(\boldsymbol{\theta}\right) &\equiv \log p\left(\boldsymbol{\theta} \mid I\right) && (2.13) \\ \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{D}\right) &\equiv \sum_k \log p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \boldsymbol{\theta}, I\right) && (2.14) \end{aligned}$$

The likelihood (2.14) and the model prior (2.13) terms are discussed in more detail next.

2.2.1 Model Likelihood

Substituting the parametric form (2.4) for the model likelihood in (2.14) we have

$$\mathcal{L}\left(\boldsymbol{\theta};\mathcal{D}\right) \equiv \sum_{k}\left(\sum_{C\in\mathcal{C}^{(k)}} U_C\left(\mathbf{y}_C^{(k)},\mathbf{x}^{(k)};\boldsymbol{\theta}_C\right) - \log Z\left(\mathbf{x}^{(k)}\right)\right) \tag{2.15}$$

The set of functions $\{U_C\}_{C\in\mathcal{C}^{(k)}}$ depends on the particular unknowns $\mathbf{y}^{(k)}$, and thus $\mathcal{C}$ is indexed by the particular example k.

For certain forms of compatibility functions, it can be shown that the objective function $\mathcal{L}\left(\boldsymbol{\theta};\mathcal{D}\right)$ is convex, which means that global optima can be found by gradient ascent or other convex optimization techniques [13]. In particular, if the compatibility functions are linear in the parameters, then the log likelihood (2.14) is convex. Throughout this thesis, we use linear compatibility functions, which have the general form

$$U_C\left(\mathbf{y}_C,\mathbf{x}_C;\boldsymbol{\theta}_C\right) = \boldsymbol{\theta}_C\left(\mathbf{y}_C\right)\cdot F_C\left(\mathbf{x}\right), \tag{2.16}$$

where $F_C : \Omega \to \mathbb{R}^{d(C)}$ is a vector of features of the observation, the dimensionality of which depends on the particular set C. The parameter vector $\boldsymbol{\theta}_C \in \mathbb{R}^{|Y_C|\times d(C)}$ is conveniently thought of as a function $\boldsymbol{\theta}_C : Y_C \to \mathbb{R}^{d(C)}$ that takes an assignment $\mathbf{y}_C$ and returns an associated set of weights for the features F_C.

Taking the gradient of the objective (2.15) with respect to the parameters yields

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{L}\left(\boldsymbol{\theta};\mathcal{D}\right) &= \sum_{k}\sum_{C\in\mathcal{C}^{(k)}}\Big(\nabla_{\boldsymbol{\theta}_C}U_C\left(\mathbf{y}_C^{(k)},\mathbf{x}^{(k)};\boldsymbol{\theta}_C\right) - \\
&\qquad E_C\left[\nabla_{\boldsymbol{\theta}_C}U_C\left(\mathbf{y}_C,\mathbf{x}^{(k)};\boldsymbol{\theta}_C\right) \mid \mathbf{x}^{(k)};\boldsymbol{\theta}_C\right]\Big) \qquad (2.17)\\
&= \sum_{k}\sum_{C\in\mathcal{C}^{(k)}}\left(F_C\left(\mathbf{x}\right) - E_C\left[F_C\left(\mathbf{x}\right) \mid \mathbf{x}^{(k)};\boldsymbol{\theta}_C\right]\right). \qquad (2.18)
\end{aligned}$$

where E_C indicates an expectation with respect to the marginal probability distribution $p\left(\mathbf{y}_C \mid \mathbf{x},\boldsymbol{\theta},I\right)$. Equation (2.17) is the gradient for general compatibility functions, while (2.18) is for linear compatibilities (2.16).

To calculate the log likelihood and its gradient, and thus find the optimal model $\boldsymbol{\theta}$, we will need to be able to calculate $\log Z\left(\mathbf{x}\right)$, the so-called log partition function, and the marginal probabilities of each $\mathbf{y}_C$. In general, these both involve combinatorial sums, so approximations must be made. Most of these are described in Section 2.3.2, but we describe two here that are more closely related to the log-likelihood and the objective function.

2.2.1.1 Parameter Decoupling

One simple approximation that may be made is to decouple the parameters in $\boldsymbol{\theta}$ during training. For instance, in the example described at the end of §2.1.2 and shown in Figure 2.1, there are two types of compatibility functions, one for recognizing characters based on their appearance and another for weighting bigrams. If the parameter vector is decomposed into the parameters for the recognition and bigram

functions, i.e., $\boldsymbol{\theta} = \left[\begin{array}{cc} \boldsymbol{\theta}^A & \boldsymbol{\theta}^B \end{array} \right]$, we might then decouple the parameters by assuming they are independent, which means

$$p\left(\boldsymbol{\theta} \mid \mathcal{D}, I\right) = p\left(\boldsymbol{\theta}^A \mid \mathcal{D}, I\right) p\left(\boldsymbol{\theta}^B \mid \mathcal{D}, I\right). \tag{2.19}$$

This gives two new parameter posteriors in the form of (2.9), which may then be independently optimized as described in this Section (§2.2).

Decoupling the parameters in this fashion assumes certain "views" of the data are independent, i.e., the language "view" and the appearance "view." Only some subset of the factors end up being present in the probability models used in the likelihood, which can simplify training. Moreover, this makes it possible to use partial training data. For instance, when $p\left(\boldsymbol{\theta}^A \mid \mathcal{D}, I\right)$ and $p\left(\boldsymbol{\theta}^B \mid \mathcal{D}, I\right)$ are optimized separately, we might use two different sets of training data. For the appearance model, $\boldsymbol{\theta}^A$, we only need the images of individual characters and their associated labels, rather than a full sequence of characters as they might appear in context. Similarly, for the bigram model $\boldsymbol{\theta}^B$, we do not need any image data, only a training character sequence.

This strategy ameliorates the need to acquire large portions of fully-labeled training data. Indeed, this has long been an advantage of directed generative graphical models, which make similar independence assumptions and naturally factor the training process. One major difference is that in directed generative models, local normalizations serve to temper the disparity in magnitudes between different types of compatibilities. When $\widehat{\boldsymbol{\theta}^A}$ and $\widehat{\boldsymbol{\theta}^B}$ are found independently, there is no guarantee that the magnitude of the factors using them will be scaled appropriately. In other words, $\widehat{\boldsymbol{\theta}^B}$ might be fine for a probability containing only compatibilities U^B to model bigrams, but a compatibility U^A using $\widehat{\boldsymbol{\theta}^A}$ might have much larger values and thus (inappropriately) dominate a probability containing both U^A and U^B. In practice, we have not found this to be a problem, but a small amount of data may be used to learn linear weights on the resulting compatibilities U^A and U^B with $\widehat{\boldsymbol{\theta}^A}$ and $\widehat{\boldsymbol{\theta}^B}$ fixed.

2.2.1.2 Piecewise Training

Rather than decoupling parameter types, a second simple approximation involves what amounts to an independence assumption among unknowns for different factors. Typical probability models involve several overlapping factors explaining the same unknowns. This is the main reason the global normalizer $Z\left(\mathbf{x}\right)$ is necessary and intractable.

Rather than evaluate the likelihood of the entire probability model, we may decompose it into tractable pieces and evaluate them independently, collecting the results as an approximate likelihood. This so-called piecewise training approximation, due to Sutton and McCallum [111], can be justified as minimizing an upper bound on $\log Z\left(\mathbf{x}\right)$, which is a necessary part of the likelihood objective function $\mathcal{L}\left(\boldsymbol{\theta}; \mathcal{D}\right)$.

Formally, piecewise training involves approximating the likelihood with product of "independent" terms,

$$p\left(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I\right) \approx \prod_{C \in \mathcal{C}} \frac{1}{Z_C\left(\mathbf{x}\right)} \exp\left\{U_C\left(\mathbf{y}_C, \mathbf{x}, \boldsymbol{\theta}_C\right)\right\} \tag{2.20}$$

where the normalization constant

$$Z_C\left(\mathbf{x}\right) = \sum_{\mathbf{y}_C \in Y_C} \exp\left\{U_C\left(\mathbf{y}_C, \mathbf{x}, \boldsymbol{\theta}_C\right)\right\} \tag{2.21}$$

is now a sum over the typically much smaller space Y_C, rather than Y_V. Using the approximation (2.20) changes the likelihood objective to

$$\mathcal{L}^{\mathrm{PW}}\left(\boldsymbol{\theta}; \mathcal{D}\right) = \sum_k \sum_{C \in \mathcal{C}^{(k)}} \left(U_C\left(\mathbf{y}_C^{(k)}, \mathbf{x}^{(k)}; \boldsymbol{\theta}_C\right) - \log Z_C\left(\mathbf{x}^{(k)}\right)\right). \tag{2.22}$$

The log partition functions $\log Z_C\left(\mathbf{x}\right)$ are much easier to compute, and the gradient of $\mathcal{L}^{\mathrm{PW}}\left(\boldsymbol{\theta}; \mathcal{D}\right)$ has the same form as $\mathcal{L}\left(\boldsymbol{\theta}; \mathcal{D}\right)$, except the marginals used for the expectation are now computed using the individual probability "factors" of (2.20).

Both piecewise training and parameter decoupling will be important for training the models used throughout the thesis. Next, we turn to the issue of the prior probability for models $\boldsymbol{\theta}$ in the objective function.

2.2.2 Model Priors

Prior probabilities for model parameters are important because they prevent us from drawing too many conclusions from the training data. In other words, they capture our prior state of knowledge, which the training data must then sway us from. There are numerous justifications for various priors, especially as they relate to the exponential models we use (for instance, see previous work [123] and references therein). However, for this thesis it will suffice that priors are important, and thus are used, but the particular forms and justifications are not germane to the topics discussed. Therefore, we only present equations for the relevant priors employed with minimal discussion.

The simplest prior for parameters is the uniform prior,

$$p\left(\boldsymbol{\theta} \mid I\right) \propto 1. \tag{2.23}$$

Most of our study will involve Gaussian and Laplacian priors for the parameters. The Gaussian prior has the form

$$p\left(\boldsymbol{\theta} \mid \sigma, I\right) \propto \exp\left(-\frac{1}{2\sigma^2} \left\|\boldsymbol{\theta}\right\|_2\right), \tag{2.24}$$

where $\left\|\boldsymbol{\theta}\right\|_2$ is the ℓ_2 norm of the vector. The Laplacian prior has the form

$$p\left(\boldsymbol{\theta} \mid \alpha, I\right) \propto \exp\left(-\alpha \left\|\boldsymbol{\theta}\right\|_1\right), \tag{2.25}$$

where $\left\|\boldsymbol{\theta}\right\|_1$ is the ℓ_1 norm of the vector. The objective functions for (2.24) and (2.25) respectively are thus

$$\mathcal{P}^G\left(\boldsymbol{\theta}; \sigma\right) \;=\; -\frac{1}{2\sigma^2} \left\|\boldsymbol{\theta}\right\|_2 \tag{2.26}$$

$$\mathcal{P}^L(\boldsymbol{\theta};\alpha) = -\alpha\|\boldsymbol{\theta}\|_1 \tag{2.27}$$

with gradients

$$\nabla_{\boldsymbol{\theta}}\mathcal{P}^G(\boldsymbol{\theta};\sigma) = -\frac{1}{\sigma^2}\boldsymbol{\theta} \tag{2.28}$$

$$\nabla_{\boldsymbol{\theta}}\mathcal{P}^L(\boldsymbol{\theta};\sigma) = -\alpha\text{sgn}(\boldsymbol{\theta}), \tag{2.29}$$

where $\text{sgn}(\boldsymbol{\theta})$ returns the signum of every component in $\boldsymbol{\theta}$.

Functionally, both priors tend to promote small weights, but the Laplacian prior has heavier tails and prefers weights that are either zero or large in magnitude. While there are so-called "hyper-priors" to be fully Bayesian about the hyper-parameters parameters σ and/or α, we will assign these reasonable values or use cross-validation with data to select them.

In addition, the Laplacian prior has the advantage of implicitly performing feature selection [17]. Formally, any parameter θ that is zero, must have a likelihood gradient magnitude (2.14) that exceeds α, or else the total gradient of the objective (2.12) is taken to be zero for the parameter. This has the result of keeping the weight at zero, implicitly pruning the feature from the model.

2.3 Model Inference

Even if we use the approximations given in Section 2.2 to select the model $\widehat{\boldsymbol{\theta}}$, we still need to give a strategy for making predictions from the resulting probability distribution $p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I\right)$. The next section lists some of these, and we follow with a brief description of the algorithm that may be used to approximate them.

2.3.1 Prediction Strategies

Given the observation $\mathbf{x}$ and a model $\widehat{\boldsymbol{\theta}}$, we only have a posterior distribution on labels. When we need to pick a hard and fast label for each region of the image, the question becomes what to do with that distribution; what estimator do we use? A simple, oft-used answer is to find the most likely labeling. That is, use maximum *a posteriori* (MAP) estimation:

$$\widehat{\mathbf{y}} \equiv \arg\max_{\mathbf{y}\in Y_V} p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I) \tag{2.30}$$

Unfortunately, we have the issue of an intractable search space Y_V. Simulated annealing [57] could be used to search for a maximum. In the following section, we review another algorithm, called max-product, for finding such maxima.

The MAP estimator has an important caveat: poor predictions can result when the maximum of the posterior is not representative of most of the other likely labelings [39, 32]. In other words, the highest peak is not at the center of most of the posterior volume.

An alternative method for prediction is called maximum posterior marginal (MPM) estimation:

$$\hat{y}_i = \arg\max_{y_i \in Y_i} p\left(y_i \mid \mathbf{x}, \boldsymbol{\theta}, I\right), \forall i \in V. \tag{2.31}$$

It corresponds to choosing the label for each unknown that maximizes the probability with all other labelings marginalized. This can often be a more effective way of considering the probabilities of all the labelings, rather than simply the maximum (joint) labeling, as in MAP. Marginalization, however, suffers from the same computational complexity problems. Sum-product loopy belief propagation, described next, can be used to approximate these marginals.

In practice, the MAP estimate tends to be conservative, trying to give the most correct labels, while MPM tends to give higher detection rates. These methods are used variously in our experiments.

2.3.2 Belief Propagation

Because they all involve combinatorial sums or search spaces, we require approximations to compute the log partition function for the likelihood objective (2.15), the marginal probabilities for its gradient (2.18), or to make predictions with MAP or MPM, Fortunately, there is one family of algorithms that handles all of these. Namely, the loopy sum-product or belief propagation algorithm. In short, it exploits the nature in which the joint probability factorizes into a product of local functions.

Recall the bipartite "factor graph" between unknowns $\mathbf{y}$, represented by nodes in the graph, and the compatibility functions (or factors, when they are exponentiated) over the unknowns, as shown in the example of Figure 2.1 on page 22. The algorithm operates by iteratively passing messages between the nodes representing the unknowns and the factors. When these messages converge to a stable fixed point, the results are equivalent to minimizing the so-called Bethe free energy, a variational method for approximate inference from statistical physics [136]. When the factor graph is a tree, the sum-product algorithm [60] efficiently performs exact inference. In most cases, our graphs are not trees, and thus the results are only approximate.

Let $\mathcal{N}\left(i\right) \equiv \left\{C \in \mathcal{C} \mid i \in C\right\}$ be the family of indices for the factors that neighbor the ith unknown. This corresponds to the set of factors having y_i as an argument. The set of edges in a factor graph are thus

$$E\left(\mathcal{C}\right) \equiv \left\{(i, C) \mid i \in V \wedge C \in \mathcal{N}\left(i\right)\right\}. \tag{2.32}$$

For all edges in the factor graph $(i, C) \in E\left(\mathcal{C}\right)$, the node-to-factor messages have the general form

$$m_{i \to C}\left(y_i\right) \quad \propto \prod_{C' \in \mathcal{N}(i) \setminus C} m_{C' \to i}\left(y_i\right) \tag{2.33}$$

so that the message from a node to a factor is the product of all the messages to that node from all other neighboring factors. The resulting functional message is normalized (i.e., sums to 1 over y_i) for numerical stability. Note that for parsimony,

we drop the dependence of the messages m and beliefs b on the observation $\mathbf{x}$ and parameters $\boldsymbol{\theta}$.

The factor-to-node messages combine the local information expressed in the factor (exponentiated compatibility) and the current messages from its other arguments,

$$m_{C \to i}(y_i) = \sum_{\mathbf{y}_{C \setminus \{i\}} \in Y_{C \setminus \{i\}}} \exp U_C(\mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta}_C) \prod_{j \in C \setminus \{i\}} m_{j \to C}(y_j). \tag{2.34}$$

These messages are iteratively passed throughout the graph until they converge. Convergence is not guaranteed, but the algorithm empirically tends to give reasonable results in many applications. The name sum-product derives from the form of (2.34), which is a sum over products.

At any step in the belief propagation algorithm, the current belief (approximate marginal probability) at a node is represented by the normalized product of messages to that node from its neighboring factors

$$b_i(y_i) \propto \prod_{C \in \mathcal{N}(i)} m_{C \to i}(y_i). \tag{2.35}$$

Usually this is only used when the algorithm has converged, but in some experiments we will use it in the middle of belief propagation. These beliefs may then be used to approximate the MPM prediction (2.31).

The approximate marginals of sets of nodes corresponding to $C \in \mathcal{C}$ may be represented by the normalized product

$$b_C(\mathbf{y}_C) \propto \exp U_C(\mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta}_C) \prod_{i \in C} m_{i \to C}(y_i). \tag{2.36}$$

These may be used to approximate the marginals $p(\mathbf{y}_C \mid \mathbf{x}, \boldsymbol{\theta}, I)$ required for calculating the expectations in the likelihood gradient (2.18).

The likelihood $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I)$ may be directly approximated by the product ratio

$$b(\mathbf{y}) = \frac{\prod_{C \in \mathcal{C}} b_C(\mathbf{y}_C)}{\prod_{i \in V} b_i(y_i)^{|\mathcal{N}(i)|-1}}. \tag{2.37}$$

An approximate method for finding the most likely labeling, or MAP estimate (2.30), involves changing the summation in the factor-to-node messages to a max

$$m_{C \to i}(y_i) = \max_{\mathbf{y}_{C \setminus \{i\}} \in Y_{C \setminus \{i\}}} \exp U_C(\mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta}_C) \prod_{j \in C \setminus \{i\}} m_{j \to C}(y_j). \tag{2.38}$$

Using this form in the message passing is called the max-product algorithm.

Greater detail about factor graphs and approximate inference may be found in articles by Kschischang et al. [60] and Yedidia et al. [136].

2.4 Summary

In this chapter, we have reviewed the framework of modeling a predictive probability distribution using a discriminative Markov model. These flexible models allow us to conjure all sorts of associations and dependencies between unknowns that may be useful for accurate prediction. Training these models inevitably requires approximations. We can simplify training data requirements by decoupling parameters and simplify even already approximated optimizations with a piecewise decomposition. The belief propagation algorithm can be used to make predictions with the resulting predictive probability.

These models will be used throughout the thesis. By defining a compatibility that compares the texture of neighboring regions, we use data-dependent spatial context to improve the detection of text and signs. We may also make use of what little text may be present in a region by creating a function that compares two characters, rating whether they are the same or different. Thus, even when the compatibility function responsible for identifying characters by their appearance is weak or misled by local context, the (dis)similarity of a character compared to others can be used to boost accuracy. How does recognition affect detection and vice-versa? By defining the associations between the unknowns, we can easily incorporate any available information sources simultaneously in an integrated framework that reports the probability of a hypothesis, a meaningful, interpretable, and comparable number that is ultimately useful for making predictions.

CHAPTER 3

TEXT AND SIGN DETECTION

Before text and signs may be recognized, one must determine where they are in a scene. Setting aside for the moment the fact that detection ultimately is a form of recognition, we focus in this chapter on improving interpretation-free detection. Several such bottom-up feature-based approaches to text detection were reviewed in Chapter 1 (§1.2). Most systems presently in use rely on some form of machine learning to achieve good performance on the detection task. While the best approaches use very powerful and discriminative features, many still rely on heuristic rules for doing layout analysis on regions identified as text. Heuristics have limited application because they tend to be more difficult to craft as target data becomes more complex, and their parameters are seldom optimal. For this reason, we advocate a more complete machine learning approach for detection.

Text detection approaches with integrated, learned layout analyses have only recently appeared. One relies on uninformed segmentation [137], while the other uses an edge detector [104]. Both of these methods have drawbacks, as described in Section 1.2. Our method, proposed contemporaneously, is designed to robustly detect text and signs at many scales and world orientations through the use of powerful local texture features. However, the main contribution is the inclusion of data-dependent spatial context for layout analysis with a model learned from training data.

Portions of this chapter are reprinted, with permission, from previous papers appearing in the 2004 IEEE International Workshop on Machine Learning for Signal Processing [124], ©2004 IEEE, the 2005 IEEE Workshop on Computer Vision Aids for the Visually Impaired [107] ©2005 IEEE, and a technical report [123].

3.1 Overview

Signs may be of arbitrary size, color, and shape, and if the definition is broadened beyond text to other visual classes of signs, the scope of the term "sign" becomes enormous (see Figure 3.1). While previous work on the more specialized text detection tasks has either used a local classifier or layout heuristics, the more generic sign detection task is more complicated. Heuristics are more prone to failure or will become increasingly difficult to engineer as we allow the textual input to have more complicated (though still human readable) layouts. Moreover, signs containing or solely consisting of logos also have properties that can be described well with features we will apply to the problem. This difficult detection task will require a generic, but

Figure 3.1. Example input scene for sign detection. Signs may contain a variety of fonts, perspectives, lighting conditions, and logos.

very powerful method for classifying sign and non-sign regions. For us, this comes in the form of a discriminative Markov field.

The advantages of this model are twofold. First, since it is discriminative, it need not describe how the data is generated but only how to distinguish among the classes of interest. Traditional machine learning methods like logistic regression or neural networks and newer techniques such as AdaBoost and Support Vector Machines are all examples of discriminative models that have been successfully applied to the text detection problem. The second advantage is that discriminative Markov fields can model the spatial context and dependencies among regions of an image. Formal probabilistic models accounting for such dependencies in images have existed for two decades [37], however they are now seeing more widespread use with greater computational power, effective approximate inference methods, and especially the advent of the discriminative model, which often produces better results.

The value of contextual information in computer vision tasks has been studied in various ways for many years (e.g., [127, 110, 54, 82, 61, 117, 18]). Two types of context are important for the region labeling problem: label context and data context. By data context, we mean the image data surrounding any region in question. Data context can be of almost any scale, from the immediate vicinity of the region to the entire image or an image sequence. Similarly, label context consists of any labels surrounding a region in question. In the absence of label context, local regions are reasoned about independently. Disregard for the (perhaps unknown) configuration of labels often leads to isolated false alarms and missed detections upon classification. Likewise, the absence of data context means ignoring potentially helpful image information from around the region being classified. Both types of context are important.

For example, since neighboring regions often have the same label, we could encourage smoothness by penalizing label discontinuities. Such regularity is typically imposed without regard for the actual data in the regions. The downside is that when the local evidence for a label is weak, the continuity constraints typically override the local data. On the other hand, if the neighboring data is considered, local evidence for a region to be labeled "sign" might be weak, but witnessing a strong edge in the *neighboring* region could bolster belief in the presence of a sign at the site because the edge indicates a transition. Thus, we will consider the labels *and* data of neighboring regions when making classification decisions.

The next section details the structure of the model used for contextual text and sign detection. After that we briefly review the image features, both local and contextual, used with the model. Then we demonstrate the results with a set of experiments contrasting local detection methods with the contextual approach. We conclude the chapter by enumerating the contributions of this approach.

3.2 Markov Field for Detection

Markov fields were reviewed in Chapter 2. For the detection task, the unknowns will represent small regions of the image, and each region takes the label of either sign or background. We will decompose an input image into a regular grid of squares and assign labels to these regions. Because its discriminative nature easily allows it, the evidence for labeling each region will be drawn from parts of the image beyond the region itself, but the net effect is that each pixel belonging to a region is assigned the same label. This has practical effects and will require some special treatment of the data for training. We discuss this after introducing details of the model.

3.2.1 Detection Model

In this context, $\mathbf{y}$ represents the grid of squares, and $\mathbf{x}$ is of course the observed image. The usual discriminative Markov field model (repeated here for convenience)

$$p\left(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I\right) \equiv \frac{1}{Z\left(\mathbf{x}\right)} \exp\left\{\sum_{C \in \mathcal{C}} U_C\left(\mathbf{y}_C, \mathbf{x}; \boldsymbol{\theta}_C\right)\right\} \tag{3.1}$$

will employ two types of compatibilities. One is local, relating image features to a label for a particular region, and the other is contextual, relating image features to the labels for pair of neighboring regions. Both of these take the usual linear form

$$\begin{aligned} U_i\left(y_i, \mathbf{x}, \boldsymbol{\theta}_i\right) &= \boldsymbol{\theta}_i\left(y_i\right) \cdot F_i\left(\mathbf{x}\right), \; i \in V & (3.2) \\ U_{ij}\left(y_i, y_j, \mathbf{x}, \boldsymbol{\theta}_{ij}\right) &= \boldsymbol{\theta}_{ij}\left(y_i, y_j\right) \cdot F_{ij}\left(\mathbf{x}\right), \; i \sim j, & (3.3) \end{aligned}$$

where $i \sim j$ indicates that regions i and j are neighbors in the grid. Note that for parsimony we have modified the more general set-based indexing of the compatibility functions, i.e., U_C where $C = \{i\}$, to a simple subscript U_i. The factor graph representation of the model is shown in Figure 3.2.

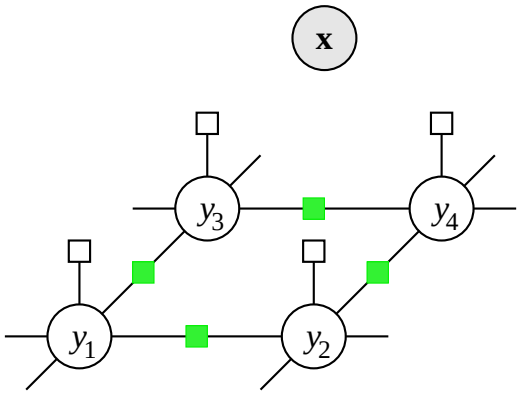


Figure 3.2. Factor graph for contextual detection. White factors relate image features to a single label, while shaded (green) factors associate image features such as texture gradients with pairs of neighboring labels. All factors are implicitly dependent upon the observation $\mathbf{x}$, but these edges are omitted for clarity.

Although the F_i indicates the region of the image $\mathbf{x}$ from which the features are to be extracted, they are the same features for all regions. Similarly, F_{ij} are the same features, but extracted from different regions. Thus, the field is effectively homogeneous (translation invariant). While the unique compatibilities rely on features from a particular region of the image, all compatibilities of the same class (local or contextual) use the those features in the same way.

The parameters $\boldsymbol{\theta}_i$ are "tied" for all regions; that is, $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j$ for all $i, j \in V$. However, we use anisotropy in the contextual factors, giving up rotational invariance, so that the model may learn any orientational bias of the labels that may be manifest in the data. Specifically, this means that while some contextual parameters $\boldsymbol{\theta}_{ij}$ are tied, they are only tied between compatibility functions that represent the same orientation between neighbors, horizontal or vertical. Furthermore, the field is not symmetric. That is, $\boldsymbol{\theta}_{ij}(y_i, y_j) \neq \boldsymbol{\theta}_{ij}(y_j, y_i)$, so that "left-of" and "above" arrangements are distinct from "right-of" and "below." Once again, this flexibility allows the model to learn any preference for making such distinctions from the data.

3.2.2 Model Training

Breaking the image into regions is a way of reducing the computational overhead of image labeling, since it does not require that classifications be made for every pixel. This is a reasonable approximation since local image properties like texture, which forms the basis of our image features, tend to be fairly static except at meaningful boundaries (e.g., between a building and sky). However, the strategy can be problematic when the training data contains image regions that straddle such boundaries. Unless the model or training method is designed to handle such a grouping of heterogeneous data, the approximation may be detrimental to performance.

Depending on the objective of the system, this issue is related to the so-called "multiple instance learning" (MIL) problem [27]. There, a set of instances are provided

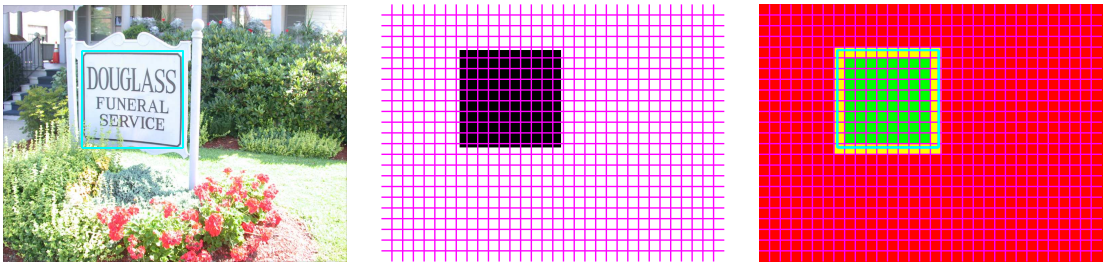


Figure 3.3. Decomposition of images into regions on a grid. LEFT: Original image with ground-truth contour. CENTER: Grid overlaid on foreground sign mask. RIGHT: Pure sign (green), background (red), and mixed (yellow) areas.

to a classifier and it must determine whether any positive instances are present. During training, the label of the *set* is provided (whether any positive instance is present in the set), but the positive instances are not identified. When boundaries are manually drawn around signs and text in an image, some grid squares naturally straddle these boundaries. In this case, the positive instances would be the pixels within the drawn boundaries (see Figure 3.3).

The question for us becomes, given the Bayesian training methodology defined in Chapter 2 (§2.2), how should this problem be handled? We propose and investigate a few alternatives involving how grid regions that straddle manually drawn boundaries should be treated. In all cases, grid regions that do not straddle a boundary consist of purely sign, or purely background pixels. Such regions may clearly be given Sign or Background labels in the training data. Possible alternatives regarding the remaining regions include

1. Giving them positive Sign labels, as in the MIL framework
2. Giving them negative Background labels, thereby saying we are only interested in detecting purely foreground regions
3. Give them a label by thresholding the percentage of pixels manually labeled sign
4. Giving them no labels at all, effectively omitting them from the training data

We explored options 1 and 3 in previous work [124, 123], and found them less than satisfactory. Later, we partially attempted option 4, but abandoned the spatial context of the model to do it [107].

We found that using only pure regions—those containing only sign or no sign at all—we could achieve better results with a strictly local classifier trained on this type of data [107], than with the contextual classifier on the MIL-like data [123]. The features and the Markov field model are not equipped to handle the MIL situation optimally. However, context and dependency modeling can still bring important performance improvements. Therefore, rather than abandoning the contextual advantages of the Markov field model or forcing the data to be labeled in a suspect fashion,

we will use the laws of probability to incorporate all of the spatial information, while leaving the mixed regions unlabeled. This allows the model to form its own beliefs about whether such regions should be labeled Sign or Background, without attempting to train it to take an unnaturally rigid stance on such data.

A naïve approach for giving the mixed regions no labels is to literally omit them from the training data. This is reasonable for a local model, where predictions are made independently, but it is the wrong approach once spatial dependencies are introduced. In the grid for the image, the mixed regions would simply be removed from the graph, eliminating all compatibility functions involving such regions. This will be problematic because features and nodes at the interface of Background and Sign regions are discarded and the model cannot learn the properties of the transitions between different region types.

Properly training the model to incorporate the spatial dependencies with incomplete labels involves using a marginal conditional likelihood in the parameter posterior. If $\mathcal{D}_{\mathbf{y}}$ only contains *some* of the labels for the images $\mathcal{D}_{\mathbf{x}}$, then there are "missing," unobserved labels $\mathcal{D}_{\mathbf{u}}$ that must be accounted for. These are the yellow regions in Figure 3.3.

Recall that maximizing the parameter posterior $p(\boldsymbol{\theta} \mid \mathcal{D}, I)$ involves the likelihood $p(\mathcal{D}_{\mathbf{y}} \mid \mathcal{D}_{\mathbf{x}}, \boldsymbol{\theta}, I)$. By the product rule of probability, we have that

$$p(\mathcal{D}_{\mathbf{y}} \mid \mathcal{D}_{\mathbf{x}}, \boldsymbol{\theta}, I) = \frac{p(\mathcal{D}_{\mathbf{y}}, \mathcal{D}_{\mathbf{u}} \mid \mathcal{D}_{\mathbf{x}}, \boldsymbol{\theta}, I)}{p(\mathcal{D}_{\mathbf{u}} \mid \mathcal{D}_{\mathbf{y}}, \mathcal{D}_{\mathbf{x}}, \boldsymbol{\theta}, I)} \tag{3.4}$$

for all values of $\mathcal{D}_{\mathbf{u}}$. Therefore, to compute the marginal conditional likelihood (3.4), we may pick any values we like for the unobserved labels $\mathcal{D}_{\mathbf{u}}$. Computing the numerator is the same as when all the labels were observed (since we may pick any assignment for $\mathcal{D}_{\mathbf{u}}$). Because we have a Markov field, it is straightforward to enter the given labels $\mathcal{D}_{\mathbf{y}}$ as evidence and compute the conditional likelihood of the denominator in a similar fashion. Using the product rule is much simpler than explicitly marginalizing $\mathcal{D}_{\mathbf{u}}$ from the joint probability in the numerator via summation, yet we still refer to it as a marginal.

Training our model to be more effective with pure regions while incorporating spatial dependencies now only involves substituting the marginal conditional likelihood (3.4) into the parameter posterior (i.e., Eq. (2.10) on page 23) for optimization. We show in the experiments of section 3.4 that the model with spatial context greatly improves detection over the typical local model and that properly marginalizing unobserved labels during training is necessary for accurate results.

3.3 Features

A detailed description of the features we use for sign detection may be found in a prior technical report [123], but we very briefly review them here, followed by details of how they are actually employed by our model.

3.3.1 Feature Overview

Rather than simply using functions of single filters (e.g., moments) or edges, we use a richer representation that captures important relationships between responses to different scale- and orientation-selective filters. To measure the general textural properties of both sign and especially background image regions, we use the statistics of filter responses described by Portilla and Simoncelli [93], which include the correlations between responses to steerable pyramid filters at different scales and orientations. A steerable pyramid of four scales and four orientations is computed over the entire image. Statistics of the filter responses within each region are then calculated.

A biologically inspired non-linear texture operator for detecting gratings of bars at a particular orientation and scale is described by Petkov and Kruizinga [92]. Scale and orientation selective filters, such as the steerable pyramid or Gabor filters, respond indiscriminately to both single edges and one or more bars. Grating cells, on the other hand, respond selectively only to multiple bars (three or more). This property is an ideal match for detecting text, which is generally characterized by a "grating" of strokes. The filter is implemented by a non-linear function ensuring strong responses to bar detectors (polarity-sensitive oriented filters) along a receptive field line orthogonal to the bars. We use as features local statistics of these filter responses for several filter scales. In particular, responses to grating filters at eight scales are computed only for vertical strokes with a horizontal orientation. Like the steerable pyramid filters, statistics of the grating filter responses within each region are then calculated. We should note that although the statistics are calculated only for responses within a particular grid region, the grating filter responses can depend on image data far beyond this region. This way, text at scales much larger than the individual grid regions can be detected.

Additionally, we use normalized histograms of the hue and saturation within each grid region. This also allows us to measure color discontinuities between regions. Since hue can be unreliable when saturation is low, only pixels with saturation exceeding a threshold (here, 0.1 on a $[0, 1]$ scale) are counted in the histogram. Furthermore, the hue histogram is weighted to so that each pixel's contribution is its saturation. Thus, weakly colored pixels do not contribute as much to the color distribution. We use eight evenly spaced bins for hue to roughly characterize the color and ten bins for saturation to avoid over-quantization with relatively small windows.

A graphical overview of all these features is shown in Figure 3.4 on the following page.

3.3.2 Feature Details

Here we describe the two classes of features used in the model: the local features F_i and contextual features F_{ij}. As alluded above, local features F_i are comprised of the various steerable pyramid and grating filter statistics, along with the color histograms. These are more precisely decomposed into the following sets; we describe the nature of each *type* of set and the number of sets implied:

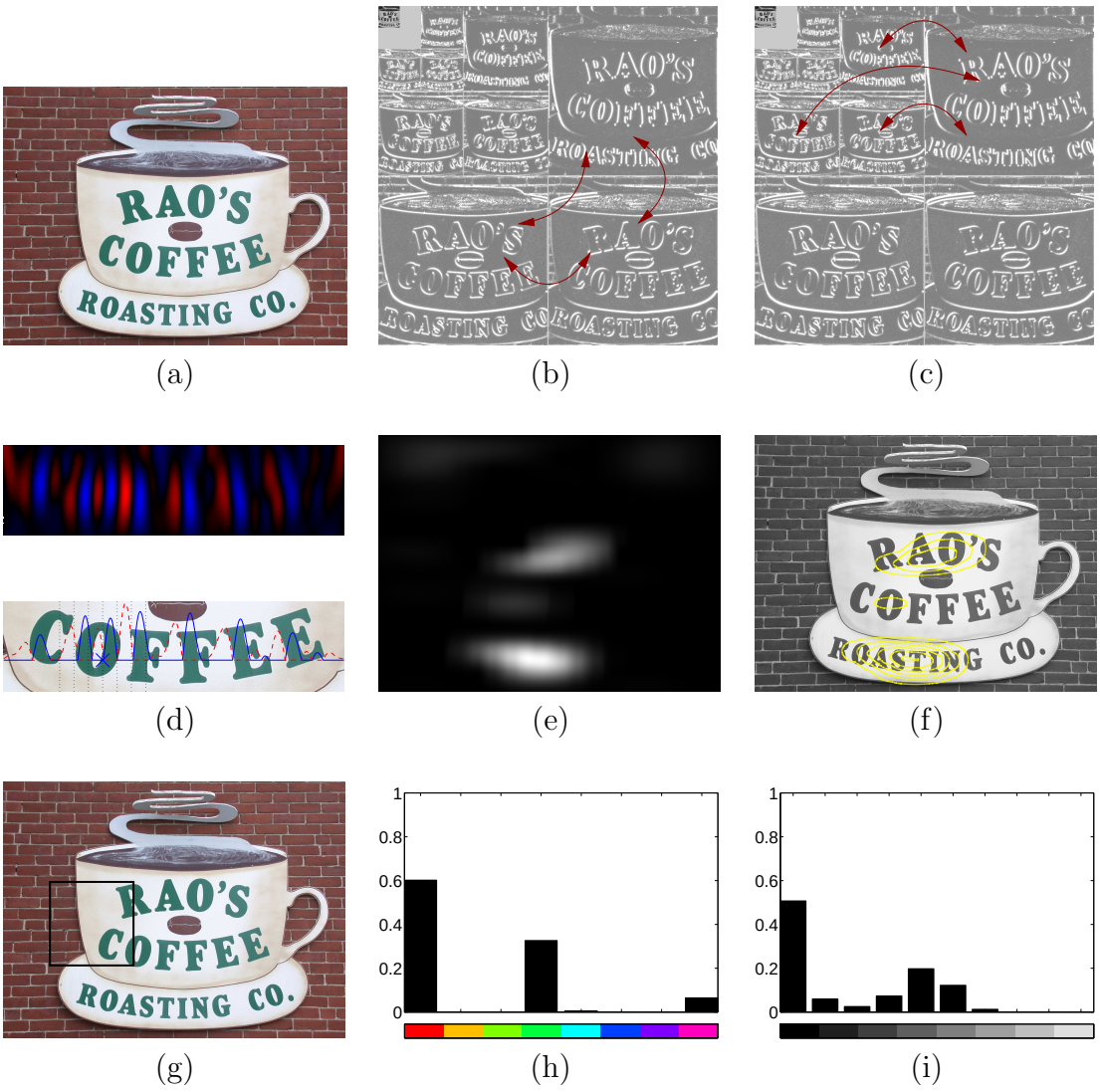


Figure 3.4. Graphical overview of detection features on (a) an input image. For an image region, statistics including the cross-correlation of filter responses at (b) different orientations of the same scale and (c) different scales of the same orientation are calculated. Non-linear grating filters are used to help detect text for a particular scale and orientation. Strong responses to (d) polarity sensitive oriented filters are required along a receptive field line and combined to give (e,f) a grating response. For (g) an image decision region, normalized histograms of (h) hue and (i) saturation are also used.

- Raw pixel statistics (e.g., range and central moments) [1]
- Low-pass image statistics
  - Kurtosis [1]
  - Skew [1]
- Average magnitude
  - Pyramid channels [16]
  - Low-pass image [1]
  - High-pass residual [1]
- Auto-correlation of low-pass image at each pyramid level [5]
- Auto-correlation of magnitude of each channel and pyramid level [16]
- Cross-correlations between channels at the same pyramid level
  - Magnitudes [1]
  - Real-values [1]
- Cross-correlations between channels of the same orientation and neighboring pyramid levels
  - Magnitudes [1]
  - Real-values [1]
- Variance of the high-pass residual [1]
- Color histograms
  - Hue [1]
  - Saturation [1]
- Grating filter statistics [1]

For instance, when we calculate the auto-correlation, we keep the 13 unique central values of a 5×5 window. Each of these values is collected into one set of features. Since there are four scales (or pyramid levels) and four orientations (or channels) in the pyramid there are 16 such sets. Furthermore, all of the magnitude cross-correlations between channels at the same level are collapsed into one set.

These 50 groups are then used for the contextual features. Each is a set of image features for a local grid region. To compare neighboring image regions, we use as contextual features the ℓ_2 norm of the difference of each set. These norms of differences between sets are collected in the 50 contextual features F_{ij}.

Figure 3.5. Example scene image with ground truth contours around sign regions.

3.4 Experiments

In this section we describe experimental results testing our contextual discriminative model for sign and text detection against purely local classifiers and a layout heuristic. First we briefly outline the experimental data, followed by our procedure and results.

3.4.1 Experimental Data

Images of downtown Amherst, MA were collected with a Nikon Coolpix 995 during September 2002. A total of 309 24-bit color images of size 1536×2048 were taken during various times of day throughout the month.[1] Signs in the image were then labeled (masked) by hand. The general criteria for marking a sign were as follows:

- it should be readable (if text) or recognizable (if a logo) at 25% magnification (384×512)
- if a sign's natural border is reasonably close to the text or logo, extend the mask to the border, otherwise the mask should extend to roughly 128-256 pixels in any direction beyond the text or logo.

It should be noted that these were not hard and fast rules, but heuristic guidelines. There could be small signs present in an image that are not labeled as such because they can not be read legibly, but could still have their text detected. The second guideline allows color continuity and presumed edge evidence for signs to be used to a reasonable extent. An example is shown in Figure 3.5.

[1] Available from `<http://vis-www.cs.umass.edu/projects/vidi>`.

3.4.2 Experimental Procedure

For our experiments, we scaled the images by half to 768×1024 and measured the features described in Section 3.3 for overlapping 64×64 regions. This yields a 23×31 grid. Whereas the masks described above have roughly pixel precision, these regions might be composed of all sign, no sign, or something in between. As described in Section 3.2.2, how these regions are labeled for training and testing impacts the results.

In preliminary work, we treated the task as a binary classification problem [124]. Here, we give only those regions entirely consisting of masked pixels the label Sign and those regions entirely consisting of unmasked pixels the label Background. All other regions (those straddling the boundaries in the manual segmentations) were left unlabeled.

A contextual model with any mixed regions removed during training will likely not have any neighboring nodes of differing classes. The spurious learned result is an expected value of zero for any feature on an edge between Sign and Background regions. To prevent this, we set $\boldsymbol{\theta}_{ij}(y_i, y_j) = 0$ for $y_i \neq y_j$ when training with these regions removed. Although this improves the results over leaving such parameters free during learning, we find in the next section that simply removing the mixed regions performs much worse than the alternatives, even the context-free local model.

A "local" or "context-free" model results when only the local compatibility functions U_i in Equation (3.2) are used in the probability (3.1). In training such a model, the unlabeled regions cancel from the marginal likelihood optimization (3.4). We employ a Gaussian prior (2.24) on both the local parameters $\boldsymbol{\theta}_i$ and the contextual parameters $\boldsymbol{\theta}_{ij}$, however the Gaussian parameter σ is not necessarily the same for both classes of parameters. Next, we describe how this "hyper-parameter" is chosen.

The images are evenly and randomly split into ten subsets for evaluation. In a ten-fold cross-validation procedure, we iteratively held out one set for testing. Of the remaining nine sets, four were used for an initial training set where we trained models with several different values of the hyper-parameter σ (evenly spaced on a log-linear scale). The value of σ that yielded the highest likelihood on the remaining five (of nine) sets was used to train on all nine of the training subsets. When an optimal σ was found for the local compatibility function parameters in a context-free model, this σ was fixed and an analogous (one-dimensional) search for the optimal σ for the contextual compatibility function parameters was conducted. The tenth subset was then used for evaluation of the model trained on the other nine, and all ten folds are collected for the results presented below.

To try and emulate the contextual nature of the discriminative Markov field, we also compare a simple hysteresis method. Using the local classifier, anywhere a prediction of Sign is made with a particular threshold $p > \tau$, any neighboring region (in a four neighbor system), is also classified as Sign if it is above the lower threshold $p > \tau/2$.

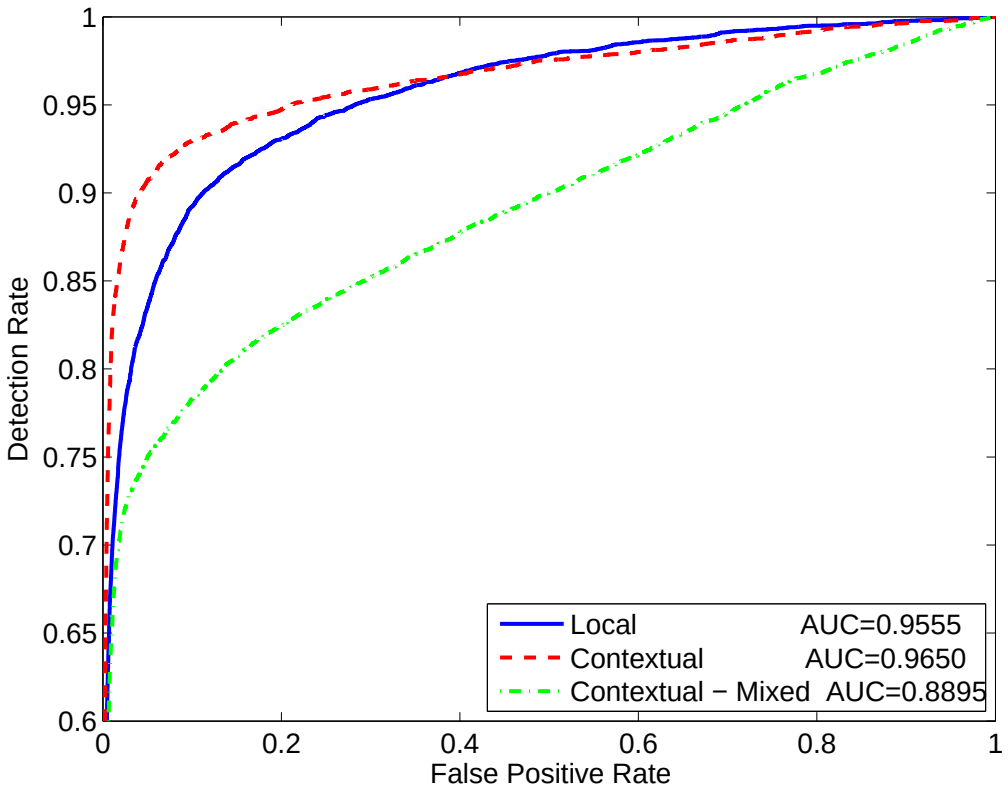


Figure 3.6. Comparison of local and contextual sign and text detectors with the receiver operating characteristic (ROC) curve as parameterized by the posterior marginal of each region.

3.4.3 Experimental Results

With our features, we have compared the results using only a local discriminative classifier to the contextual discriminative Markov field. Evaluating the results on regions that are purely sign or background, we show in Figure 3.6 three receiver operating characteristic (ROC) curves based on the posterior marginal probability $p\left(y_i \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I\right)$ for each region. The curves characterize the trade-off between false positives and false negatives. The results of two methods for training the contextual model are shown: one that optimizes the marginal likelihood and one that simply eliminates mixed regions from any graph, maximizing the resulting (fully observed) likelihood. The unqualified "contextual" model refers throughout to the one learned using the marginal likelihood.

Figure 3.7 shows some example detection results. The signs missed entirely by the contextual classifier are shown in Figures 3.8 and 3.9 on page 44. Note that the absolute pixel scale of the signs are the same both within and across the two figures.

Tables 3.1 and 3.2 evaluate the detections at the sign level, rather than the component region level. With MPM prediction (c.f., Eq. (2.31)), the threshold for classifying

Figure 3.7. Example contextual detection results: green (solid) boxes indicate correctly detected regions, while red (dashed) boxes are false positives. Cyan contours are the manual ground truth.

Figure 3.8. All the difficult or unusual signs missed by the contextual detector.

Figure 3.9. Conspicuous signs missed by the contextual detector.

Table 3.1. Sign detection results with MPM prediction.

	Local	Local + Hysteresis	Contextual	Contextual - Mixed
Sign Detection Rate	85.9±1.9%	87.4±1.8%	84.7±2.0%	100±0%
Average Coverage	74±1.6%	80±1.4%	88±1.2%	100±0%
Median Coverage	81%	89%	100%	100%
False Pos Regions/Image	1.1±0.097	1.1±0.093	0.47±0.050	0.91±0.034
False Pos Area/Image	0.97±0.097%	1.3±0.12%	1.0±0.14%	67±1.9%

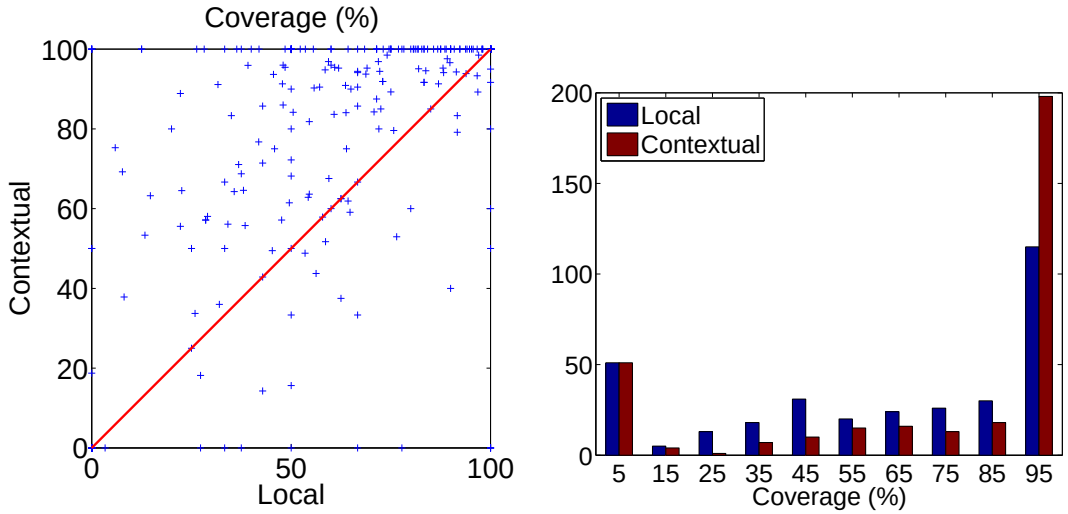


Figure 3.10. Comparison of sign coverage between the local and contextual classifiers with MPM prediction. LEFT: Scatter plot of the coverage of the local and contextual classifier on each sign. The diagonal line indicates equal coverage performance. RIGHT: Histograms of coverage for each classifier.

a particular region as Sign is $p > 0.5$. In both tables, *sign detection rate* is the percentage of signs having at least one region covering it labeled as Sign. *Coverage* is the percentage of regions detected on a sign. *False positives* are counted as the number of connected areas, rather than individual regions. This is because in a visual sign recognition framework, the connected areas are passed as input to a recognizer. To give an idea of scale, we also present the total false positive *area*, as a percentage of the image size. Figure 3.10 quantitatively compares the sign coverage for each classifier, showing how much of each sign is detected, while the examples in Figure 3.11 on the following page qualitatively illustrate this difference.

Using the ROC curve in Figure 3.6, in a *post hoc* fashion we determine the equal error rate (EER) for each classifier at the region level. EER is the point on the curve, parameterized by the classifier threshold on marginal probability, where the region false positive rate equals false negative rate (the complement of the region detection

Figure 3.11. Visual comparison of sign detection coverage with local (left) and contextual (right) classifiers.

Table 3.2. Sign detection results at the *region-level* equal error rate for prediction threshold.

	Local	Context
Patch Equal Error Rate	10.5%	7.8%
Threshold	$p > 0.0269$	$p > 0.0026$
Sign Detection Rate	97.6±0.84%	97.0±0.94%
Average Coverage	93±0.81%	92±0.99%
Median Coverage	100%	100%
False Pos Regions/Image	9.9±0.44	10±0.42
False Pos Area/Image	13±0.66%	43±1.5%

Figure 3.12. Signs detected with logos (rather than text) as prominent features.

rate). Table 3.2 reports the rates and the thresholds at which they are found along with the resulting sign detection metrics.

Figure 3.12 shows signs where a logo is equally or more prominent than any text.

3.4.4 Discussion

With the addition of contextual information, the "missing" area under the ROC curve is reduced by more than 21%, so the contextual model achieves more detected regions for the same number of false positives. Example detection results in Figure 3.7 show that text and signs at multiple scales, orientations, typefaces, and languages can be detected, even on backgrounds that are typically associated with non-text (e.g., brick and wood grain). The performance of the contextual classifier with mixed regions simply removed during training performs much worse than the other models. This confirms that when spatial context is important, removing nodes from the graph during training is the wrong approach and results in optimizing the "wrong" function. With one model, we can detect all manner of signs, including those that are mostly or only logos (Figure 3.12).

We may draw a few conclusions from Table 3.1. First, the default sign detection rate is perhaps low. However, a more detailed analysis of the data shows things are not as bad as they may initially appear. Of the 48 signs that are missed completely by the contextual classifier, 41 suffer from one or more of the following conditions: small and blurry (19%), projective foreshortening (15%), low contrast (44%), behind glass with specular reflection (35%), or text at an uncommon orientation (11%). Horizontal

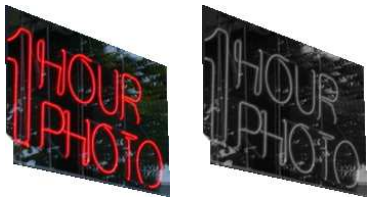


Figure 3.13. Color and grayscale image comparison. Although color features are used, wavelet features (and any image gradients) are calculated on the gray scale image, which has low contrast when compared to the color image.

and vertical text ($\pm 30°$) is most common, so these textural properties are learned by the model, but any remaining text ($45 \pm 15°$) does not appear often enough to be learned well. Signs do not need to be strictly parallel to the image plane, as many signs with foreshortening are in fact detected. However, failure can occur when the projective distortion shrinks the space between letters enough to virtually eliminate the edges and grating effect. The MPM threshold is conservative, keeping the false alarm rate very low, but causing these signs to be missed. By lowering the threshold, we can detect 97% of the signs, although with many more false alarms.

In addition, although color features (histograms) are used, the features that are similar to image gradients are calculated on the grayscale image. Figure 3.13 shows that while some text is easily seen in the color image, it can be very low contrast in the corresponding grayscale image. Signs like this are thus frequently missed by the detector (Figure 3.8).

The net effect of removing the mixed regions from the training data is a severe over-fitting of the "continuity" properties of the model. That is, all of the signs are detected, but strong label continuity effects cause too much of the image to be labeled Sign, resulting in a drastic increase in false positives.

Interestingly, there is no significant difference[2] between the base sign detection performance of the local and contextual classifiers. However, by studying the coverage of the detected signs we can quickly see the stark difference. The contextual classifier does a significantly better job of detecting more of each sign. In fact, the discriminative Markov field has greater coverage than the local classifier on 90% of the signs. Moreover, on the signs it detects, the contextual detector has complete coverage of 56% of them, compared to 34% for the local detector.

We also note that the contextual detector has half as many false positive regions per image. If a recognizer is complex or computationally intensive, reducing the number of false positives is important for overall system speed.

Finally, we examine the equal error rate (EER) performance of each classifier in Table 3.2. By dropping the acceptance threshold, many more signs are detected, and the coverage of the local classifier rises to meet the discriminative Markov field. However, there are *many* more false positives. Thus, while recall (true positives or detections) is high, the precision (or true negatives) suffers greatly.

[2]Significance is assessed by a paired, two-sided sign test.

In conclusion, the "default" prediction mode (MPM) tends toward slightly lower sign-level recall, for both methods but the discriminative Markov field yields fewer false positives and a markedly greater coverage of the signs it detects. Moreover, nearly all the signs that are missed could be detected with richer features and training data; the rest may simply too difficult to detect without filtering false positives with interpretation. We discuss this further in the chapter's conclusions.

3.5 Contributions

There are three primary contributions of our model and approach to generic text and sign detection: the use of a learned contextual dependency, simultaneous multi-scale analysis, and using marginalization to discard boundary cases while preserving spatial context for improved performance.

3.5.1 Learned Layout Analysis

While many others have employed discriminative methods for detecting text, none of these have included a learned model for incorporating spatial dependencies. Our aim here is to find any incidental text or signs in an image. As such text goes beyond ordinary horizontal or vertical layouts or undergoes perspective distortions, manual heuristic layout analysis will become more complex, and likely more brittle. Our model requires no hand-tuning of parameters yet still takes advantage of the frequent spatial continuity of text regions.

One other learned layout analysis model appeared contemporaneously [137], but it suffers from the limitation that it depends on a prior, uninformed segmentation stage before classifying regions as text or non-text. As noted earlier, uninformed segmentation becomes virtually impossible as the image degrades. While the discriminative Markov field model described here only detects text regions (rather than give a binarization of text and background), we show in subsequent chapters how to move from text regions to detecting and identifying individual characters in a more informed, top-down fashion.

3.5.2 Multi-Scale Analysis

We calculate features in the image at several scales that capture local or very broad portions of the image. In most other approaches to multi-scale text detection, the features are computed for a particular scale of text, a classifier is subsequently applied for that scale, and this process is repeated for a range of scales.

In contrast, we similarly compute the range of multi-scale features, but these are all simultaneously used in one classifier that detects signs of any scale. To be sure, prediction in the Markov field model entails some computational burden, but this is outweighed by the benefit of a learned use of context, balanced by the elimination of multi-scale classification, and mitigated by the continuing development of faster hardware and approximate inference algorithms.

3.5.3 Spatial Context without Boundary Cases

Discriminative models with latent attributes have recently appeared [95, 112]. However, in training these models, the latent properties are always latent, and the observed properties are always observed. For example, the identity of an object is always given in training data, but the constituent parts are unlabeled, or in text, noun phrase chunking might be given while parts of speech tags are modeled but not observed. In these cases, the latent attributes serve an auxiliary purpose; they provide added structure to the model to aid in prediction of some other primary property of interest.

In our model, all unknowns are of the same type, namely, whether a region is sign. However, given our agglomeration of image pixels, there are some regions for which the complementary labels Sign and Background do not make sense. By freeing the model to focus discriminative capability on those regions that do meet the pure label criterion, we can improve the predictive performance of the model on these regions. We have also demonstrated that simply eliminating the portions of the model that do not meet the labeling guidelines performs rather poorly. The marginalization we have proposed is necessary for improved performance.

3.6 Conclusions

Our experiments have demonstrated the utility of linear and non-linear texture features for text and sign detection and a large improvement in detecting more of a sign by modeling spatial context.

A few modifications could improve performance. First, adding more extreme text orientations and low contrast signs to the training data could allow the system to detect more of the signs missing in Figure 3.8. Second, more complex features that employ color gradients might also aid in detecting the class of false negatives highlighted in Figure 3.13 and also make the entire system more robust. Finally, it is probable that the only way to reduce the large numbers of false positives incurred when the threshold is dropped to increase detection is to introduce a measure of interpretation. After discussing ways of unifying all of the information useful for recognizing text in the next chapter, we develop a method for more closely tying detection and recognition together during learning. Chapter 5 therefore discusses how we introduce interpretation to the detection process without necessarily explicitly performing recognition.

CHAPTER 4

UNIFYING INFORMATION FOR READING

As we have said many times, several sources of information factor into the reading process. This has been known for some time. What we require is a computational model that can piece these factors together in a unified framework. In this chapter, we propose a model that brings together bottom-up and top-down information as well local and long-distance relationships into a single elegant framework. Continuing with data-dependent contextual models, we present a method that smoothly integrates character similarity with traditional character appearance and language methods. The same unified probabilistic framework will also allow us to incorporate higher-level language information, such as a lexicon. Our model adapts to the data present in a small sample of text, as typically encountered when reading signs, while also utilizing higher level knowledge to increase robustness.

4.1 Overview

Generative hidden Markov models (HMMs) have long been a stalwart of research in OCR. Their advantages include incorporating some level of label context into the prediction process and an unsupervised training setting. Here, we present developments with discriminative Markov models, which are similar, but have many powerful advantages over HMMs.

Section 1.3.3.2 described how character similarity has been used to ensure that characters of similar appearance are given the same label. The principle drawback of these approaches was that the comparison and labeling stages were segregated in separate processes, with no or only ad hoc feedback. Thus far, the *dissimilarity* between character images has not been used as evidence *against* giving them the same label, but this is also a useful approach. The previous clustering-based methods only ensure that all cluster members are given the same label; they do not prevent different clusters from being assigned the same label. One model we present in this chapter smoothly integrates character similarity and dissimilarity with the recognition process.

Consider the example in Figure 4.1. The top row of text is the result of reading the sign at the left using only basic information about character images, and the

Information	Result
Appearance	`FIeet`
Appearance, Language	`Fteat`
Appearance, Language, Similarity	`Fleet`
Similarity → Appearance, Language	`Fteet`

Figure 4.1. A query image (left) is interpreted with varying amounts of image and linguistic information. Only when unified with similarity information is the other contextual information constrained to global consistency.

lowercase `l` (*ell*) is mistaken for an uppercase `I` (*eye*). The next result combines the image information with some basic language information. This does not correct the error but in fact introduces new errors. The image and language information is based on local context and does not require any global consistency. By factoring in character similarity information in the third line, the errors are corrected; the two `e` characters that appear the same are given the same label, while the `l` and `t` characters of dissimilar appearance are given different labels. In contrast, using similarity information first to determine which characters are the same and then identifying character clusters, as shown in the last line, does not perform as well as a unified model. This is particularly true when the number of characters is very small. We present in the first part of this chapter a model that incorporates all of these important information sources.

Higher-level knowledge like a lexicon also helps improve recognition. When images are of low-resolution, a lexicon can inform an otherwise unreliable character recognizer [52]. Additionally, humans are more reliable at reading characters when they are present in words, or pseudo-words [97]. Therefore it is important for us to consider how this information may be incorporated in the recognition process. We present a simple addition to our discriminative model that incorporates a bias for strings from a lexicon. Our model allows efficient approximate inference schemes by eliminating the need to consider all possible strings, only considering entries from the lexicon. Notably, we can speed this process even further by applying an approximate "sparse inference" technique: by eliminating characters with weak support from consideration, we drastically reduce the set of words that must be considered.

In total, by fusing the available information sources in a single model, we improve overall accuracy and eliminate unrecoverable errors that result from processing the information in separate stages.

4.2 Markov Models for Recognition

Just like the discriminative Markov field for detection in Chapter 3, a similar model for recognition involves defining parameterized compatibility functions for the data and the labels. For the recognition problem, the model input will be size-normalized character images and the output is the predicted character labels. In this section we

will outline the details of our model, including the form of the input and features, the relevant information being utilized, and the particular compatibility functions that are learned to form the model.

Our model and the subsequent experiments make the following assumptions:

1. The input is all of the same font
2. Characters have been segmented (that is, the coordinates of their bounding boxes are known), but not binarized
3. Word boundaries are known

Our conditioning information I consists of these in addition to our other basic information. Assumption 1 is especially reasonable for signs containing small amounts of text. Although there certainly exist exceptions to this, our database of signs has only a few that stretch the assumption, and it is not difficult to imagine introducing a factor for the proposition that two characters or words are in the same font when its truth is unknown. Assumption 2 is not ideal or the most general, but with high-resolution digital cameras, adequately lit scenes and an area of interest that occupies sufficient area on the sensor, it is reasonable. Note that it does not require a binary image (as shown in Figure 1.5 on page 12), only a rough localization of each character. Finally, assumption 3 is not overly restrictive since these could mostly be found by modeling intra- and inter-word character spacings. These assumptions are all warranted and are not overly constraining for the problem we are trying to solve, namely, reading short amounts of text found on signs in images of scenes. Looking ahead, Chapter 6 will present a model that eliminates assumptions 2 and 3, at the cost of sacrificing the use of character similarity information.

In the remainder of this section we build up our model from its constituent compatibility functions, based on the assumptions listed above. These will reflect several useful sources of information. Namely

- character appearance (*what do As and Bs tend to look like?*)
- language properties (*what letters tend to follow other letters? where do we expect capital letters?*)
- character similarity (*which characters look similar or different?*)
- lexicon (*is this string more likely to be elm or clm)?*

Each of these are combined effortlessly into a unified model for character recognition with the basic form outlined in Chapter 2, but slightly different from the manifestation used for detection in the previous chapter.

Let $\mathbf{x}$ be the input image representation of a string of characters to be recognized and $\mathbf{y}$ be the corresponding string of character labels. The index i now represents a particular character in the string, and the set of labels Y is the same for every

character $y_i \in Y$. Later in this section we describe the particular features and compatibility functions used for predicting $\mathbf{y}$ from $\mathbf{x}$, as outlined above. Eventually, we will want to compare the results of the model with various factors or information sources included. Since we denote the assumptions or information I that a particular model $p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I\right)$ employs, this is used to also indicate the information sources (compatibility functions, or factors) being used. To this end, when a particular class of compatibility function U^A is used, we indicate this by conditioning the model on the corresponding "information" I^A.

The model resulting from the combination of the compatibilities we will define is shown in Figure 4.2 on the next page. All of the various classes of factors may be combined in one model, but we show them in two separate graphs for clarity. The top graph focuses on the "adaptive" or locally consistent model that uses similarity between the character images from one font as part of the recognition process. The bottom graph demonstrates how factors may be introduced to promote the recognition of strings as lexicon words. Details of each of these factor types is given in the remainder of this section.

4.2.1 Appearance Model

The most obvious component of a recognition model involves relating character appearances to their identity. Gabor filters are an effective and widely used tool for feature extraction that decompose geometry into local orientation and scale [25]. Their success in handwriting recognition [26] and printed character recognition [22] demonstrates their utility for this task. Using a minimally redundant design strategy [75], a bank of 18 Gabor filters spanning three scales (three full octaves) and six orientations (30° increments from 0° to 150°) is applied to the grayscale image $\mathbf{x}$, yielding complex coefficients $\mathbf{f}$ that contain phase information. The real and imaginary parts of the filter are even and odd functions, respectively.

Taking the complex modulus of the filter outputs $|\mathbf{f}|$ provides phase invariance and makes the responses less sensitive to translations of the input; see Figure 4.3. Practically, this makes the filter responses invariant to the *polarity* of the text (white-on-black versus black-on-white). After filtering, the complex modulus of each response image is downsized by applying a Gaussian blur and downsampling. This adds a slight amount of insensitivity to feature location for different fonts, but it mostly serves to reduce the ultimate size of the feature vector used as input to the model. All of the downsized responses are collected into a single feature vector for each character, $\mathbf{d}_i$, which is a function of the original image $\mathbf{x}$.

Given a relationship between the identity of the character and the filter responses, which signify local scale and orientation, this information is denoted I^A because it is based on the appearance of the character image. We then associate character classes with these filtered images by a linear energy

$$U_i^A\left(y_i, \mathbf{x}; \boldsymbol{\theta}^A\right) = \boldsymbol{\theta}^A\left(y_i\right) \cdot F_i\left(\mathbf{x}\right), \tag{4.1}$$

where $F_i\left(\mathbf{x}\right) \equiv \mathbf{d}_i$. The same appearance parameters are used for every character, so there is no dependence of $\boldsymbol{\theta}^A$ on index i.

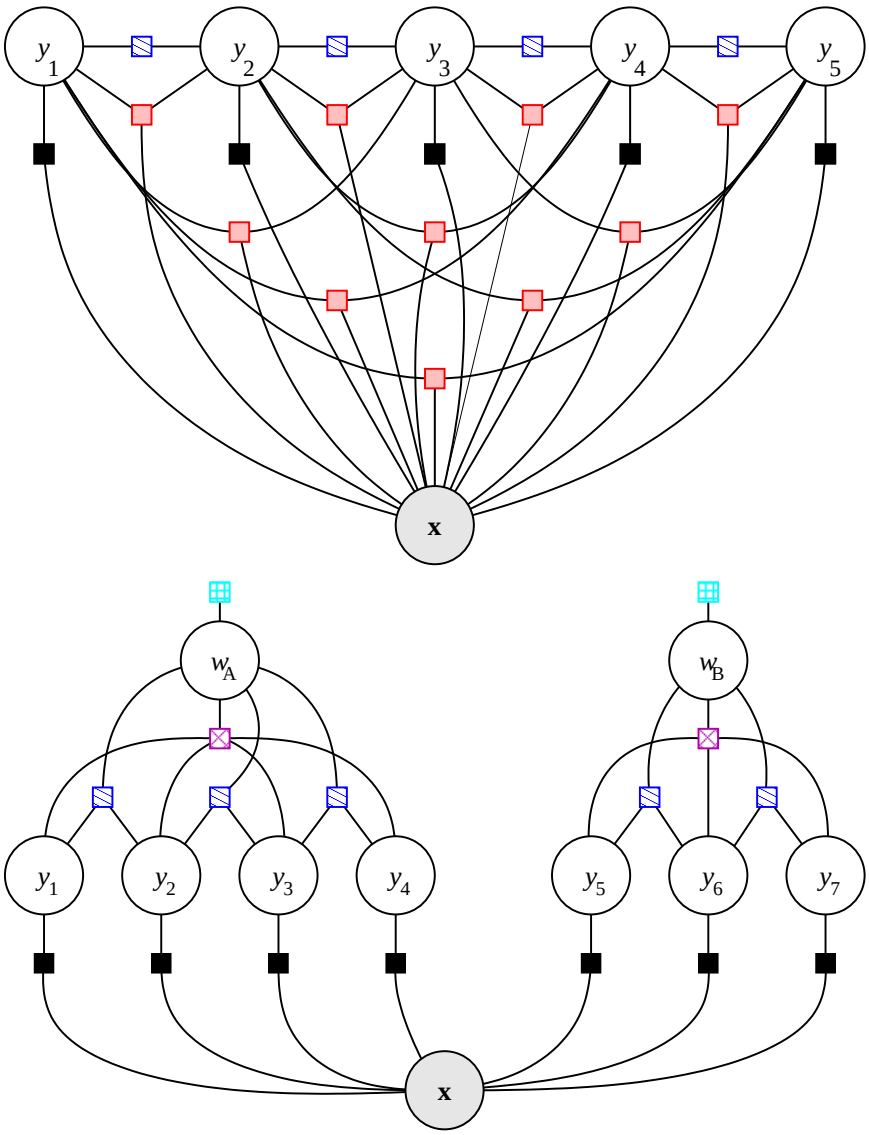


Figure 4.2. Factor graphs for inferring characters $\mathbf{y}$ from a given image $\mathbf{x}$. The solid (black) factors capture relationships between the image and character identity, I^A. Hatched (blue) factors between neighboring y capture language information including bigrams, I^B, and letter case, I^C. Shaded (red) factors among $\mathbf{y}$ account for similarities between characters in $\mathbf{x}$ for jointly labeling the string, I^S. Cross-hatched (magenta) factors can constrain portions of $\mathbf{y}$ to be drawn from a lexicon, I^W, while the tiled (cyan) factors capture the bias for lexicon words, I^L. TOP: Model corresponding to $p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I, I^A, I^B, I^C, I^S\right)$. BOTTOM: Model corresponding to $p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I, I^A, I^B, I^C, I^W, I^L\right)$.

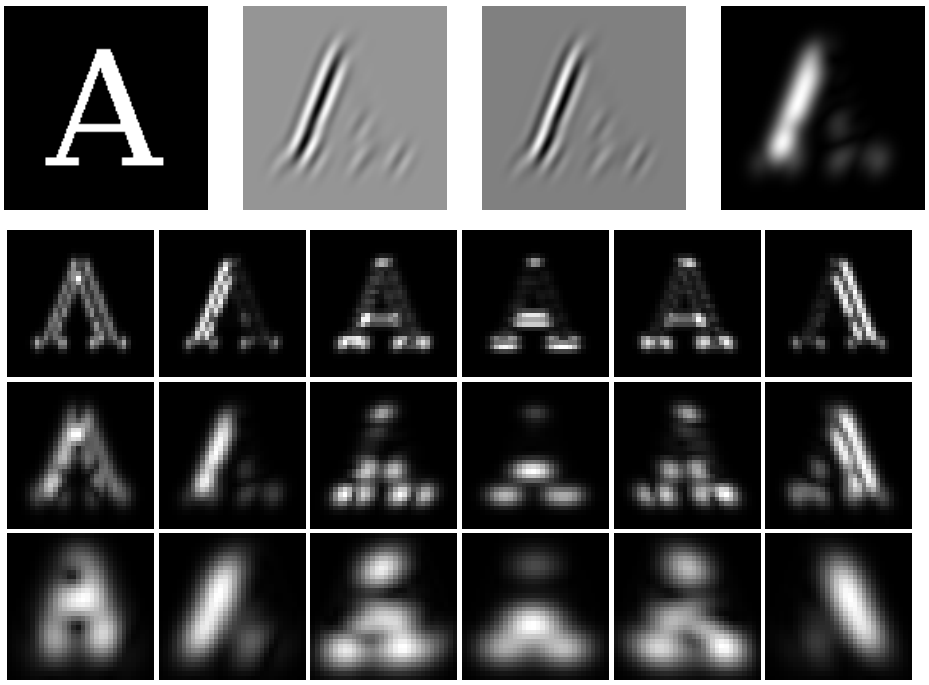


Figure 4.3. Gabor filter responses on a character input image for the appearance model. TOP: An example training character with (left to right) real, imaginary, and complex modulus filter responses for one orientation and scale. BOTTOM: Downsized filter responses for all orientations and scales.

4.2.2 Language Model

Properties of the language are strong cues for recognizing characters in previously unseen fonts and under adverse conditions. The many previous works reviewed in Section 1.3.1 have made use of it in various ways. We add simple linguistic properties to the model in the form of two information sources: character bigrams and letter case.

It is well known that the English lexicon employs certain character juxtapositions more often than others. N-grams are a widely-used general feature for character and handwriting recognition [14]. Our model uses this information I^B via the linear features

$$U_{ij}^B\left(y_i, y_j; \boldsymbol{\theta}^B\right) = \boldsymbol{\theta}^B\left(y_i, y_j\right), \tag{4.2}$$

where i and j are ordered, adjacent characters within a word. In this model, we do not distinguish letter case in the bigrams, so the weights in $\boldsymbol{\theta}^B$ are tied across case (i.e., $\boldsymbol{\theta}^B\left(\mathtt{R}, \mathtt{A}\right) = \boldsymbol{\theta}^B\left(\mathtt{r}, \mathtt{A}\right) = \boldsymbol{\theta}^B\left(\mathtt{R}, \mathtt{a}\right) = \boldsymbol{\theta}^B\left(\mathtt{r}, \mathtt{a}\right)$).

Prior knowledge of letter case with respect to words also proves important for accurate recognition in English. In some fonts, potentially confusable characters may have different cases (e.g., `l` and `I`, lowercase *ell* and uppercase *eye*). Since we do not binarize the images, there is no direct method for measuring the relative size of neighboring characters. We can improve recognition accuracy in context because English rarely switches case in the middle of the word. Additionally, uppercase to lowercase transitions are common at the beginning of words, but the reverse is not. Note that digit characters have no case. This information I^C is incorporated with the feature weights

$$U_{ij}^C\left(y_i, y_j; \boldsymbol{\theta}^C\right) = \begin{cases} \theta^{C,s} & y_i, y_j \text{ same case} \\ \theta^{C,d} & y_i, y_j \text{ different case} \\ 0 & \text{otherwise,} \end{cases} \tag{4.3}$$

when i and j are adjacent characters *within* a word and

$$U_{ij}^C\left(y_i, y_j; \boldsymbol{\theta}^C\right) = \begin{cases} \theta^{C,u} & y_i \text{ lowercase, } y_j \text{ uppercase} \\ 0 & \text{otherwise,} \end{cases} \tag{4.4}$$

when i and j are the first and second characters of a word, respectively. Thus ,for this letter case model, U^C, we have the parameters $\boldsymbol{\theta}^C = \begin{bmatrix} \theta^{C,s} & \theta^{C,d} & \theta^{C,u} \end{bmatrix}$. Note that the functions (4.3) and (4.4) still have the same general linear form as presented in Section 2.2, but we present their more compact, tied form here.

4.2.3 Similarity Model

An important, underused source of information for recognition is the similarity among the character images themselves—two character images that look the same should rarely be given different labels. Toward this end, we need a comparison func-

tion for images. We have found the vector angle between the concatenated real and imaginary parts

$$\mathbf{f}'_i = \left[\begin{array}{cc} \Re\left(\mathbf{f}_i\right) & \Im\left(\mathbf{f}_i\right) \end{array} \right]$$

of filtered image vectors $\mathbf{f}_i$ and $\mathbf{f}_j$ for each character to be a robust indicator of image discrepancies. We use

$$\kappa_{ij} = 1 - \frac{\mathbf{f}'_i \cdot \mathbf{f}'_j}{\sqrt{\mathbf{f}'_i \cdot \mathbf{f}'_i}\sqrt{\mathbf{f}'_j \cdot \mathbf{f}'_j}} \tag{4.5}$$

as a distance measure, which has range $[0, 2]$. If θ is the angle between the two vectors $\mathbf{f}'_i$ and $\mathbf{f}'_j$, the distance is related by $\theta = 1 - \arccos \kappa$. When the distance is small the characters are very similar, but when large they are dissimilar. Using the information I^S, we add the features

$$U^S_{ij}\left(y_i, y_j, \mathbf{x}; \boldsymbol{\theta}^S\right) = \delta\left(y_i, y_j\right) \boldsymbol{\theta}^S \cdot F_{ij}\left(\mathbf{x}\right), \tag{4.6}$$

where $\delta\left(\cdot, \cdot\right)$ is the Kronecker delta, and

$$F_{ij}\left(\mathbf{x}\right) = \left[\begin{array}{ccc} -\ln\left(\kappa_{ij}\right) & \ln\left(2 - \kappa_{ij}\right) & 1 \end{array} \right] \tag{4.7}$$

is a vector of basis functions that transform the distance κ_{ij} between two character images in $\mathbf{x}$. The first two functions each have a distance range boundary as an asymptote, and the last is a bias term. Thus, the first weight in the parameter vector $\boldsymbol{\theta}^S$ establishes a high compatibility reward for small distances, the second weight a low compatibility penalty for larger distances, and the bias helps (in conjunction with the first two) establish the crossover point. This is qualitatively similar to the inverse of the sigmoid function with a scaled range, except that it is no longer symmetric about the zero-crossing; see Figure 4.4. Once again, we note that the function (4.6) has the general linear form given in Section 2.1.2, but we present its more compact version here.

4.2.4 Lexicon Model

A lexicon is a useful source of high level information that can be applied to recognition, and several prior systems employing it were reviewed in Section 1.3.1. We propose another set of factors for our model that incorporates lexicon information. First, we add a set of auxiliary unknowns that will represent lexical decisions. We then add two new compatibility functions involving these and the other character unknowns. The first compatibility is simply a bias determining how likely it is *a priori* for a given string to be from the lexicon. The second is a simple binary function that connects all the constituent characters of a word with the lexical indicator. Although this compatibility function is simple in appearance, a naïve implementation would present a great deal of difficulty for common message-based approximate inference methods such as loopy belief propagation. Fortunately, one simple trick makes the implementation much easier, although it is still linear in the lexicon size. This can be problematic when the lexicon is large, therefore we introduce the use of a sparse message passing scheme for a lexical model that avoids most of the overhead required

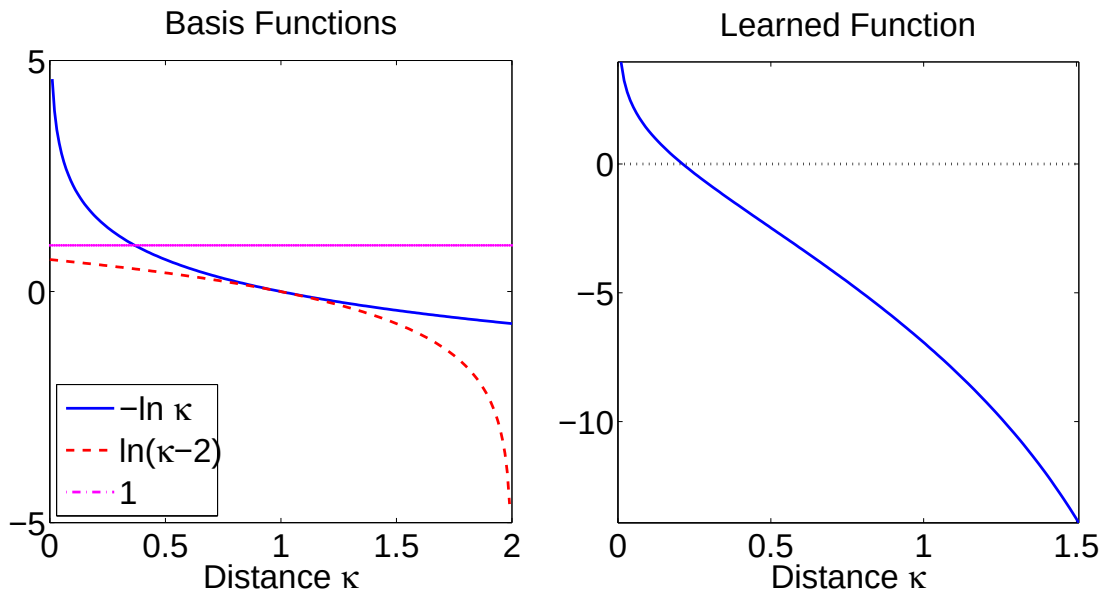


Figure 4.4. Similarity basis functions and the learned compatibility for the distance between different images of the same character; the coefficients are $\boldsymbol{\theta}^S = \begin{bmatrix} 0.9728 & 9.3191 & -6.9280 \end{bmatrix}$. The dotted line in the right-hand figure shows the crossover from reward to penalty, which occurs at a vector angle of about 37°.

with no loss of accuracy on our data. In the remainder of this section, we introduce the new lexical factors, followed by the specialized message passing scheme for inference in the resulting model.

4.2.4.1 Lexicon Factors

We add the auxiliary unknowns $\mathbf{w}$ to our model, so that each $w_k \in \{0, 1\}$ represents whether the kth string to be recognized (as a part of $\mathbf{y}$) as a word unit is drawn from a lexicon L, a subset of of all possible strings from the alphabet Y. Let C be the set of unknowns relating to a single word unit; such a set will contain the indices of some letters $\mathbf{y}$ and one entry from $\mathbf{w}$. The compatibility function relating the lexicon, the predicted string, and the lexical decision, is a simple "binary" function

$$U_C^W(\mathbf{y}_C, w_C) = \begin{cases} -\infty & w_C = 1 \wedge \mathbf{y}_C \notin L \\ 0 & \text{otherwise,} \end{cases} \tag{4.8}$$

where we have written w_C to indicate the value of the sole index of $\mathbf{w}$ present in C. Thus, the corresponding factor, or *exponentiated* compatibility (4.8), is zero only when w_C indicates the string is a word but $\mathbf{y}_C$ is not found in the lexicon. This tautological compatibility function simply represents the proposition

$$w_C = 1 \Rightarrow \mathbf{y}_C \in L$$

and would not be much use were it not for the fact the value of w_C is unknown. The indicator w_C could help control other aspects of interpretation in the model. For

instance, we might want to disable the influence of the local language compatibilities when $w_C = 1$; no matter how unlikely the word *yukky* is[1], it is in the lexicon and should not be discounted for its unusual bigrams during recognition.

The other new compatibility is a simple term biasing the preference for strings to be drawn from the lexicon

$$U_k^L\left(w_k; \theta^L\right) = (1 - w_k)\,\theta^L. \tag{4.9}$$

This function also has the general linear form as given in Section 2.1.2, but we present here its interpretable compact form, so that the single parameter θ^L can be thought of as penalty for non-lexicon predictions.

Introducing these two new classes of compatibilities U^W and U^L will be reflected by conditioning on information I^W and I^L. The factor graph for a model including these factors appears in the bottom of Figure 4.2 on page 55. In the next section, we describe more about how the two new compatibility functions (4.8) and (4.9) affect inference in the model and introduce the application of a sparse inference technique for making predictions using loopy belief propagation.

4.2.4.2 Sparse Belief Propagation

Inference, even approximate inference, in this model might be computationally taxing in general. The sum-product algorithm, briefly reviewed in Section 2.3.2, involves computing local marginals of factors, which is generally much easier than the more global marginalization desired. However, marginalizing the lexicon word compatibilities U^W grows *combinatorially* with the length of the word. This is true in general, as the complexity of the message passing equations depends roughly combinatorially on the size of the compatibility function's index set C, or rather in the size of the resulting domain Y_C: $O\left(|Y_C|\right)$. For instance, with a six letter word in our 62 character alphabet, each iteration of message passing would require a sum over 62^6 or nearly 57 *billion* strings.

Fortunately, the on/off "gating" behavior of the function U_C^W allows us to take advantage of its special form. The effect is that when $w_C = 1$, the "product" in the sum-product equation only needs to be summed over words in the lexicon. For the case when $w_C = 0$, it is summed over all strings, but the sums over constituent characters become independent. This means we can make the calculation a much more efficient product of sums. Thus, the special form (4.8) makes the inference calculation linear in the size of the lexicon or the character alphabet, rather than exponential in word size. The computational expense of a six letter word drops from billions of possible strings to just a few thousand lexicon words.

As a concrete example, let us assume that the set index C corresponds to the nodes of the second, three-letter word in the bottom factor graph of Figure 4.2 on page 55. For clarity, we will use numerical indices for the character nodes $\mathbf{y}$ and alphabetical indices for the word nodes $\mathbf{w}$. In this example, w_C is the generic reference to the word

[1]Under a bigram model trained on a corpus of English text, the word is actually the least likely in a large lexicon.

node for the factor C, while w_{B} is the particular node. The general form of a factor-to-node message (cf. Eq. 2.34 on page 29) is a product of the factor times the messages to that factor from all its arguments except the message recipient. This product is then summed over all those arguments leaving a function whose value is dependent on the recipient node. For the lexicon factor we are calling C, the specialized form of the message from C to the character y_6 has the form

$$\begin{aligned} m_{C\to 6}(y_6) &= \sum_{\substack{\mathbf{y}_{C\setminus\{6\}} \in Y_{C\setminus\{6\}},\\ w_{\mathrm{B}} \in \{0,1\}}} \Big[\exp U_C^W(\mathbf{y}_C, w_{\mathrm{B}}) * \\ &\qquad\qquad m_{7\to C}(y_7)\, m_{8\to C}(y_8)\, m_{\mathrm{B}\to C}(w_{\mathrm{B}})\Big] \quad (4.10) \\ &= \sum_{\{\mathbf{y}_C \in L : y_6\}} m_{7\to C}(y_7)\, m_{8\to C}(y_8)\, m_{\mathrm{B}\to C}(w_{\mathrm{B}}=1) + \\ &\quad \sum_{y_7\in Y, y_8\in Y} m_{7\to C}(y_7)\, m_{8\to C}(y_8)\, m_{\mathrm{B}\to C}(w_{\mathrm{B}}=0) \quad (4.11) \\ &= \sum_{\{\mathbf{y}_C \in L : y_6\}} m_{7\to C}(y_7)\, m_{8\to C}(y_8)\, m_{\mathrm{B}\to C}(w_{\mathrm{B}}=1) + \\ &\quad m_{\mathrm{B}\to C}(w_{\mathrm{B}}=0) \quad (4.12) \end{aligned}$$

We first separate the sums for the two values of w_{B}. Because the factor $\exp U_C^W(\mathbf{y}_C, w_C)$ is zero whenever the argument $\mathbf{y}_C$ is not in the lexicon but $w_C = 1$, the sum can be restricted from $Y_{C\setminus\{6\}}$ to the portion of the lexicon that agrees with the argument value y_6 when $w_{\mathrm{B}} = 1$. Furthermore, when $w_{\mathrm{B}} = 0$, the factor is always one. In the last line, we may push the sums over each character value y_7 and y_8 in against the corresponding messages. Because these messages are normalized to sum to one in practice, these terms are dropped, leaving us with a relatively simple sum over a subset of lexicon terms. To calculate the message to character 6 for all values of y_6 involves a sum over all lexicon words of the appropriate length. Messages to other character nodes will have the same form, with the number of node-to-factor messages in the product depending on the length of the word.

The message to the word node w_{B} undergoes a similar transformation:

$$\begin{aligned} m_{C\to \mathrm{B}}(w_{\mathrm{B}}) &= \sum_{\mathbf{y}_C \in Y_C} \exp U_C^W(\mathbf{y}_C, w_{\mathrm{B}})\, m_{6\to C}(y_6)\, m_{7\to C}(y_7)\, m_{\mathrm{B}\to C}(y_8) \quad (4.13) \\ &= \begin{cases} \sum_{\mathbf{y}_C \in L} m_{6\to C}(y_6)\, m_{7\to C}(y_7)\, m_{8\to C}(y_8) & w_{\mathrm{B}} = 1 \\ 1 & w_{\mathrm{B}} = 0 \end{cases}. \quad (4.14) \end{aligned}$$

As before, the term for $w_{\mathrm{B}} = 1$ only needs to be summed over lexicon words of the appropriate length. When $w_{\mathrm{B}} = 0$, sums over individual characters are again pushed against their messages and simplify to their normalized sum of one, leaving only the product of unity.

Although these sums have an upper bound complexity linear in the size of the lexicon, it still presents a computational drag in practice. The top-down information

is very important for accurate recognition, so we use a bottom-up scheme to speed the recognition process. Pal et al. [89] propose a probabilistically motivated sparse inference method that simplifies the message passing calculations. The central idea is to reduce the number of summands in the factor-to-node messages $m_{C\to i}(y_i)$, like (4.12) and (4.14), by creating zeros in the node-to-factor messages $m_{i\to C}(y_i)$.

At each iteration in the loopy version of belief propagation, a belief state, or local approximate marginal probability, at every node is represented by the normalized product of messages to that node from its adjacent factors (cf. Eq. (2.35) on page 29). Each factor combines information from its adjacent node arguments and returns updates to them. As described above, the update for the lexicon factor involves a sum over every word in the lexicon (of the appropriate length), even those words containing characters with low probabilities. We may therefore desire to eliminate these unlikely lexicon words from consideration during the belief update stage. The well-motivated approach given by Pal et al. is to revise the local beliefs such that the largest number of the lowest probability states have zero probability, subject to a constraint on the divergence of the sparse belief from the original. In other words, consider the fewest number of possibilities while sticking close to your original beliefs. Employing this strategy, we expect to greatly reduce the amount of lexicon scans for a given query.

If b_i represents the marginal belief for node i in the graph, our goal is to compress this distribution to b'_i such that it has the maximum number of zero entries, subject to a divergence constraint:

$$\begin{array}{rc} \text{maximize} & \sum_{y_i \in Y_i} \delta\left(b'_i(y_i), 0\right) \\ \text{subject to} & \mathrm{KL}\left(b'_i \parallel b_i\right) \leq \epsilon, \end{array} \tag{4.15}$$

where

$$\mathrm{KL}\left(b'_i \parallel b_i\right) = \sum_{y_i \in Y_i} b'_i(y_i) \log \frac{b'_i(y_i)}{b_i(y_i)} \tag{4.16}$$

is the Kullback-Liebler divergence between the original and compressed beliefs. This can easily be accomplished for each node in time $O\left(|Y_i| \log |Y_i|\right)$ by sorting the beliefs and calculating the log cumulant. Once the sparse belief b'_i is calculated, the messages to the factors $m_{i\to C}$ (cf. Eq. (2.33) on page 28) are compressed to respect the sparsity of b'_i and re-normalized.

These sparse node-to-factor messages are then subsequently used to calculate the reverse factor-to-node messages. The practical effect of sparse belief propagation is that certain characters are eliminated from consideration. For instance, the visual and contextual evidence for y_7 to be a `t` may be so low that it can be assigned a zero belief without greatly changing the current local marginal. When this happens, we may eliminate summands for any word whose second character is `t` in the messages (4.12) and (4.14). Taken together, pruning highly unlikely characters reduces the lexicon under consideration from tens of thousands of words to just a few, dramatically accelerating message passing-based inference.

We should note that depending on the order of operations, characters may not be strictly eliminated from possibility. If outgoing messages to factors are made sparse in agreement with the compressed disribution b'_i, this only means that terms are dropped

Figure 4.5. Examples of sign evaluation data. LEFT: Regular font similar to those often found in documents. CENTER: Unusual or custom regular font (all repeated characters are the same). RIGHT: Custom irregular font where repeated characters have a different appearance.

from the summation used to calculate messages to other nodes. The return message may not be sparse at all, and in general it is not. Thus, using sparse methods to "eliminate" characters means only that we lose the influence of the dropped character hypotheses upon their logically dependent nodes. The final belief at a node (from which predictions are made) is calculated using the most recent incoming messages from the factors, which are not generally sparse. Therefore we have not necessarily committed to a mistaken elimination of correct character hypotheses.

4.3 Experiments

In this section we present experimental validation of our model on sign images containing novel fonts and non-lexicon words. The alphabet of characters we will recognize, Y, consists of 26 lowercase, 26 uppercase, and the 10 digit characters (62 total).

First we describe the data used in our experiments for both training and testing, and then the procedures used to train and evaluate the models. The section concludes with the experimental results and a subsequent analysis and discussion.

4.3.1 Experimental Data

Because we have such a rich model involving many information sources, there are many corresponding data sets for training, including character images, English text corpora, and a lexicon. We describe these after detailing the nature of the primary evaluation data.

Sign Evaluation Data Our evaluation data comes from pictures of signs captured around downtown Amherst, MA in September and October 2002. There are 95 text regions (areas with the same font) totaling 215 words with 1,209 characters. Many signs have regular fonts (that is, characters appear the same in all instances) that are straightforward, such as basic sans serif, and should be easily recognized. Other signs contain regular fonts that are custom or rarely seen in the course of typical document recognition. Finally, there are a few signs with custom irregular fonts that pose the greatest challenge to the premise that similarity information is useful. Examples of each of these three categories are shown in Figure . The signs are imaged without extreme perspective distortion—they are roughly fronto-parallel. Following

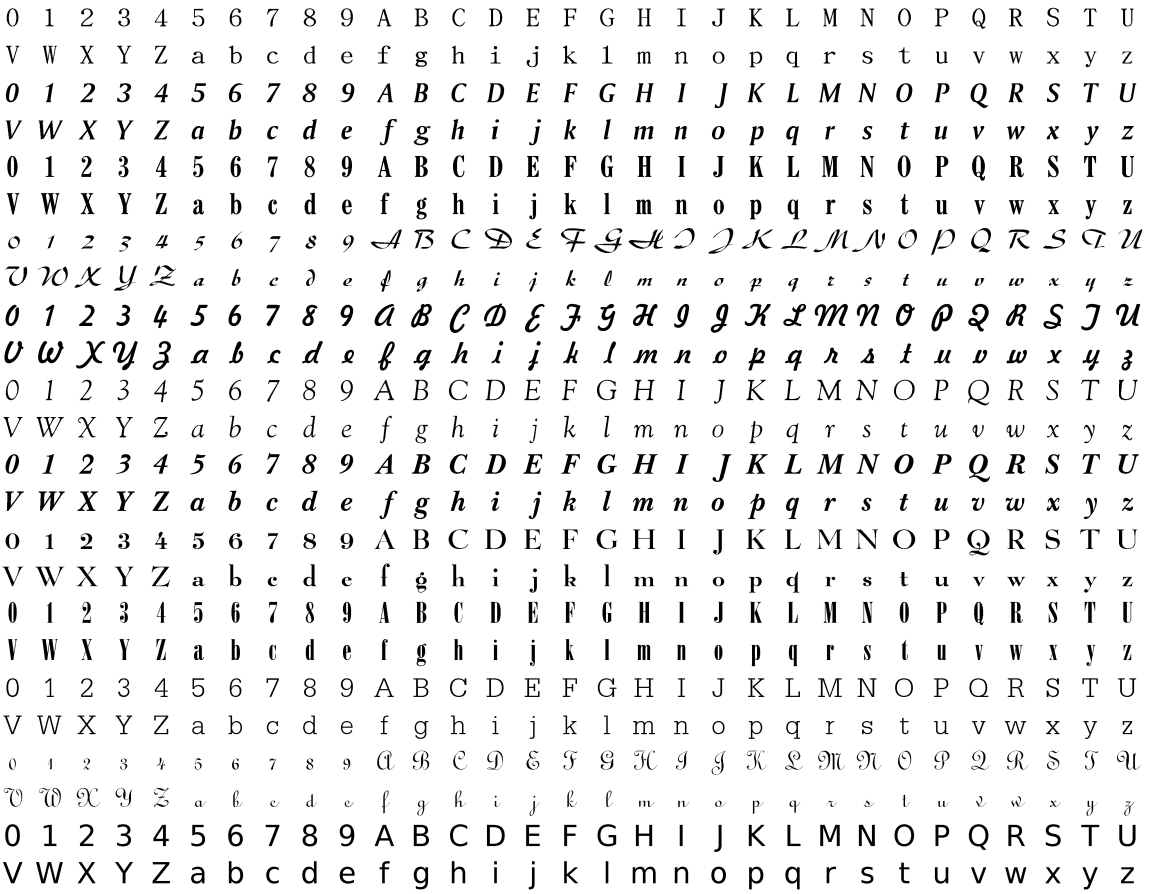


Figure 4.6. Example fonts from synthetic training data.

the assumptions laid out in Section 4.2, we have annotated our evaluation data with the approximate bounding boxes for the characters.

Synthetic Font Training Data We generated images of each character in several commercially available fonts using GIMP.[2] Each image is 128×128 pixels with a font height of 100 pixels (an x-height of roughly 50 pixels). No anti-aliasing was used in the image synthesis and the bounding box of the character is centered in the window. Examples are shown in Figure 4.6.

Text Corpora A corpus of English text was acquired from Project Gutenberg[3]—82 books including novels, non-fiction, and reference for a total of more than 11 million words and 49 million characters (from our 62 character alphabet).

[2]GNU Image Manipulation Program `http://www.gimp.org`.

[3]`http://www.gutenberg.org`.

Lexicon The lexicon we use is derived from SCOWL[4] and contains 187,000 words including proper names, abbreviations, and contractions. Because our current model does not account for word frequency in its lexical bias, we only use those words in the list up to the 70th percentile of frequency for our lexicon.

4.3.2 Experimental Procedure

In this section we describe the procedure used for training and evaluating our model. We first outline the nature of the overall model parameter training followed by details of training for each component of the model. The section concludes with a brief description of how the model is applied to the actual image data for evaluation.

4.3.2.1 Model Training

The model parameters $\boldsymbol{\theta} = \left[\begin{array}{ccccc} \boldsymbol{\theta}^A & \boldsymbol{\theta}^B & \boldsymbol{\theta}^C & \boldsymbol{\theta}^S & \theta^L \end{array} \right]$ are learned from the data outlined above. Typical parameter estimation procedures in such discriminative joint models requires labeled data that involves all the information at once. In other words, training data should be like the testing data. For example, training character images must be used within a stream of actual English text to simultaneously learn parameters for the appearance discriminant U^A and the bigram model U^B. To eliminate this requirement, we approximate the model likelihood using the parameter decoupling scheme described in Section 2.2.1.1. The parameters $\boldsymbol{\theta}^A$, $\boldsymbol{\theta}^B$, $\boldsymbol{\theta}^C$, and $\boldsymbol{\theta}^S$ are each learned independently in this fashion, while the lexicon bias parameter θ^L is chosen by cross-validation. Next, we detail the training procedures for each of these parameter sets.

Appearance Model The character image appearance model parameters $\boldsymbol{\theta}^A$ are trained on 200 fonts, and 800 fonts are used as a validation set. The value of hyperparameter α for the Laplacian prior (cf. Eq. (2.27) on page 26) that yields the highest likelihood on the validation set is the one used for optimizing the posterior for $\boldsymbol{\theta}^A$.

The filter outputs for the 128×128 training images are downsized by a factor of four to 32×32 to get $\mathbf{d}_i$. Although some information from the highest frequency filters is lost, this reduces the dimensionality of $\boldsymbol{\theta}^A$ by a factor of 16.

The evaluation data is far from having perfect contrast (a nearly 0/1 binary image). As a very simple alternative to a more elaborate contrast normalization scheme, we scale the training images so that the contrast (absolute difference between background and character intensity) is 0.3. The contrast of the characters in the evaluation data is shown in Figure 4.7.

Language Model To avoid the need for performing inference on large chains of text, we use the piecewise training method described in Section 2.2.1.2 to approximate the (already decoupled) likelihood. This is especially advantageous for the bigram and

[4] `http://wordlist.sourceforge.net`.

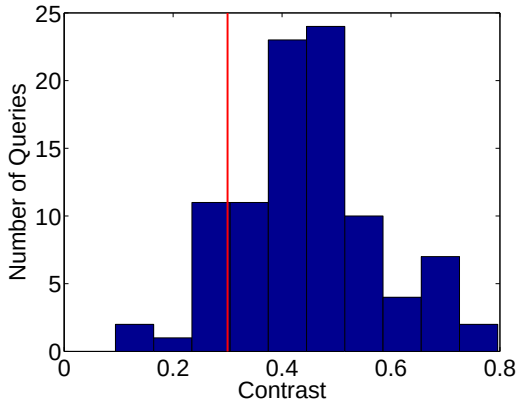


Figure 4.7. Histogram of evaluation data character contrast. The vertical red line indicates the contrast given to training images.

case switch models (4.2), (4.3), and (4.4), which do not depend on an observed image. Thus, training instances may be collapsed into unique cases and weighted by their frequency. For example, the corpus of 49 million characters contains nearly 780,000 occurrences of the bigram `th`. Rather than doing inference on the entire chain of text with an exact method, we need only do inference once in a two-node chain for `th` and count it 780,000 times.

The books were split into two sets of roughly equal size, one for training and one for validation. The (case-insensitive) bigram counts were taken for each set, and the value of the hyperparameter α for the Laplacian prior (cf. Eq. (2.27) on page 26) that yields the highest likelihood on the validation set is the one used for optimizing the posteriors for $\boldsymbol{\theta}^B$ on the entire corpus.

In practice, we found that enabling the language model, regardless of the value of the auxiliary word indicators $\mathbf{w}$ improved accuracy over disabling it whenever the corresponding $w_k = 1$. Our results reflect this aspect of the model.

A uniform prior was used for the case model parameters $\boldsymbol{\theta}^C$.

Similarity Model Because the function U^S is zero whenever its two character arguments have different labels and otherwise has a functional value parameterized by $\boldsymbol{\theta}^S$ (displayed in Figure 4.4 on page 59), we may equivalently learn the parameters for a function U^S that takes only a single argument y with a label of Same or Different. The piecewise training approximation described in Section 2.2.1.2 follows naturally because these character pairs are completely decoupled from any related stream of text.

To learn the similarity parameters $\boldsymbol{\theta}^S$ we generated pairs of the same character (in the same font) and pairs of different characters (also in the same font) with the following procedure. First, we select a font and a character uniformly at random. To produce a similar character, we generate a random linear transformation

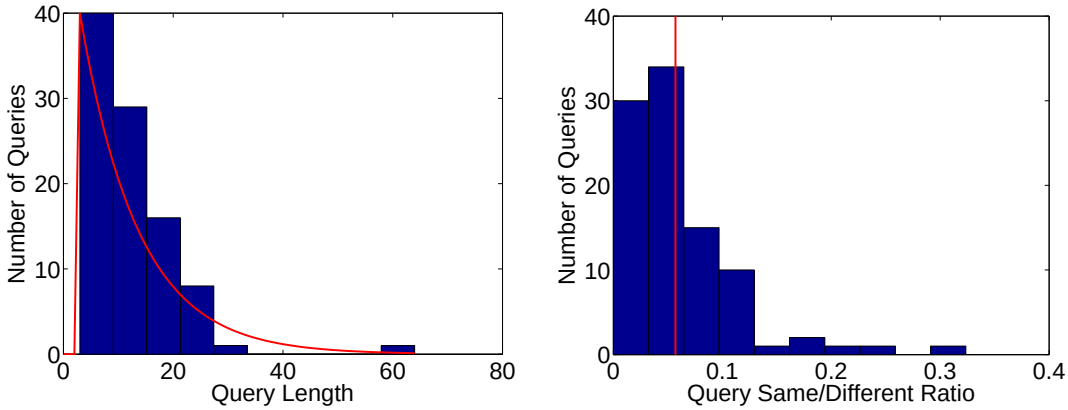


Figure 4.8. Comparison of training and evaluation data for similarity model. LEFT: Histogram of query lengths (number of characters in a given instance of text). The red curve is the (scaled) geometric distribution used for sampling excerpts from the training corpus. RIGHT: Histogram of query same/different character ratio. The red line is the overall ratio acquired from the corpus sample.

$$T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \sigma_x & \rho_x \\ \rho_y & \sigma_y \end{bmatrix} \tag{4.17}$$

with rotation $\theta \sim \mathcal{N}(0, 1^\circ)$, scale factors $\sigma_x, \sigma_y \sim \mathcal{N}(1, 0.01)$, and skew factors $\rho_x, \rho_y \sim \mathcal{N}(0, 0.005)$. This transformation is then applied to the original image. To produce a dissimilar pair, a different character is chosen uniformly at random. We choose a different character from the same font because this reflects our problem, having assumed the input is from a single font. Furthermore, these are likely to be more similar than different characters from different fonts, allowing a better and more appropriate threshold to be learned. Additive Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.01)$ is added to the original and transformed images prior to Gabor filtering. Unlike for the appearance model, the full-size (128×128) filter outputs are used to calculate the distance κ_{ij} between images. The finer details are useful for these comparisons, and the dimensionality is not an issue since U^S only has three parameters.

For optimal predictive discrimination, the ratio of same to different pairs in the training data should be the ratio we expect in testing data. Toward this end, we sample small windows of text from our corpus. The window length is sampled from a geometric distribution with a mean of 10 characters and length at least 3; these parameters are chosen based on our prior expectation of sign contents. In 10,000 samples, the same/different ratio is consistently about 0.057. This ratio controls the relative number of similar and dissimilar pairs we generated (100,000 total). Figure 4.8 compares these model parameters and sampling methods with the evaluation data.

A uniform prior was used for the similarity model parameters $\boldsymbol{\theta}^S$.

Lexicon Model

Parameter Learning We found acceptable performance for models conditioned on I^A, I^B, I^C, I^S with corresponding parameters found in the decoupled fashion as detailed above. One way of doing this for the bias parameter θ^L is to use our English corpus, using each word from the text as a training instance with $w = 1$ if it is in our lexicon L and $w = 0$ otherwise. We found that decoupling the learning of θ^L in this way does not yield a strong enough lexical bias to improve results as originally hoped, so we turn to a cross-validation strategy to "re-couple" the parameter learning.

To add I^L to the model, we keep all other parameters fixed at their values learned from decoupling. The 95 regions in the evaluation sign data is randomly split into ten subsets. In a ten-fold cross-validation procedure, we iteratively held out one set for testing. Several values of θ^L are used, and the one with the highest word accuracy on the nine training sets is then applied to the test set for evaluation.

We can also force the model to always predict words from the lexicon by adjusting the bias θ^L to $-\infty$. We will use $I^L_{-\infty}$ to indicate such a closed-vocabulary assumption.

Sparse Belief Propagation Sparse message passing as proposed by Pal et al. [89] was intended for belief propagation in a chain-structured graph where a well-defined forward-backward schedule for message passing achieves exact inference. While the graph based on I^A, I^B, I^C is a chain, adding the lexical information I^W makes this graph not only not chain-structured, but cyclic. Thus, the results of belief propagation will not be exact in general. It is only I^W that is truly problematic from a computational standpoint. The other messages—of which there are only a few—only require complexity of at most $O(Y^2)$, which is substantially less than the messages from the lexical factor. For this reason, we run belief propagation in a phased schedule, only sending any lexical factor to node messages after the others have converged. Once these messages have converged, we have the best possible local marginals on the available information, excepting I^W and I^L. We then use these beliefs for computing the sparsity of the character states $\mathbf{y}$. This sparsity is calculated once, then the lexical information I^W, I^L is introduced, and the same sparsity structure is maintained. Belief propagation then continues until the termination criterion is reached (convergence or an iteration limit).

This phased processing has two advantages. First, because messages are passed within a limited portion of the model until convergence, the beliefs used to calculate sparsity should be more reliable since the available information has flowed throughout the graph. This stands in contrast with the alternative of doing state pruning with the initial beliefs, which will only be based on factors immediately adjacent to the nodes. Longer distance dependencies certainly exist in these types of models, and these could have an effect on the sparsity and correctness of the approximate beliefs. The second and arguably more important advantage is that it avoids the need to make a complete lexical scan required in the messages from the lexical factor. Since the messages are initialized to uniform, the lexical factor merely ends up contributing positional unigrams to the initial belief. This is not worth the cost of the lexical scan

and could be modeled directly if we wished. Returning to our initial point, we prefer to use the best available information before incorporating the lexicon.

We use $\epsilon = 10^{-7}$ as the divergence bound for sparse belief propagation. This corresponds to keeping nearly all of the probability mass ($e^{-\epsilon}$) for each character. The runtime was sensitive to this, since it controls the amount of pruning, but we found accuracy was not.

4.3.2.2 Model Application

Here we add a few additional details of how the evaluation images are processed for the model. The height of the input characters in the evaluation data is normalized so that the font size is roughly that of the appearance training data. Only filter responses from within the annotated bounding box of each character are used when calculating the energies for appearance U^A and similarity U^S; indices in $\mathbf{f}_i$ that are outside the bounding box are zeroed out. Note that Gabor filters are applied to the actual grayscale image; no binarization is performed.

4.3.3 Experimental Results

Here we describe the performance of several variants of our model on the evaluation data, as well as alternatives from prior approaches to challenging character recognition problems. First, we demonstrate the impact of using similarity information in a unified model for recognition. Then we investigate how incorporating a lexicon affects results.

4.3.3.1 Unified Similarity

Prior work using similarity (reviewed in Section 1.3.3.2) incorporated this information in a processing stage separate from that using character appearance. Here we will compare our unified model to the approach of Breuel [15, 16], where characters are first clustered using the degree of similarity as a distance metric. Following this approach, to cluster letters, we maximize $p\left(\mathbf{y} \mid \mathbf{x}, \widehat{\boldsymbol{\theta}}, I, I^S\right)$ for $\mathbf{y}$ via simulated annealing, initialized at the prediction derived from I^A (the strategy taken by Breuel [15]). The identity of each cluster is then chosen by using the classification of each character derived from other models (sans I^S) as a vote. Ties are broken by choosing the label whose members have the lowest average entropy for their posterior marginal, a strategy that slightly outperforms random tie breaking.

Table 4.1 gives the results of the unified model using different combinations of appearance information I^A, language information I^B, I^C, and similarity information I^S. Table 4.2 shows the results when the similarity information is used first to cluster the characters, and the other information (used separately) is then used to vote on character identities. Character accuracy is the percentage of characters correctly identified (including case). To evaluate the ability of our model to recognize different instances of the same character in the same font, for intra-sign and intra-font characters we measure:

Table 4.1. Recognition results (percentages) of the unified model with varying amounts of information. Overall character accuracy as well as the false negative (FNR), false positive (FPR), and hit rates (HR) for pairs (see text) are given.

Information	Character Accuracy	FNR	FPR	HR
I^A	84.04	11.42	0.51	91.07
I^A, I^S	84.04	11.42	0.51	91.07
I^A, I^B	87.92	9.14	0.53	93.81
I^A, I^C	87.92	8.79	0.87	94.03
I^A, I^B, I^C	91.65	6.85	0.66	98.68
I^A, I^B, I^C, I^S	93.22	5.45	0.14	99.26

Table 4.2. Recognition results (percentages) of clustering followed by recognition and voting. Overall character accuracy as well as the false negative (FNR), false positive (FPR), and hit rate (HR) for pairs (see text) are given.

Information	Accuracy	FNR	FPR	HR
I^S	-	22.67	0.25	-
$I^S \rightarrow I^A$	83.54	7.03	0.69	88.28
$I^S \rightarrow I^A, I^B$	87.92	4.39	0.80	91.73
$I^S \rightarrow I^A, I^C$	87.76	5.80	1.02	92.72
$I^S \rightarrow I^A, I^B, I^C$	91.40	3.69	0.88	97.26

False negative rate: Percentage of character pairs that are the same but are given different labels.

False positive rate: Percentage of character pairs that are different but are given the same label.

Hit rate: Percentage of character pairs that are the same, given the same label, and correct (correctly labeled true positives).

Figure 4.9 contains examples of signs correctly read, as well as examples from the evaluation set that are more difficult.

Another interesting comparison is provided by the likelihood ratio of the data under models with different amounts of information. Let $\mathcal{D} = \left\{\mathbf{y}^{(k)}, \mathbf{x}^{(k)}\right\}_k$ represent our evaluation label and image data. The geometric mean of the likelihood ratio between the language-informed and appearance-only models is

$$\left[\prod_{k=1}^{N} \frac{p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \widehat{\boldsymbol{\theta}}, I, I^A, I^B, I^C\right)}{p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \widehat{\boldsymbol{\theta}}, I, I^A\right)}\right]^{\frac{1}{N}} \approx 85.33. \tag{4.18}$$

Adding the similarity information to the model also yields a modest increase in belief about the correct labels for the data:

Figure 4.9. Examples from the sign evaluation data. LEFT: Signs read correctly with I^A, I^B, I^C, I^S. RIGHT: Challenging signs that have unique fonts, are hand-painted, or contain three-dimensional effects, real and virtual.

$$\left[\prod_{k=1}^{N} \frac{p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \widehat{\boldsymbol{\theta}}, I, I^A, I^B, I^C, I^S\right)}{p\left(\mathbf{y}^{(k)} \mid \mathbf{x}^{(k)}, \widehat{\boldsymbol{\theta}}, I, I^A, I^B, I^C\right)}\right]^{\frac{1}{N}} \approx 1.02. \tag{4.19}$$

4.3.3.2 Lexicon-Based Model

In addition to the unified similarity model, we also test the effect of the integrated lexicon and the impact of using sparse belief propagation. Table 4.3 compares the character and word accuracy for our model with varying amounts of information. For comparison with some methods reviewed in Sections 1.3.1 and 1.3.2, the output of our best lexicon-free model is passed through the spell-checker Aspell, keeping the top suggestion. Figure 4.10 shows results on example data of varying difficulty, including where corrections were made and errors introduced.

We show in Figure 4.11 the histogram of how many characters remain possible after belief compression with sparse belief propagation for several of the models. The elimination of many characters from consideration excludes certain words in the lexicon with characters in particular positions. The resulting reductions in length-appropriate lexicon words is shown in the histograms of Figure 4.12. Different word lengths have differing numbers of possible words in the lexicon, so we give length-specific lexicon size-normalized values. However, to illustrate the raw impact we also give the median absolute size of the resulting lexicon.

Table 4.3. Word and character accuracy with various forms of the model.

Information	Character Accuracy	Word Accuracy
I^A	84.04	46.05
I^A, I^B, I^C	91.65	75.35
I^A, I^B, I^C, I^S	93.22	78.60
I^A, I^L, I^W	93.63	72.56
$I^A, I^L_{-\infty}, I^W$	91.56	68.84
I^A, I^B, I^C, I^L, I^W	93.88	**85.58**
$I^A, I^B, I^C, I^S, I^L, I^W$	**94.46**	**85.58**
$I^A, I^B, I^C, I^L_{-\infty}, I^W$	92.39	81.40
$I^A \rightarrow$ `Aspell`	73.78	53.49
$I^A, I^B, I^C \rightarrow$ `Aspell`	89.50	77.21
$I^A, I^B, I^C, I^S \rightarrow$ `Aspell`	90.98	79.07

4.3.4 Discussion

4.3.4.1 Similarity Model

Figure 4.9 on the preceding page contains examples of signs correctly read without the lexicon, showing that the features are robust to various fonts and background textures (e.g., wood and brick). Although the number of characters per sign is small compared to OCR applications, adding similarity information undoubtedly improves character recognition accuracy, reducing overall error by nearly 20% (Table 4.1). Not surprisingly, most of this improvement comes from greatly reducing the cases when different characters are given the same label (pair false positives).

Perhaps surprisingly, adding similarity information I^S to the simple image information I^A does not alter the results. This is probably because test images have relatively little noise and are mostly difficult due to font novelty and non-fronto-parallel orientations. Therefore, it is expected that the same characters, though novel, would often be given the same label in different locations, due to their logical independence solely with information I^A. However, when other sources of information are introduced to help resolve ambiguity, the similarity information does make a difference because the bigram and case information are based on local context. These can push the beliefs about characters in different directions, even though they tend to look the same, because their contexts are different. Adding the similarity information on top of these other sources ensures that the local context does not introduce a contradictory bias, as was demonstrated in Figure 4.1 on page 52. Adding bigram information pushes the second `e` to an `a` because preference for the `ea` bigram outweighs both `ee` and the character/image energy. Similarly, adding case information pushes the `l` from being recognized as the upper case `I` to lower case `t`; due to kerning in this italic font, some of the `F` overlaps in the `l`'s bounding box, leaving a little crossbar indicative of a `t`. Finally, adding the similarity information corrects the `l` since it is very different from the final `t`, and corrects the `e`s since they are very similar.

Model			
Correct	Free checking	31 BOLTWOOD	DELANOS
No Lexicon	Free crecking	31 BOLTWOOD	RELAmo3
Lexicon	Free checking	31 BOLTWOOD	RELAmo3
Forced Lexicon	Free creaking	SI BENTWOOD	DELANOS
Aspell	Frag recurred	31 BELLWOOD	Reclaim

Model			
Correct	HOOK UPS	MYSTERY TRAIN	REVOLVE
No Lexicon	HTOR UP5	MYOUGRP ndaIN	FERTEVE
Lexicon	HOOK UPS	MNLUGRP RMRIN	REREERE
Forced Lexicon	HOOK UPS	LATHERY ANTIN	RESTORE
Aspell	THOR UPI	MARGERY AARON	RETRIER

Figure 4.10. Example recognition results on difficult data. Correct words indicated in bold. Model examples are I^A, I^B, I^C, I^S (No Lexicon), $I^A, I^B, I^C, I^S, I^L, I^W$ (Lexicon), $I^A, I^B, I^C, I^S, I^L_{-\infty}, I^W$ (Forced Lexicon) and $I^A, I^B, I^C, I^S \rightarrow$ `Aspell` (Aspell).

All of the differences in accuracy for the unified model (Table 4.1) are statistically significant.[5] In particular, adding the similarity information I^S to I^A, I^B, I^C reduces character classification error by 19%. While the reduction of false negatives is not significant with the addition of I^S, the false positives are cut by 79%. When the unified model is compared to the pipelined clustering approach, the differences between I^A, I^B, I^C, I^S and $I^S \rightarrow I^A, I^B, I^C$ are significant for character accuracy, false negative rate, and false positive rate.

As expected, adding more prior information to the model boosts the likelihood of the data. The model using appearance alone is relatively weak, since a probability has an upper bound of one, yet the ratio in (4.18) is quite large. In addition to improving the prediction accuracy, adding the similarity information yields an increase in the degree of belief for the correct labels, as shown by (4.19). Although the increase is slight on average, more than ten percent of the signs in our evaluation data exhibit an increase of at least one order of magnitude. This could be important when confidence in the model's prediction helps to determine how to handle a query.

The results of clustering the letters prior to recognition appear worse than doing recognition outright with no similarity information. However, unifying all the information available—including similarity— does yield better results than a distinct clustering step. It is interesting that clustering yields fewer false negatives than the unified approach. This is most likely because clusters are not forced to have different labels at the secondary assignment stage. Thus, instances of the same character assigned to different clusters are not forced to have different labels (up to the fact that

[5] In all cases, significance is assessed by a paired, two-sided sign test on the accuracy per query.

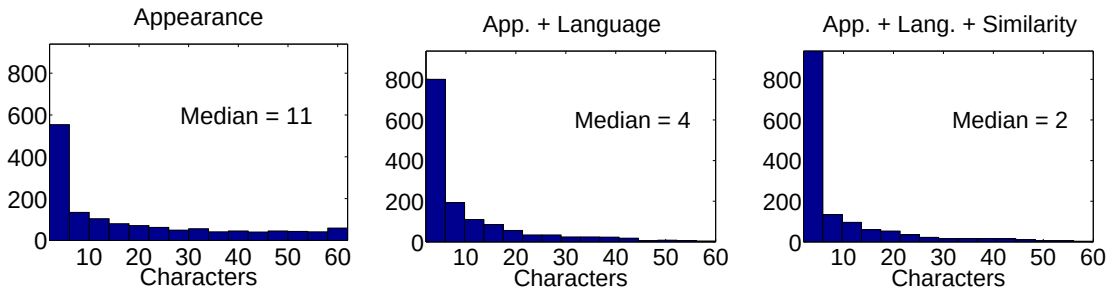


Figure 4.11. Histograms of character state space cardinality after belief compression. LEFT: Appearance only model I^A, I^L, I^W. CENTER: Appearance and language model I^A, I^B, I^C, I^L, I^W. RIGHT: Full model with appearance, language, and similarity $I^A, I^B, I^C, I^S, I^L, I^W$.

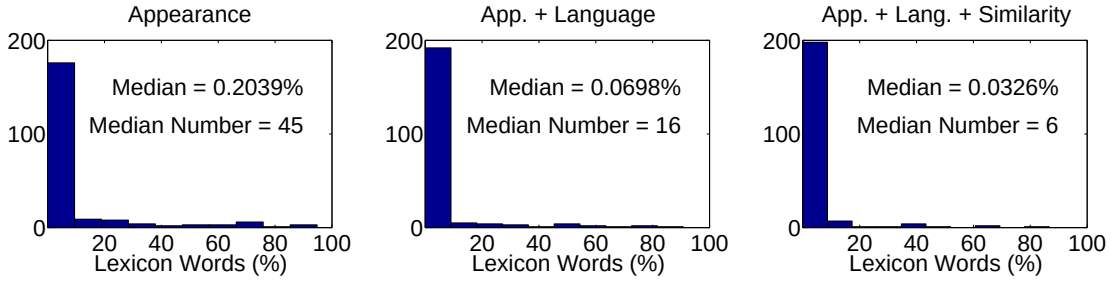


Figure 4.12. Histograms of the percentage of length-appropriate lexicon words considered after belief compression. LEFT: Appearance only model I^A, I^L, I^W. CENTER: Appearance and language model I^A, I^B, I^C, I^L, I^W. RIGHT: Full model with appearance, language, and similarity $I^A, I^B, I^C, I^S, I^L, I^W$.

there are only as many clusters as characters in our alphabet Y). Indeed, if this *were* the case, the false negative rate would be intolerably high. Conversely, the clustering pre-processing step does commit unrecoverable errors by pairing characters that are not the same; subsequent information cannot reduce the false positive rate. This is especially critical because the probability of two characters being the same *a priori* is much smaller than their being different, thus the false positive rate has a greater impact on total errors than the false negative rate.

Some signs in our data set present tremendous difficulty and challenge the assumption that characters of the same "font" appear similar. Some of these are due to rendered warping effects, custom fonts, or inconsistent shadow effects (see Figure 4.9 on page 71). Other signs just have unique fonts that are very different from those in the training set.

4.3.4.2 Lexicon Model

Here we discuss the results of adding the lexicon, some of which are shown in Figure 4.10 on page 73. `31` and `BOLTWOOD` are not in the lexicon, so errors arise with the forced lexicon and Aspell models. `DELANOS` is in the lexicon, but the image evidence overpowers the bias in this case; forced to be a lexicon word, it is correctly interpreted. The last two images exemplify some of the more difficult text in our data set.

Incorporating the lexicon factor boosts the character accuracy, but adding the language model (i.e., bigrams) after the lexicon seems to have little impact. However, the word accuracy reveals a 41.5% error reduction with the inclusion of the lexicon. Results do improve over an appearance-only model when words are forced to be from the lexicon, but some proper nouns and numbers in the data are not lexicon words and thus are misinterpreted. Using Aspell fixes some simple errors, but most errors are more complex. Ignoring the character image for poorly recognized words tends to reduce overall character accuracy (since poor suggestions are made). We also experimented with trigrams and word frequencies (i.e., using a word-specific value for U^W) , but found no improvement in word accuracy on our evaluation data.

Sparse belief propagation speeds the lexicon integration by eliminating characters from consideration after belief compression (Figure 4.11). This results in a 99% reduction of candidate lexicon words overall. We must consider different lexicon words for strings of different lengths. The median elimination of candidate words for each string was 99.97% (Figure 4.12), or just 6 remaining candidates when not normalized for the differing sizes of the original candidate lists. Adding language information makes character beliefs more certain, allowing more characters and lexicon words to be pruned.

4.4 Contributions

We have laid out a general framework for recognition that is probabilistically well-motivated and can accept long range information in a unified fashion. The conceptual advantage provided by discriminative Markov models easily allows one to imagine and implement a relationship among the unknowns.

Our principle contributions are as follows. First, we have constructed a model that allows unified processing of several important pieces of information (appearance, language, similarity to other characters, and a lexicon). Second, we show how a similarity function can be learned and integrated so that recognition is improved and more consistent with small samples of novel fonts. Finally we have proposed a simple construction that incorporates a lexicon into the model and facilitated its use by applying principled sparse methods.

The basic discriminative framework for character recognition is not new, but it has typically been relegated to individual characters. As outlined in Sections 1.3.1 and 1.3.2, language information is usually employed after recognition in a post-processing clean-up. The models that have integrated language with recognition are *generative*, which has two drawbacks. First, generative models are typically outperformed by

their discriminative counterparts, since the latter merely focus on the recognition task rather than the (often more complex and sometimes irrelevant) data modeling task. Second, the independence assumptions of generative models often prohibit them from using richer features of the data (observations). A recent exception is the work of Jacobs et al. [52], whose discriminative model forces recognition output to be lexicon words. In contrast, our lexicographic model allows a smooth trade-off between the interpretation of a string as a known word, or some other string.

Section 1.3.3 highlights that classifier adaptation is a useful strategy for recognition. However, when recognizing signs or scene text, there is a scant amount of data, and it is generally insufficient for reliable use with the existing methods for coping with novel typefaces. Our recognition strategy improves on two issues lacking in previous approaches. First, by simultaneously incorporating character identity and similarity information into a unified probabilistic model, we eliminate the need for distinct clustering/recognition steps and the potential for unrecoverable errors. Second, we treat similarity and dissimilarity as two sides of the same issue, which prevents dissimilar characters from being given the same label.

It has long been known that the use of a lexicon can improve recognition accuracy. Although some computational tricks exist, the size of a lexicon can often be prohibitive for processing that integrates recognition, rather than using it as a post-processing step. Our model provides a natural, practical testbed for the sparse inference methods proposed by Pal et al. [89] for acyclic models. This has the advantage over the traditional approach, which is to prune to one possibility for higher-level processing or use a more ad hoc method to consider a reduced number of alternatives. By eliminating characters from outgoing messages in a principled fashion, we are able to drastically reduce the size of the lexicon that is used for a given query. This does not necessarily mean that characters are eliminated from possibility, since the incoming messages—from which beliefs are calculated—are not generally sparse. We have also introduced lexical decision into a model that also includes other important linguistic cues, such as bigrams.

4.5 Conclusions

In this chapter we have presented a model for character recognition that ties together several important information sources. We have shown that the unified model clearly improves results over pipelined processing. No doubt many opportunities exist to add other information sources. A richer character recognition model could easily be incorporated to boost accuracy, and higher order n-grams for both characters *and* words could be added. All manner of language models could be considered, and there is likely much mileage to be gained by integrating these with the recognition process, rather than using them as post-processors.

This model still is geared only for recognition. In the next chapter, we present a method for coupling the earlier detection stage—like that discussed in Chapter 3—and the recognition stage by jointly learning detection and recognition models.

CHAPTER 5

UNIFYING DETECTION AND RECOGNITION

The task of reading involves sensing an image, locating the regions of text, and identifying constituent characters. As outlined in the introduction, these processes are generally treated in a hierarchical pipeline by first running a detector and then feeding detections into an appropriate recognizer. In the previous chapter, we demonstrated a flexible model for recognition. However, it assumed that the characters had already been detected and only needed to be recognized. In Chapter 3 we saw an illiterate detection model. It performed reasonably well but still suffered from false positives where a small amount of interpretation might eliminate non-text regions from further consideration. In this chapter, we begin to knit these processes more tightly by considering them jointly.

Classifiers for object categorization and identification, such as face detectors and face recognizers, are often trained separately and operated in a feed-forward fashion. Selecting a small number of features for these tasks is important to prevent over-fitting and reduce computation. However, when a system has such related or sequential tasks, independently training and selecting features for these tasks may not be optimal. Here we propose a framework for choosing features to be shared between categorization (detection) and identification (recognition) tasks. The result is a system that achieves better performance with a given number of features. We demonstrate with experiments using text and car detection as categorization tasks, and character and vehicle type recognition as identification tasks.

5.1 Overview

Many real-world problems must solve multiple classification tasks simultaneously or sequentially. For example, a vision system may need to discriminate between cars, people, text, and background as high-level categories, while also recognizing particular cars, people, and letters. The categorization/detection task is to determine whether an image region corresponds to an object from a class of interest (e.g., text) or not. The identification/recognition task discriminates among members of that category (e.g., if this is text, is the character a `p` or a `q`?). Often the categorization and identification tasks are treated in a sequential manner by first running a category-level detector and then feeding detections into a category-specific recognizer. Moreover, although the classifiers for the two tasks are related, they are usually trained independently. This work seeks to knit these processes more tightly by considering them jointly during model training.

Figure 5.1. The categorization/detection task must only discriminate characters (left) from background patches (right), while the identification/recognition task must identify the centered character.

Constructing a model for a classification task involves many issues, including deciding which features or observations are relevant to the decision. Two reasons for limiting the number of features involved in classification include preventing over-fitting and reducing the amount of computation needed to reach a decision. Models with too many irrelevant features are prone to poor generalization since they are fit to unnecessary constraints. Even when there is no over-fitting, if certain features are redundant or unnecessary, the classification process can be expedited by eliminating the need to compute them.

Feature selection may be important for both detection and recognition, the primary difference being the generality of the classification tasks. However, if these problems are treated in isolation, we may not achieve a feature selection that is optimal—in computational or accuracy terms—for the *joint* detection-recognition problem .

While some features will undoubtedly be useful primarily for detecting object categories (e.g., text) and others will have the greatest utility for recognizing objects from a particular category, there may be some features with utility for *both* tasks. When this is the case, a method accounting for overlap in utility may have two advantages. First, a feature useful for object identification may boost category detection rates for the class by incorporating more object-specific information in the search. Second, if such dual-use features have already been computed for the purposes of category detection, they may subsequently be utilized for identification, effectively reducing the amount of computation necessary to make a classification.

In this chapter, we propose a framework for jointly considering the general categorization and specific identification tasks when selecting features and compare it to two other approaches. The first attempts only to predict identities and has no explicit representation of categories. The second approach has a category model and several category-specific recognizers, all of which are trained independently. Our approach is to jointly learn and select features for the category and identity models. The three methods are compared in experiments on text and vehicle detection and recognition, examining both the overall and category-level classification accuracy as the number of features increases. We find that the joint approach reduces error by 50% over the independent approach when the number of features is small and 20% as the number increases.

Our earlier character recognition approach used a "flat" model for discrimination (cf., §4.2.1). That is, oriented edge features were calculated on a grid laid over the

character and these were directly used to train the discriminant. An alternative, and more robust strategy, is to calculate some intermediate features. These intermediate "part-based" features operate on small, local portions of the low-level features, and typically correspond to higher level combinations of the lower features. There is psychological evidence for both the flat and parts-based models in humans [98], and input processing time generally determines which is used. On average, it is the part-based model that is more robust, as shown in both psychological experiments and practical computational models. For this reason, we pursue a parts-based approach to character recognition.

We reviewed relevant work on visual features and feature selection in Section 1.4. Next we will present the alternative strategies for model learning highlighted above, followed by the various features available for use in our classification models. Experiments and analysis demonstrating the advantage of joint feature selection are then given, and we conclude by highlighting our contributions.

5.2 Simple Models for Detection and Recognition

For every query image or sub-image region, our goal is to determine whether the query belongs to some general category of interest, and if so, to recognize it as a particular instance of that category. For instance, given an image region, we may first want to determine whether it is text, and if so, to identify the character. Thus, every query is assigned both a category and category-specific identity; if it does not belong to any particular category of interest, we may call it "background." First we give several model formulations for this problem, followed by the feature selection strategies for each.

5.2.1 Model Formulations

In this section, we describe three alternatives to modeling and training (Figure 5.2). First, we describe the common method of treating detection and recognition independently, followed by a simple flat model that forgoes category modeling and only aims to identify each query. Finally, we propose a factored model for jointly learning detection and recognition.

Let $c \in \mathcal{C}$ represent the category (e.g., text, vehicle, etc.) of an image query $\mathbf{x}$. For each category c, there is a set $\mathcal{Y}_c$ of particular objects in that category; e.g., $\mathcal{Y}_{\mathsf{c}}$ might be characters for the text category and $\mathcal{Y}_{\mathsf{v}}$ might be classes of vehicles. If we include a background category $\mathsf{b} \in \mathcal{C}$ for other regions, then every image region takes a category label from $\mathcal{C}$ and an identity from $\mathcal{Y} = \{\mathsf{b}\} \cup \bigcup_{c \in \mathcal{C}} \mathcal{Y}_c$. We assume that objects belong to only one category.

We will use the same class of discriminative Markov models described in Chapter 2. In this chapter, all the models will have only one unknown. Thus, they belong to a class of "local" classifiers that are equivalent to the familiar discriminative maximum entropy distribution or multinomial logistic regression. To further clarify the exposition on feature selection in the next section, we explicitly include the features in the

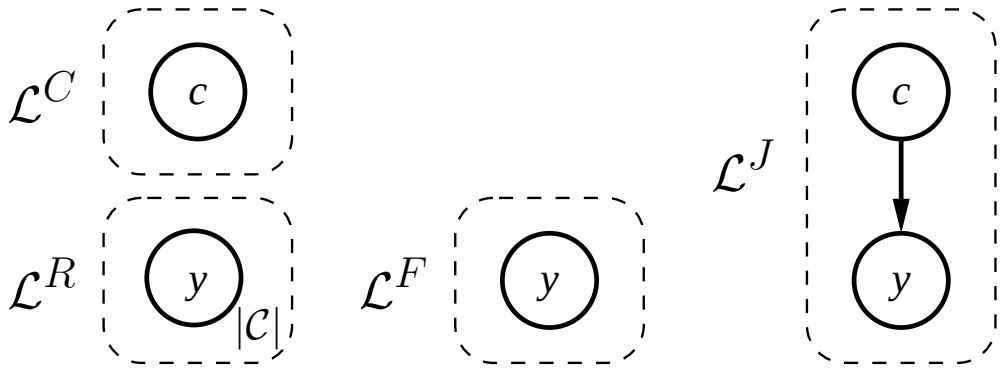


Figure 5.2. Graphical representation of the models and training strategies. Dashed lines indicate the unknowns considered by a training objective function. LEFT: Independent models (there are $|\mathcal{C}|$ recognition objectives, one for each category). CENTER: Flat model. RIGHT: Factored joint model.

probability conditioning information, compatibility function arguments, and learning objective functions, rather than having them fixed and implicit as before.

Models for the category detection and recognition problems are typically learned independently. Formally, our category detection model $p\left(c \mid \mathbf{x}, \boldsymbol{\theta}^C, F, I\right)$ has only one compatibility function, in the typical linear form

$$U^C\left(c, \mathbf{x}; \boldsymbol{\theta}^C, F\right) = \boldsymbol{\theta}^C(c) \cdot F(\mathbf{x}) \tag{5.1}$$

for $c \in \mathcal{C}$. Similarly, a recognition model will be a probability conditioned on the category, $p\left(y \mid c, \mathbf{x}, \boldsymbol{\theta}^R, G, I\right)$, also with a single compatibility function

$$U^R\left(y, c, \mathbf{x}; \boldsymbol{\theta}^R, G\right) = \begin{cases} \boldsymbol{\theta}_c^R(y) \cdot G_c(\mathbf{x}) & y \in \mathcal{Y}_c \\ -\infty & \text{otherwise} \end{cases} \tag{5.2}$$

where $G = \{G_c\}_{c \in \mathcal{C}}$ are the features of $\mathbf{x}$ for identifying objects in a category c, and $\boldsymbol{\theta}^R = \left\{\boldsymbol{\theta}_c^R\right\}_{c \in \mathcal{C}}$ are the corresponding parameters. The parameters and features are indexed by c because recognition models for different categories need not use the same features. With the infinite term in (5.2), we have defined $p\left(y \mid c, \mathbf{x}, \boldsymbol{\theta}^R, G\right) = 0$ for $y \notin \mathcal{Y}_c$, which says it is logically impossible for the identity y to be outside of the given category c. This is equivalent to defining $|\mathcal{C}|$ separate models that only make predictions of identities from a specific category.

Given a set of examples having category and identity labels $\mathcal{D} = \left\{\left(c^{(k)}, y^{(k)}, \mathbf{x}^{(k)}\right)\right\}_k$, the typical method of training is to independently optimize log posteriors for the categorization and category-specific recognition models. Here, we rewrite equations (2.12-2.14) from page 23 for these particular models.

For categorization, the objective function is the log posterior for $\boldsymbol{\theta}^C$:

$$\mathcal{O}^C\left(\boldsymbol{\theta}^C; F, \mathcal{D}\right) \equiv \mathcal{P}\left(\boldsymbol{\theta}^C\right) + \mathcal{L}^{\mathcal{C}}\left(\boldsymbol{\theta}^C; F, \mathcal{D}\right) \tag{5.3}$$

$$\mathcal{P}^C\left(\boldsymbol{\theta}^C\right) \equiv \log p\left(\boldsymbol{\theta}^C \mid I\right) \tag{5.4}$$

$$\mathcal{L}^C\left(\boldsymbol{\theta}^C; F, \mathcal{D}\right) \equiv \sum_k \log p\left(c^{(k)} \mid \mathbf{x}^{(k)}, \boldsymbol{\theta}^C, F, I\right). \tag{5.5}$$

We may similarly define recognition objective functions for *each category* separately using a prior $\mathcal{P}_c^R$ and likelihood $\mathcal{L}_c^R$. The likelihood then takes the form

$$\mathcal{L}_c^R\left(\boldsymbol{\theta}^R; G, \mathcal{D}\right) \equiv \sum_{k:c^{(k)}=c} \log p\left(y^{(k)} \mid c^{(k)}, \mathbf{x}^{(i)}, \boldsymbol{\theta}^R, G\right). \tag{5.6}$$

There are equivalently $|\mathcal{C}|$ recognition models, with a $\boldsymbol{\theta}_c^R$ and G_c for each category, so we may write the total recognition likelihood

$$\mathcal{L}^R\left(\boldsymbol{\theta}^R; G, \mathcal{D}\right) \equiv \sum_{c\in\mathcal{C}} \mathcal{L}_c^R\left(\boldsymbol{\theta}^R; G, \mathcal{D}\right). \tag{5.7}$$

The forms of category specific priors $\mathcal{P}_c^R$ and posteriors $\mathcal{O}_c^R$ are analogous to (5.4) and (5.3), respectively, while and the total prior $\mathcal{P}^R$ and posterior $\mathcal{O}^R$ are analogous to (5.7). Training separate models with independent objectives $\mathcal{L}^C$ and $\mathcal{L}^R$ is depicted graphically in the left pane of Figure 5.2.

Since every object belongs to only one category, an alternative is to forgo category modeling altogether, and simply have one "flat" model $p\left(y \mid \mathbf{x}, \boldsymbol{\theta}, F\right)$ that aims to determine identity from among all possible labels using the compatibility function

$$U\left(y, \mathbf{x}; F, \boldsymbol{\theta}\right) = \boldsymbol{\theta}\left(y\right) \cdot F\left(\mathbf{x}\right), \tag{5.8}$$

for $y \in \mathcal{Y}$. If it is needed, the probability for a category label c can be calculated by summing the probabilities for all $y \in \mathcal{Y}_c$. Training then involves optimizing a single posterior, with the likelihood

$$\mathcal{L}^F\left(\boldsymbol{\theta}; G, \mathcal{D}\right) \equiv \sum_k \log p\left(y^{(k)} \mid \mathbf{x}^{(k)}, \boldsymbol{\theta}, G, I\right). \tag{5.9}$$

One potential problem with this approach is that the label space $\mathcal{Y}$ is potentially very large. Training such a unified flat model with the single objective $\mathcal{L}^F$ is depicted in the center pane of Figure 5.2

Alternatively, the joint probability for categorization and recognition can be written as the product of two probabilities:

$$p\left(c, y \mid \mathbf{x}, \boldsymbol{\theta}^C, \boldsymbol{\theta}^R, F, G, I\right) = p\left(y \mid c, \mathbf{x}, \boldsymbol{\theta}^R, G, I\right) p\left(c \mid \mathbf{x}, \boldsymbol{\theta}^C, F, I\right), \tag{5.10}$$

In the next section, we consider the case where the same features $F = G_c$ are used by both the categorization and all recognition models, but different parameters (discriminative feature weights $\boldsymbol{\theta}^C$ and $\boldsymbol{\theta}^R$) are used by each. Training the joint model $p\left(c, y \mid \mathbf{x}, \boldsymbol{\theta}^C, \boldsymbol{\theta}^R, F, G, I\right)$ now involves accounting for *both* the categorization and recognition models *simultaneously*, rather than treating them independently. The joint likelihood is thus

$$\mathcal{L}^J\left(\boldsymbol{\theta}^C, \boldsymbol{\theta}^R; F, G, \mathcal{D}\right) \equiv \mathcal{L}^C\left(\boldsymbol{\theta}^C; F, \mathcal{D}\right) + \mathcal{L}^R\left(\boldsymbol{\theta}^R; G, \mathcal{D}\right), \tag{5.11}$$

with the posterior $\mathcal{O}^J$ and prior $\mathcal{P}^J$ being analogous sums. Although the previous flat model (5.8) makes predictions over the same label space, the factored model (5.10)

is hierarchical (as shown in Figure 5.2), which has several potential benefits. By acknowledging the existence of categories, we may improve the results by learning the (presumably similar) categorical instances together, creating more training data at the higher level. This may also decrease the computational burden of prediction if the category classifier is run first, since $|\mathcal{C}| \ll |\mathcal{Y}|$. The difference between factored training with (5.11) and the independent training with (5.5) and (5.7) is manifest during feature selection; this will also potentially have an impact on computational performance. A graphical overview of all three models and training strategies is shown in Figure 5.2.

If categorization and recognition models are learned independently, the features used to make a category decision might not overlap with the features used for recognition, possibly increasing the total amount of computation. Furthermore, objects in the same category are visually related, yet the flat model will handle some separately. Learning category-level models may improve category detection and the ultimate object identification results by modeling them together. In the next section, we elaborate on feature selection for this class of models.

5.2.2 Feature Selection

The algorithm we use for selecting features is a greedy forward method that incrementally adds the feature providing the greatest increase in the log posterior objective function being optimized [9]. Each model has its own objective (e.g., $\mathcal{O}^C$, $\mathcal{O}^R_c$, $\mathcal{O}^F$, and $\mathcal{O}^J$), and the same algorithm is used on each.

As an example, consider the categorization probability $p\left(c \mid \mathbf{x}, \boldsymbol{\theta}^C, F, I\right)$ and corresponding parameter posterior objective function $\mathcal{O}^C\left(\boldsymbol{\theta}^C; F, \mathcal{D}\right)$. At some iteration of feature selection, the model includes a set of features F (initially empty) and parameters $\widehat{\boldsymbol{\theta}^C}$ that optimize the posterior $\mathcal{O}^C$. Then, some new candidate feature f is added to the feature set F. Let the augmented features be F' and the corresponding augmented parameters be $\boldsymbol{\theta}^{C\prime}$. After optimizing the posterior of the augmented model to find the new optimal parameters $\widehat{\boldsymbol{\theta}^{C\prime}}$, we may calculate the "gain" of the candidate feature by taking the difference of log posteriors

$$\mathcal{G}^C\left(f; \mathcal{D}\right) = \mathcal{O}^C\left(\widehat{\boldsymbol{\theta}^{C\prime}}; F', \mathcal{D}\right) - \mathcal{O}^C\left(\widehat{\boldsymbol{\theta}^C}; F, \mathcal{D}\right), \tag{5.12}$$

which is equivalent to a probability ratio test of the two models. We may thus calculate the gain of all features from a pool of candidates and add the feature with the highest gain to the model. The model is built by iteratively adding the best (highest gain) feature.

The same process may be followed for the flat model objective $\mathcal{O}^F$:

$$\mathcal{G}^F\left(f; \mathcal{D}\right) = \mathcal{O}^F\left(\widehat{\boldsymbol{\theta}'}; G', \mathcal{D}\right) - \mathcal{O}^F\left(\widehat{\boldsymbol{\theta}}; G, \mathcal{D}\right). \tag{5.13}$$

With the factored joint model objective, we ensure that the same candidate feature f is added to all the models (e.g., $\mathcal{O}^C$, and $\mathcal{O}^R_c$ for each $c \in \mathcal{C}$). Thus, we maintain the same set of features $F = G_c$ for all models in the gain calculation:

$$\mathcal{G}^J\left(f; \mathcal{D}\right) = \mathcal{O}^J\left(\widehat{\boldsymbol{\theta}^{C\prime}}, \widehat{\boldsymbol{\theta}^{R\prime}}; F', G', \mathcal{D}\right) - \mathcal{O}^J\left(\widehat{\boldsymbol{\theta}^C}, \widehat{\boldsymbol{\theta}^R}; F, G, \mathcal{D}\right). \tag{5.14}$$

For the independent method, each category-specific objective $\mathcal{O}_c^R$ goes through its own feature selection process with the gain

$$\mathcal{G}_c^R(f;\mathcal{D}) = \mathcal{O}_c^R\left(\widehat{\boldsymbol{\theta}^{R\prime}};G',\mathcal{D}\right) - \mathcal{O}_c^R\left(\widehat{\boldsymbol{\theta}^{R}};G,\mathcal{D}\right) \tag{5.15}$$

to determine the category-specific recognition features G_c.

Since many candidate features may need to be examined at each feature selection iteration, approximations are helpful for speeding the process. First, only the augmented parameters for the candidate features are optimized [9], e.g., fixing the optimal parameters $\widehat{\boldsymbol{\theta}^C}$ for the already selected features F in the gain calculation (5.12). Second, we calculate the gains on a representative subset of the training data $\mathcal{D}' \subset \mathcal{D}$, and then re-calculate the gains of only the top-ranked features using all the training data. Alternative approximations include assuming that feature gains are monotonically decreasing as other features are added [139], "grafting," which adds the feature whose likelihood gradient is greatest [91], or only including training instances currently misclassified in the data $\mathcal{D}$ for the gain calculation [79].

When two separate models are independently trained for a pipelined framework, the gain of a feature is only measured with respect to a particular task, categorization or recognition. However, considering the *entire* end-to-end task of categorization and recognition will yield a different ranking of the features in general. We show in Section 5.4 that jointly considering tasks during feature selection improves performance accuracy and speed.

5.3 Detection and Recognition Features

In this section, we describe the two types of candidate features for our model: region-based texture features and local template features.

All features are derived from the steerable pyramid wavelet basis [109], a set of scale and orientation selective filters that loosely resembles the "simple cells" in an initial layer of processing in mammalian visual systems The wavelet coefficients are complex, representing outputs from paired even and odd filters for each scale and orientation (channel). Taking complex magnitudes yields phase invariant responses, similar to complex cells in biological systems.

The first pool of candidate features is a set of image and wavelet statistics [93] originally crafted for texture synthesis, and used for text detection in Chapter 3. These include image statistics (four central moments plus min and max) at several scales, means of wavelet channel magnitudes, and local auto- and cross-correlation of wavelet channels. Although originally intended to be computed globally over an image of ergodic texture, we compute them "locally" over small image regions, which can be efficiently achieved by convolution. These features are described in greater detail in Chapter 3 (cf., §3.3).

In Chapter 4, we designed a character classifier that used the Gabor filter-based wavelet channel moduli directly as features. However, these are not robust to image deformations and our model did not capture any joint relationships between the filter

responses. Research in cognitive psychology by Rehling [98] indicates that two mechanisms operate in human character recognition: an initial "flat" recognizer that is fast and a secondary hierarchical, parts-based model like LeCun's convolutional network [65] that is slower but more accurate. To construct a model of this hierarchical framework, template-based feature maps form our second pool of candidates.

Calculating a template-based feature map from the wavelet channel magnitudes involves five steps:

1. feature vector normalization (a form of contrast normalization),
2. dilation and downsampling,
3. template correlations,
4. a summation over template channels,
5. and another dilation and downsampling.

Next we describe the processing steps for calculating the feature maps in greater detail.

The steerable pyramid is a linear wavelet transform. Let $\mathbf{s}^c$ be a filter for the particular oriented channel c. An input image region $\mathbf{x}$ is convolved with these filters and the complex modulus (taking even filter responses as the real part and odd filter responses as the complex part) is computed:

$$\mathbf{w}^c \equiv |\mathbf{s}^c * \mathbf{x}|, \tag{5.16}$$

where $|\cdot|$ is the complex modulus operator.

Because the input $\mathbf{x}$ is of unknown contrast, the differential filters of the steerable pyramid might have either strong or weak responses. It is not necessarily the magnitude of these responses that we are most interested in, but how they describe a shape together. Therefore, the wavelet magnitudes are locally normalized by a process similar to that of SIFT [71]: at each location, all the wavelet magnitudes in a local $a \times a$ window centered at pixel i are normalized to a unit ℓ_2 norm. This includes all channels c of the wavelet map $\mathbf{w}$. After normalization, the responses are clipped at a threshold (0.2 in our experiments). The same responses are then re-normalized in the same fashion. The resulting values for each channel c are taken as the normalized responses only at the center location i. Thus, a different window is used to normalize each location.

Next, to decrease spatial and phase sensitivity, the image's normalized wavelet magnitudes are downsampled after taking the maximum over a small window within each channel (a simple morphological dilation). Dilation incorporates spatial pooling and provides some amount of flexibility with regard to edge feature location, while downsampling yields a more compact stack of scale- and orientation-specific images. The wavelet map as defined in (5.16) will always undergo these post-filtering transformations in this work, so henceforth we use $\mathbf{w}^c$ to refer to the transformed version.

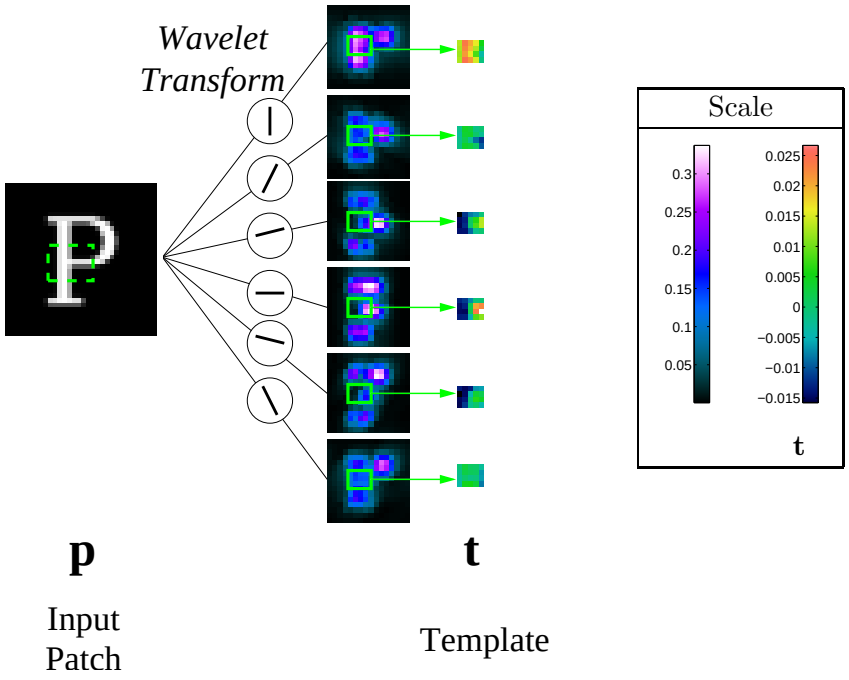


Figure 5.3. Extraction of a template from a patch of a training image. Scale and orientation selective wavelet features are applied to the training image (including the patch area $\mathbf{p}$ outlined in the dashed green box) to yield a wavelet map. The area of the wavelet map corresponding to the patch is extracted and used as a template for further processing.

A *template* $\mathbf{t}$ is a small patch extracted from the values of a training example's processed wavelet magnitudes. Let $\mathbf{p}$ be a training image patch, then we define a template channel in the same way:

$$\mathbf{t}^c \equiv |\mathbf{s}^c * \mathbf{p}| \tag{5.17}$$

Each $\mathbf{t}$ is subsequently normalized to have zero mean and unit ℓ_1 norm, over all channels. An illustration of the template extraction is given in Figure 5.3.

The *feature map* $f_{\mathbf{t}}$ for such a template is calculated by cross-correlations between an input image's wavelet features $\mathbf{w}$ and the corresponding channels from the template $\mathbf{t}$—a sum of correlations over each channel c :

$$f_{\mathbf{t}}(\mathbf{x}) = \sum_c \mathbf{t}^c \otimes \mathbf{w}^c, \tag{5.18}$$

where $\otimes$ is the cross-correlation operator. The feature map $f_{\mathbf{t}}$ is then subject to another dilation and downsampling for even further spatial pooling and dimensionality reduction. An illustration of the image-to-feature map calculation is given in Figure 5.4. Several example feature maps are shown in Figure 5.5.

Our ultimate goal will be to select the texture statistics or templates that are most useful for a particular task, be it detection, recognition, or both. Next, we

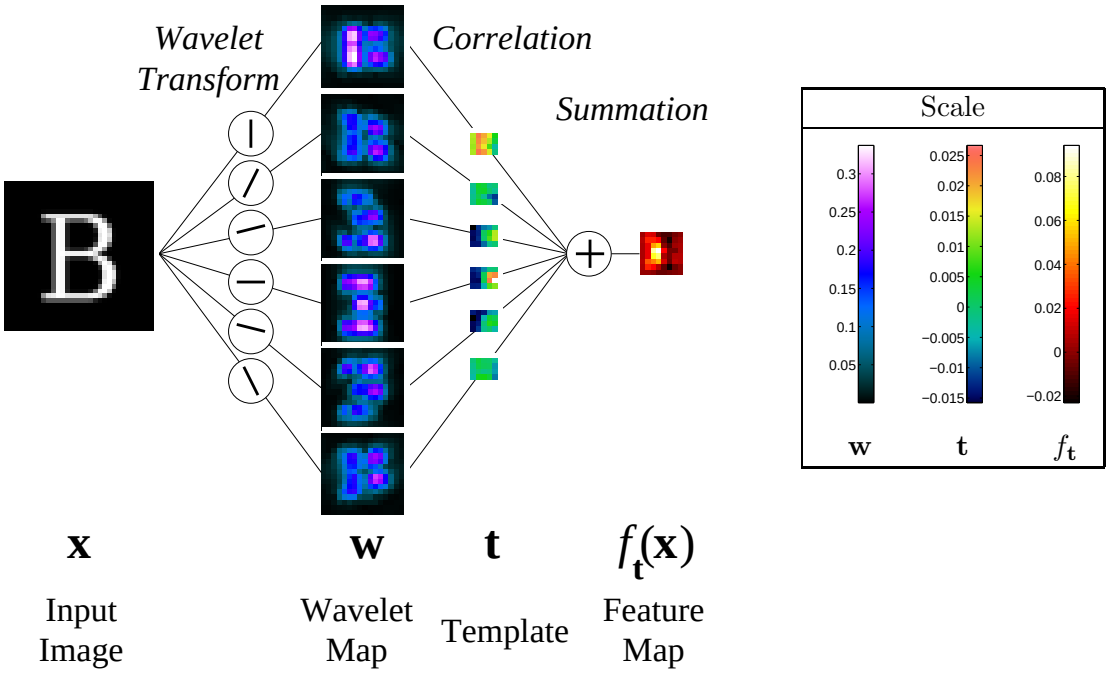


Figure 5.4. Computation of a single feature map from one template. Scale and orientation selective wavelet features are applied to input image $\mathbf{x}$, followed by a normalization and dilation/downsampling to yield $\mathbf{w}$. Correlations between corresponding channels in a template $\mathbf{t}$ and wavelet map $\mathbf{w}$ are computed and summed, followed by another dilation/downsampling to yield a feature map f. Several additional templates would be used to provide an array of feature maps, which becomes a feature vector for the classifier.

compare the results of selecting these features under the various strategies outlined in the previous section.

5.4 Experiments

In this section, we compare four training and feature selection strategies for category detection and recognition:

1. The flat classifier trained by (5.13),
2. the factored classifier jointly trained by (5.14),
3. the independently trained classifiers trained by (5.12) and (5.15), operated sequentially, and
4. the independent classifiers operated sequentially, but trained using the only the features selected for categorization by (5.12).

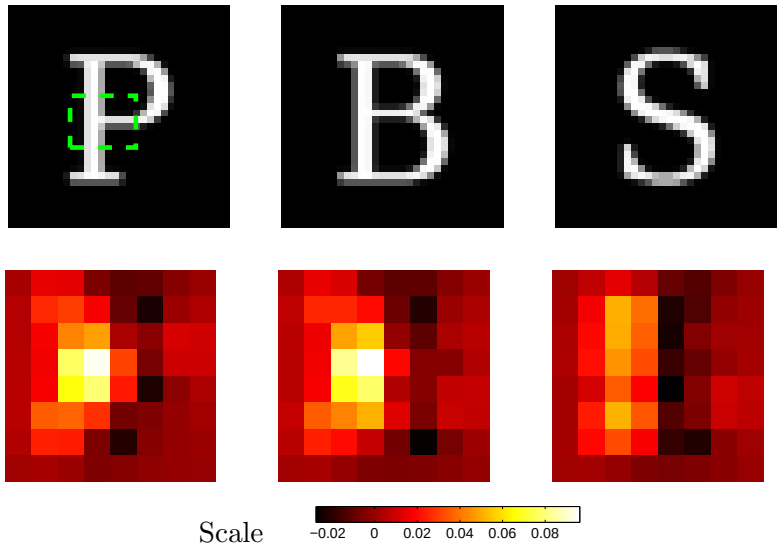


Figure 5.5. A region (dashed box in top left image) used to construct a template $\mathbf{t}$ and some resulting feature maps $f_{\mathbf{t}}$ of the template on different inputs $\mathbf{x}$.

To test our hypothesis that joint feature selection can improve speed and accuracy, we need data with labels for background, as well as both category and identity. We perform experiments on a three category problem involving background, text, and vehicles, with synthetic but realistically difficult data for the latter two categories.

5.4.1 Experimental Data

In this section we describe the data from our three categories.

Background The images from scenes around a downtown area used for detection in Chapter 3 (cf., §3.4.1) have had the sign regions manually masked out, and square patches of various scales (128×128, 64×64, and 32×32 all resized to 32×32) from the non-sign regions were extracted and labeled as background. A few examples are shown in Figure 5.1 on page 78.

Characters Rather than manually crop and label individual characters from image regions, we synthesize similar character images. There are 62 characters in our alphabet to be recognized (uppercase, lowercase, and digits), rendered in 934 different fonts at a 12.5 pixel x-height and centered in a 32×32 window. Neighboring characters were sampled from bigrams learned on a corpus of English text (cf., §4.3.1) and placed with uniform random kerning/tracking. The resulting trigram image was then subject to a random linear transformation

$$T = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \sigma_x & \rho_x \\ \rho_y & \sigma_y \end{bmatrix} \tag{5.19}$$

with rotation $\theta \sim \mathcal{N}(0, 5^\circ)$, scale factors $\sigma_x, \sigma_y \sim \mathcal{N}(1, 0.08)$, and skew factors $\rho_x, \rho_y \sim \mathcal{N}(0, 0.1)$. Despite the fact that a contrast normalization scheme is used, we use a random contrast $c \sim \text{Beta}(\alpha = 1.72, \beta = 2.89)$ and brightness $b \sim \text{Uniform}[0, 1 - c]$ in the bilevel character training images. Additive Gaussian noise $\epsilon \sim \mathcal{N}(0, 10^{-4})$ is also added to the images. All these distortion parameters are roughly modelled after the text from the scene images used in Chapters 3 and 4. Adding these factors to the data set allows the classifier to learn them and provides a reasonable test bed without having to manually ground truth individual characters in many images. The label of these character windows is the center character. Examples are shown in Figure 5.6, and may be compared to actual images of scene text in Figure 5.1 on page 78. Note that the identification task involves no character segmentation—the character in the center of the window must be recognized in the presence of neighboring character "clutter."

Vehicles Images of 21 vehicles rendered from three viewing directions and nine lighting conditions are used from work on vehicle class recognition by Ozcanli et al. [88][1]. The vehicle identification task then consists of labeling the image as one of SUV, passenger car, pickup truck, or van. Examples of the 32×32 images used are in Figure 5.6. At such small resolution, recognition is very difficult; while published experiments with this data provide the viewing angle and lighting conditions [88], our recognition model achieves similar results (57% accuracy) when the view and lighting are *unknown*.

Training Data Summary For the background category, our training set has roughly 65,000 windows at multiple scales from images of outdoor scenes. The character class has nearly 30,000 character windows (each of 62 characters in 467 fonts). The vehicle class has nearly 300 examples. The test set is roughly the same size but comes from a different set of scene images, fonts, and vehicles. Indeed, if we use the same fonts for testing even with different distortions applied, the recognition results are much higher.

5.4.2 Experimental Procedure

Features As shown in Figure 5.4 on page 86, the wavelet transform of a given 32×32 patch is dilated over a 2×2 window and then downsampled to 16×16., where template correlations are calculated. The resulting feature map is then dilated over a 4×4 window and downsampled to 4×4 for an extremely compact representation of responses for each template. Candidate template patches of various sizes (height and width selected uniformly from 2, 4, 6, 8, or 10 pixels) were randomly extracted from the training *character* images (no vehicle or background images were used). The candidate feature pool contains 2,000 template patches and 418 local wavelet statistics (from 6 orientations and 3 scales) .

[1] `http://www.lems.brown.edu/vision/researchAreas/vehicle_recognition`

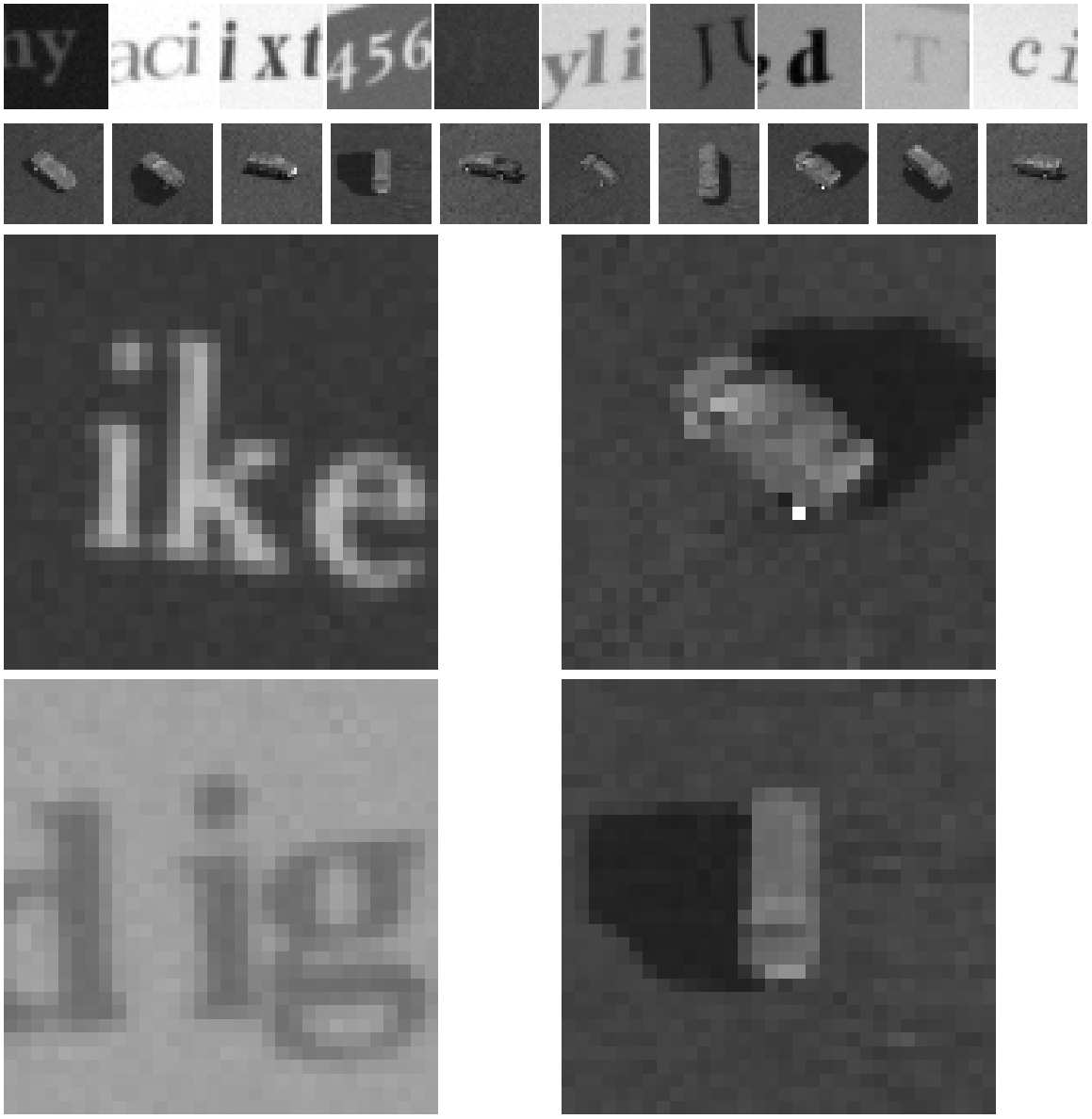


Figure 5.6. Sample images used in experiments. TOP: Randomly distorted synthetic characters. MIDDLE: Vehicles from various classes, views, and lighting conditions. BOTTOM: Enlarged views of the low resolution character and car data.

Regularization In all cases, a Laplacian (ℓ_1) prior was used for regularization, and the value of the hyperparameter α was chosen by cross-validation. The training set was split in two, half was used for training, and the value of α that yielded the highest likelihood on the other half was then used on the entire training set. All of the candidate features were included for cross-validation, since we do not know which might be useful *a priori*. However, a slightly smaller portion of the training data was used since all features for all instances vastly exceeded memory limits.

Class Priors Since text and vehicles are relatively rare in natural scenes, we weight all the data instances in training and test evaluation such that characters and vehicles both have a class prior of 1×10^{-4}; in other words, the effective ratio of text to background is almost one to ten-thousand. This is to make the experiments more realistic.

In particular, if the training data $\mathcal{D}$ includes a weight $w^{(k)}$ for each instance k , the log likelihood of each instance is multiplied by the weight in the likelihood function. For instance, the category likelihood (5.5) becomes

$$\mathcal{L}^C\left(\boldsymbol{\theta}^C; F, \mathcal{D}\right) = \sum_k w^{(k)} \log p\left(c^{(k)} \mid \mathbf{x}^{(k)}, \boldsymbol{\theta}^C, F, I\right). \tag{5.20}$$

We set the weights such that

$$w^{(k)} = N \frac{\nu}{N_{c^{(k)}}}, \tag{5.21}$$

where N is the total number of training instances and $N_{c^{(k)}}$ is the number of training instances of a particular category. The parameter $\nu = 10^{-4}$ for $c = \mathsf{Text}$ and $c = \mathsf{Vehicle}$, while $\nu = 1 - 2 \cdot 10^{-4}$ for the $\mathsf{Background}$ category. Like the original unweighted likelihood (where $w^{(k)} = 1$ implicitly), with this scheme, the total weight mass is N: $\sum_k w^{(k)} = N$.

In our evaluation, to normalize for category difficulty the accuracy is averaged over categories,

$$A_c^C \equiv \frac{1}{N_c} \sum_k \delta\left(c^{(k)}, \widehat{c}^{(k)}\right), \; c \in \mathcal{C} \tag{5.22}$$

$$A^C \equiv \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} A_c^C, \tag{5.23}$$

where the true category label is c and the MAP prediction is $\widehat{c}$ (cf., §2.3.1). To normalize for class difficulty and amount of training data, we calculate an average recognition accuracy that is the average of the accuracies for each class (equivalently, the mean of the diagonal entries in a normalized confusion matrix):

$$A_y^R \equiv \frac{1}{N_y} \sum_{k: y^{(k)} = y} \delta\left(y^{(k)}, \widehat{y}^{(k)}\right), \; y \in \mathcal{Y} \tag{5.24}$$

$$A^R \equiv \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} A_y^R, \tag{5.25}$$

where N_y is the number of test instances from a particular class $y \in \mathcal{Y}$.

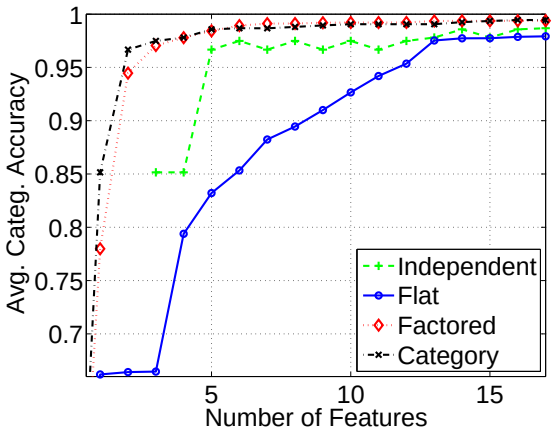


Figure 5.7. Average categorization accuracy A^C (cf., Eq. (5.23)) for four feature selection strategies.

When the independently trained models are used sequentially and the number of features is bounded, we must decide how to allocate the given number of features among those chosen by the various models. In a two category problem, this involves choosing how many selected by the categorization model (i.e., text detector) are used, leaving the remaining number for the identification model (i.e., character recognizer). This is a relatively straightforward one-dimensional optimization, but it becomes more complex for three or more categories. The problem is simplified by allocating features between categorization and recognition, where each category-specific recognition model is allotted the same number of features. This one-dimensional strategy may be sub-optimal when some categories are harder to distinguish or are more important than others. In our experiments, this one-dimensional approximate optimization strategy is performed on the *test* data, so the results of the independent method are optimistic.

5.4.3 Experimental Results

Here we describe and present the results of our experiments for the three category problem involving text, vehicles, and background. Section 5.4.4 contains an analysis and discussion of these results.

In Figures 5.7 and 5.8, the performance is shown at each round of feature selection for the flat and factored models, as well as the independently trained sequential model and the model that uses only the top categorization features.

Figure 5.7 shows the average categorization accuracy (5.23) of the four methods.

Figure 5.8 shows the average recognition accuracy (5.25) of the four methods. Two views of the curve are shown to highlight performance with different numbers of features. The left view shows the early performance, with few features, while the right view gives the trend as more features are added. The flat model curve terminates before the others because the natural training termination condition (insufficient in-

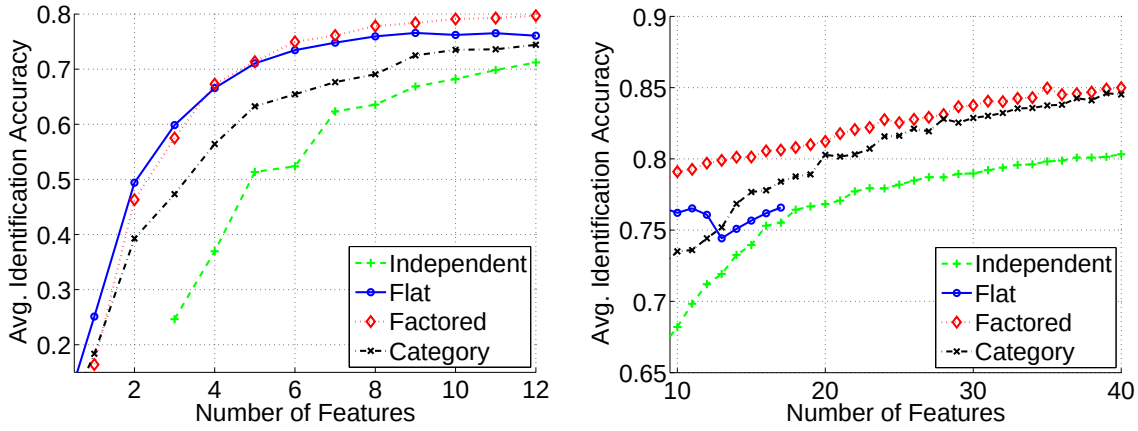


Figure 5.8. Average recognition accuracy A^R (cf., Eq. (5.25)) for four feature selection strategies. Note that the left and right plots are different views of the same curve.

crease in gain $\mathcal{G}^F$) was met and each round of feature selection takes much longer than for the other models. The curve for the independent method begins at three features because one each is allocated to the categorization model, text recognition model, and vehicle recognition model (the background category has no further sub-category recognition to be done and therefore does not require a recognition model).

Figure 5.9 shows the relative improvement (reduction in error on average identification accuracy) of the factored joint model over the optimal independently trained models.

To demonstrate how joint feature selection can impact performance, we compare the independent and factored strategies in a two category experiment involving only background and text, our primary problem of interest. Figure 5.10 compares the relative gains of features under the different strategies. For comparison, we plot *normalized* gains for each strategy. Considering the categorization task as an example, the normalized gains are defined as

$$\widetilde{\mathcal{G}}^C\left(f;\mathcal{D}\right) \equiv \frac{\mathcal{G}^C\left(f;\mathcal{D}\right)}{\mathcal{G}^C\left(\widehat{f}^C;\mathcal{D}\right)}, \tag{5.26}$$

where $\widehat{f}^C$ is the best feature

$$\widehat{f}^C \equiv \arg\max_f \mathcal{G}^C\left(f;\mathcal{D}\right).$$

Normalized gains for the other strategies may be calculated similarly. Thus, the y-axis in Figure 5.10 shows what fraction of the best possible gain is achieved by an alternative feature.

For example, in the top-left graph, we give a uniformly spaced sampling of the features sorted by their rank according to $\widetilde{\mathcal{G}}^C\left(f;\mathcal{D}\right)$. The best feature for categorization (text detection) is #2222, while the worst is #416. In this graph we see that the

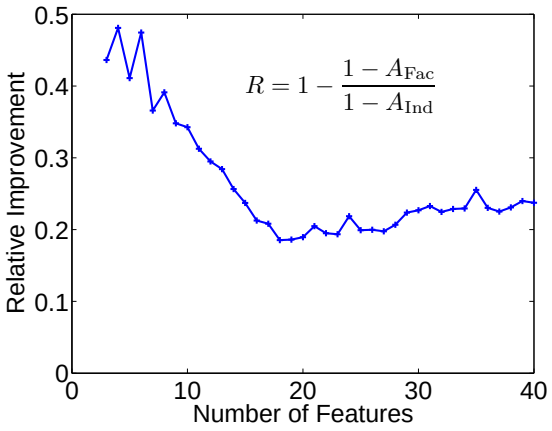


Figure 5.9. Relative improvement of the factored over independent method on the average identification accuracy (A_{Fac} and A_{Ind}).

best feature for detection of course has a normalized gain of 1.0 for the detection task (the leftmost blue bar), while the normalized gain for this feature on the recognition task is only about 0.78 (the leftmost green bar).

A scatterplot of normalized feature gains under the categorization, recognition, and factored joint models for the first round of feature selection are shown in Figure 5.11. This provides a visualization of the gain of all features for each task simultaneously. For example, a green plus (+) in the lower-right portion of the graph represents a feature that is very good for detection, very bad for recognition, and mediocre for the joint task.

5.4.4 Discussion

Our experimental results demonstrate the superiority of a factored (hierarchical) joint feature selection over the traditional independent method in several ways. The first and most obvious way is that the average identification accuracy of the independent method is worse than the alternatives for any number of features. Bar Hillel and Weinshall [42] have shown that with a binary detector for each category, using the features selected by the category detector for the category-specific recognition is better than learning individual binary sub-category recognition models independently. However, when we have a $|\mathcal{C}|$-way category-level classifier, our results show that using the best categorization features for recognition does not perform as well as jointly choosing features for the overall task of categorization and recognition.

Even though the relative improvement becomes more modest as more features are added, the problem of determining the optimal allotment of features to the independently trained categorization and identification models remains. An issue with the independent method is that when there is a prior feature bandwidth limitation—as there often is due to time constraints in practical systems or space constraints on portable devices—the optimal feature allotments will undoubtedly depend on the

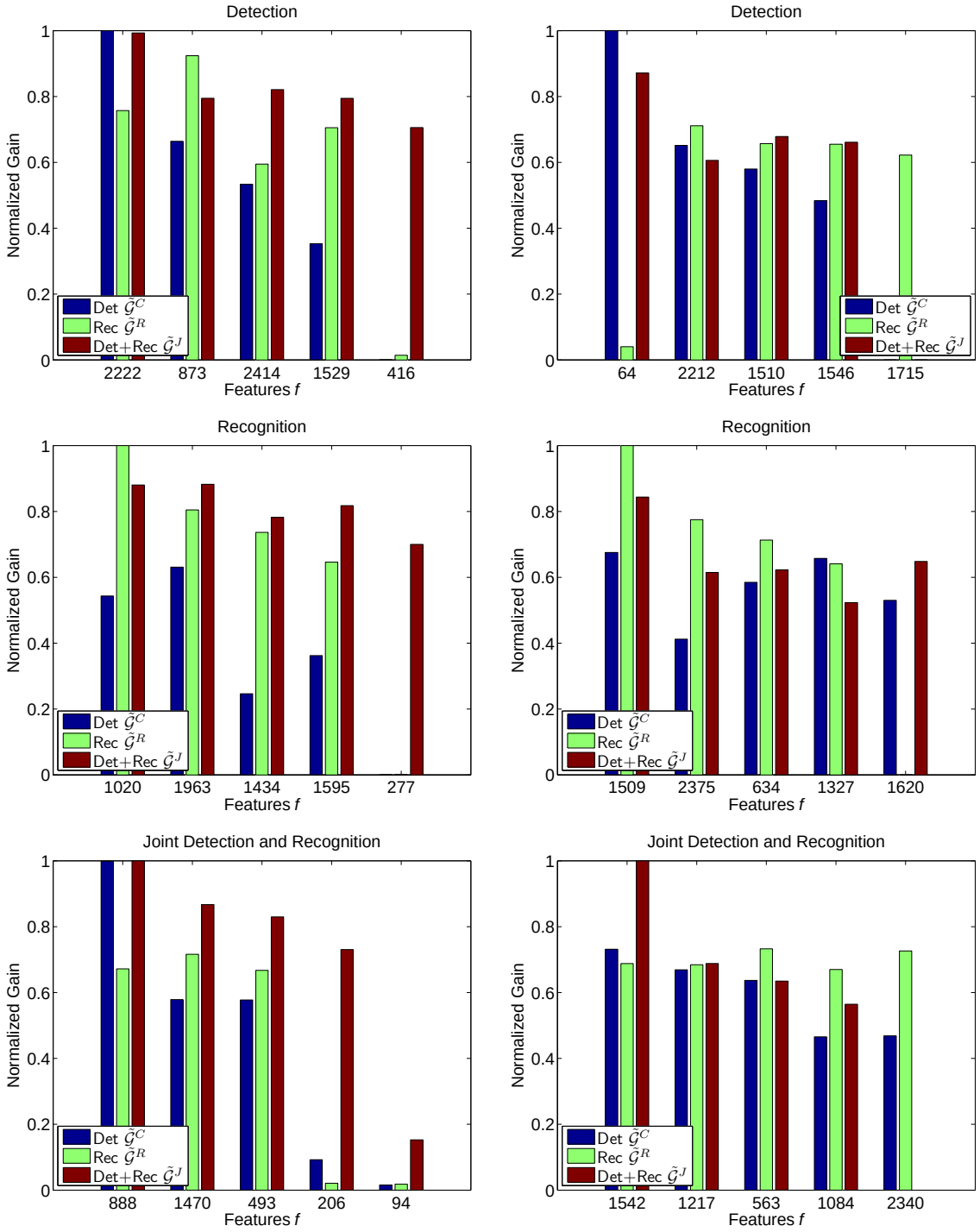


Figure 5.10. Relative gains of features for different tasks during the first (left) and fifth (right) round of feature selection in a two-category experiment with text and background. Each triplet of adjacent bars shows the gain of a particular feature under various selection strategies (objective functions $\mathcal{G}$). The triplets in each graph represent a sample of features (best to worst from left to right) for the particular selection strategy noted above the graph. See text for additional details.

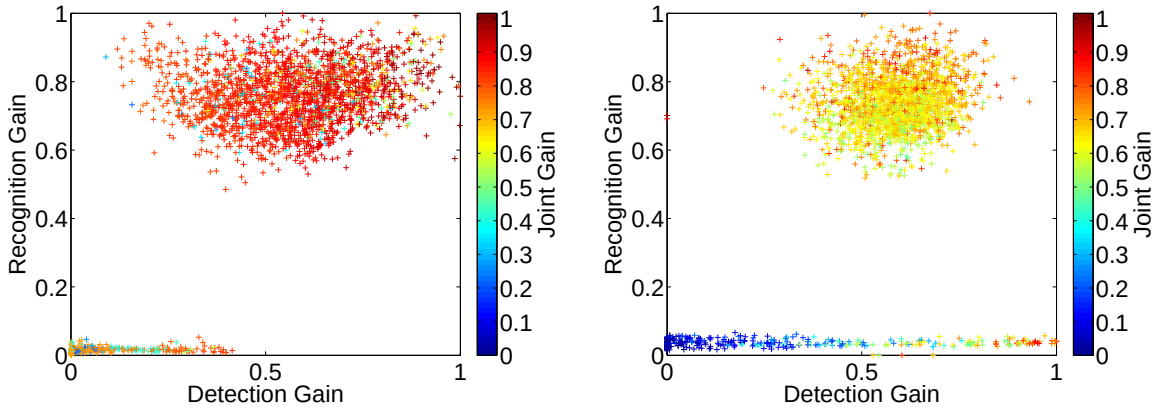


Figure 5.11. Scatterplot of normalized feature gains under three models for the two category (background and text) problem in the first (left) and fifth (right) rounds of feature selection. The gain of a feature for categorization and recognition are shown in a traditional scatterplot, and the color of a point is used to illustrate the factored joint gain.

task. To determine the number of features that should be used for categorization requires an additional level of optimization that the flat and factored methods do not.

The most salient aspect of the factored and flat joint methods' performance is the improvement over the independent method, particularly when there are fewer features available; the accuracy of the joint (flat and factored) feature selection strategies ramp up much more quickly. Even as more features are added the relative improvement of the factored method over the independent remains above 20% (Figure 5.9).

Although the flat joint model is competitive with the factored model for small numbers of features, it begins to level off and diverge from the factored model. The flat joint model is quite cumbersome and takes much longer to train since it has to discriminate between every class label. Consequently, this model is not likely to scale well to more categories and identities; so it is not overly important that we do not yet know its performance for more features.

The bottom-left of Figure 5.7 shows that the flat joint model, which does not explicitly consider categorization, does not necessarily select features that help with the more general task; it performs much worse than the factored model at categorization in the face of limited feature bandwidth or computational time. Note that the factored model, which must consider the subsequent recognition tasks, performs quite comparably to the model that uses only the best categorization features for the categorization task.

On the full categorization and recognition task, we may see the reason for the improvement of the factored over the independent method by considering a feature selected by one model (the best gain for that model) and examining that feature's gain for the other models, as shown in Figure 5.10. In the first round of feature selection, the top categorization (detection) feature is also a very good feature for the factored

joint model, and vice versa. This happens because most instances are background, and it is important to be able to do categorization early. Furthermore, because any feature will likely discriminate among some characters better than no features, the top categorization feature also has modest value (about 70% of the maximum gain) for character recognition. However, by the fifth iteration the best categorization feature often has almost no value for recognition. With this strategy, by the time a character is detected, the features that have been computed will be of limited help in actually identifying the character. By contrast, the first feature selected by the factored joint model has modest value (also about 70% of maximum gain) for recognition, but the second feature selected (not shown in the figure) has high values for both categorization (95% of maximum gain) and character identification (92% of maximum gain). Adding this jointly optimal feature to the factored model thus not only aids in detecting instances of object categories, but very early on the system is also able to identify many more of them as well. When considered independently, however, the best feature for one task (e.g., categorization) is often not as good for another (e.g., character recognition).

The figure also shows, in an abbreviated fashion, the distribution of gains under each model. For instance, by tracing the blue bars in the top-left graph, we see that categorization features have a rapid, almost linear fall-off in gain in the first round; by tracing the green bars in the middle-left graph, we find that the recognition feature gains to do not drop as quickly. The rapid fall-off in the last feature can be explained by the two classes of features in our feature pool and their relative number, as we show next.

There are two distinct clusters in Figure 5.11, due to the two types of available features (texture and template). Along the bottom is a cluster comprised of the texture features with varying gains for categorization. However, these features all have almost no gain for recognition. This makes sense because there would be very little information about character identity in a statistic of wavelet responses over the entire input. Since they are spread along the horizontal axis, there *is* some gain in these features for the text detection (categorization) problem, which makes sense because text exhibits regular textural properties. We also note that there is gain among these features for the joint problem, but this primarily only in the early stages of feature selection where the categorization tends to be more important.

The typical approach to categorization and recognition is sequential. Under such a strategy, the independent categorization model for the two category (text and background) problem selects 20 features before the model posterior plateaus, while the independent character recognizer selects 35 *different* features. For any window detected as text, the detector will have calculated 20 features, and then an additional 35 features must be calculated for recognition. Since the prior probability for text is very small, the total additional computation is modest. However, as the number of object categories grows (as in Figure 1.6 on page 17), the number of queries to category-specific recognizers gets much larger, and the impact of additional feature computation for recognition becomes non-negligible. In a street or office scene, two well-studied applications, there are many regions where generic categorization would find objects of interest such as people, cars, and text in many places and in need

of further special recognition. Having a common feature set among such classifiers with hierarchical categories could greatly reduce computation. Figure 5.8 indicates that for a given performance level, the independently trained classifiers require the computation of more features.

5.5 Contributions

The contribution of this area of research has the potential for wider impact. Whereas nearly all systems for many hierarchical tasks are trained in a segmented or pipelined fashion, we have indicated the limitations of this approach. When one considers how errors compound in a multi-stage pipeline, it becomes apparent that although one can make significant progress in one module, truly solving such problems will involve optimizing the entire chain.

LeCun et al. [65] put this idea to work in their document processing system that was trained end-to-end by complete back-propagation in a multi-stage neural network. Our models and the idea are comparable, but we add the useful notion of probability to all the parts of the model. This allows us to examine sub-components and interpret their outputs meaningfully (i.e., the probability of sign versus background).

The idea of joint feature selection and training has been shown to improve performance in the area of text detection and recognition. Feature selection has long been an important aspect in machine learning as well as visual recognition (cf. §1.4). Indeed, sharing features among classes (e.g., cars, signs, people, faces) for categorization has been shown to reduce the amount of computation and increase generalization [114]. In Figure 1.6 on page 17, this corresponds to a framework of feature selection and sharing "vertically" among the classes at the first level of the tree. Our work contributes a framework for feature selection and sharing "horizontally" along the depth of the tree.

Earlier work by Kusachi et al. [62] treated detection as a recognition problem, much like we do. Their coarse-to-fine strategy performs nearest-neighbor classification, which limits the amount of training data that can be used in a practical system. Our method is parametric, so that the classifier can improve with more data while requiring the same amount of computation. In addition, their method suffers from the same problem as our "flat" model: it does not scale to many more categories.

Our results show that consideration of the entire end-to-end task yields greater accuracy for a given number of features. In a system with limited computational resources, joint feature selection also obviates the need to optimize feature allocation for different tasks.

In more general systems, there will be many detection and recognition tasks. The benefit of multi-purpose discriminative features for these systems should be even larger than demonstrated here. With more complex object classes to detect, pooling knowledge of individual members can help boost detection rates, and having features that are useful for multiple tasks can greatly reduce the necessary amount of computation.

While recent research has focused on developing high accuracy, specialized systems for tasks such as text or face detection and character or face recognition, our results

indicate it may be time to consider returning to frameworks that allow joint training of these powerful new models on broader, end-to-end tasks.

5.6 Conclusions

This chapter has demonstrated that the use of *problem* context can be useful for improving results. That is, adding information about categories (as in the rightmost model of Figure 5.2 on page 80) when training a recognition model can boost accuracy or reduce the computation needed. In the two previous chapters we examined how other kinds of spatial and label contextual influences can improve the results of detection and recognition, respectively, but these types of context were ignored here.

In our next and final chapter, we show how the output of the detection model described in this chapter is combined with a method for recognition that uses the ideas of sparse inference from Chapter 4 to solve the segmentation and recognition problems simultaneously, giving the final output of the system.

CHAPTER 6

THE ROBUST READER

In the previous chapter, we showed that a system could be made faster and more accurate by looking more at the big picture—the end-to-end task of finding and recognizing text. However, the model treated the problem as a simple classification task. Our first model in Chapter 3 showed that detection can be improved by moving beyond a local classifier to one that incorporates more contextual information. Our goal in the final chapter shall be to create a system that synthesizes these ideas. We have modeled several computational aspects of reading, but one that remains is in the gap between detecting text regions and isolating individual characters for recognition. In this chapter, we focus on the steps between the text detection and recognition and address the problem of segmentation of the detected region for recognition. By integrating the segmentation and recognition steps, including word and character boundaries, both bottom-up and top-down information flows influence the process so that low-level commitments are not made too early, but also so that high-level knowledge does not examine unsupported hypotheses.

6.1 Overview

Our previous experiments in Chapter 4 assumed that character and word boundaries had been found prior to recognition. When images are relatively simple, of high resolution, and noise-free, this is a reasonable assumption. However, as the images degrade, these assumptions become problematic. Even if characters can be isolated, when we move beyond document processing into scene text recognition, word boundaries are not as easily predicted by the gaps between characters because the typography becomes somewhat more fluid. As shown in Figure 6.1, sign design is often less constrained by character kerning and tracking conventions. Moreover, before the space between characters can be measured, one must assume that they can be binarized without error. This is a highly unwarranted assumption when noise and low resolution can cause the binarization process to yield both broken and touching characters.

In Chapter 1, Figure 1.5 on page 12 showed the difficulties of segmenting low-resolution characters that result in a single connected mass. Figure 6.2 on the next page reveals the opposite problem: Uninformed binarization algorithms can break characters into multiple segments. Without more information about the characters, a single algorithm or parameter setting is unlikely to correctly binarize all inputs. Thus, we should no longer assume that we can easily find segments that correspond

Figure 6.1. Signs can make prior word segmentation difficult. Examples on the left have a larger inter-character gap than the inter-word spaces of signs on the right.

Figure 6.2. Broken characters from a standard segmentation algorithm. LEFT: Input image. CENTER: Binarized image. RIGHT: Connected components of binarized image.

to entire characters. In this chapter, we present a model that works under both of these circumstances.

We will focus on a common technique that allows the integration of character segmentation with recognition. First, the two-dimensional detected text region is transformed into a one-dimensional representation of the text string. This has the advantage of allowing flexibility and robustness to non-linear text layouts. This representation then allows us to use a standard technique for character segmentation and recognition; namely, a straightforward dynamic programming solution to a parsing/optimization problem. Briefly, such an algorithm examines all possible segmentations of the one-dimensional string and the assignments of character labels to those segments, and finding the optimal segmentation and labeling. Our approach makes use of this standard technique, but we extend it to finding word boundaries.

Previous work has assumed that word boundaries may be easily found. However, kerning and character spacing are often more irregular than in sign typography than typical document text. If word boundaries are known, it is relatively straightforward to force the model to interpret a given word image as a lexicon string and return the highest scoring word [52]. However, when word boundaries are unknown, this method cannot be applied. Therefore, in a fashion that is analogous to examining the hypothesis of starting a character at every location, we model the possibility of beginning a word at every location. To accomplish this, we borrow the sparse inference tools introduced in Chapter 4 to robustly eliminate unlikely word hypotheses over all possible spans or segments.

Our model requires no binarization or prior segmentation. Only a coarse estimate of the baseline is assumed. By integrating recognition with segmentation, at both the word and character levels, we eliminate the strict reliance on uninformed bottom-up techniques, once again preventing us from committing unrecoverable errors.

6.2 Semi-Markov Model for Segmentation and Recognition

Like earlier contextual models for detection (Chapter 3) and recognition (Chapter 4), our model integrating segmentation and recognition will employ compatibility functions that use various sources of information to relate the label hypotheses to each other and the image features. However, rather than using a simple discriminative Markov field model, it will be more natural to use a discriminative semi-Markov model [101]. This model captures not only the interactions between states in a sequence labeling problem, but also the duration of a particular state along the sequence. Thus, in our problem, segmentations are given by the duration of the character states along the sequence. In practice, the optimization problem of finding the most likely sequence of characters under the probability distribution can be solved using a dynamic programming algorithm.

The components of our model are analogous to the compatibility functions defined in previous models, except the probability distribution is now over all possible segmentations and their labels, rather than the labels of a given segmentation. Whereas the previous definition of $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}, I)$ was a normalized sum of exponentiated compatibility functions on unknown labels y_i, our new probability distribution will involve functions on labels *and* segments. Our goal will be to find the most likely segmentation and labeling. We will describe this process in two parts. First, we discuss how a given segmentation and labeling is scored, which involves describing the sources of information (and thus, compatibility functions) being used. Second, we discuss how to find the optimal (or nearly optimal) segmentation and labeling, which is a dynamic programming problem. All of this assumes a one-dimensional segmentation or sequence labeling problem. The next section (§6.3) will discuss how the two-dimensional image is transformed into a one-dimensional representation for segmentation.

6.2.1 Segmentation and Recognition Model

In this section we describe how a given segmentation and labeling is scored. The same basic sources of information used earlier for recognition will go into the new model that also parses the string into segments. Compatibility functions representing appearance, local language, and lexicon information are all used. Additional functions capturing layout information such as character bounding box overlap and inter-character gaps are also employed.

Our previous models assumed a segmentation, and thus a set of unknowns for which the most likely labeling must be found. In this model, a segmentation will induce the unknowns and corresponding compatibility functions. The example in Figure 6.3 shows one segmentation, or parse, of a text string. In this parse, there are five regions corresponding to character hypotheses that must be given labels. Notice in particular that one of the parse regions, y_4, corresponds to the space character, $\sqcup$. Modeling spaces explicitly as a character to be recognized will allow us to seamlessly integrate word boundary detection with character recognition and segmentation. Since, in addition to labelings, different segmentations must compete with

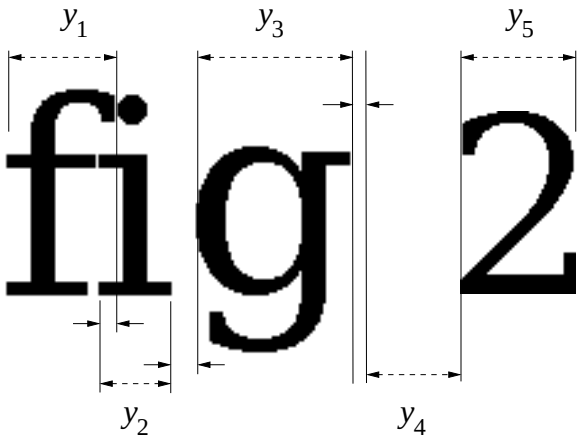


Figure 6.3. A segmentation of a string of text into five primary regions (indicated by the dashed lines) that can be scored for various labelings.

each other, the figure also contains regions (marked with solid lines and arrows) that correspond to properties of the parse itself. Namely, these are overlaps of character regions (as for y_1 and y_2) and gaps between them (as for y_2 and y_3).

When the model includes lexicon information, there will be another set of unknowns induced by a particular segmentation. These will correspond to an indicator of whether a particular chunk of the segmentation corresponds to a lexicon word or not. Within such regions, a different language model is used. The details of this will outlined in the next section.

We use five basic classes of terms in calculating the score of a particular parse. These will correspond to character appearance, character segment overlap, inter-character gaps, character bigrams, and a lexicon bias. We detail these terms in the following paragraphs.

6.2.1.1 Character Appearance

Each span or segment is scored for a particular character and span width by a learned compatibility function. As in previous models, these functions are linear energies:

$$U^A_{r,t}\left(y, \mathbf{x}; \boldsymbol{\theta}^A\right) = \boldsymbol{\theta}^A_{r,t}\left(y\right) \cdot F_{r,t}\left(\mathbf{x}\right). \tag{6.1}$$

The linear parameters $\boldsymbol{\theta}^A$ used in the dot product are dependent not only on the character identity y, but also the size of the span (calculated from r and t). The image features $F_{r,t}(\mathbf{x})$ used are extracted using the hypothesis that the character is centered in the span from r to t. The discriminative nature of the model allows us to use portions of the image outside the span without violating any independence assumptions. For example, using greater image context could help disambiguate some characters.

6.2.1.2 Character Bigrams

As in earlier chapters, local language information can be easily used in the form of bigrams. Each pair of neighboring character spans is given a bigram score from a linear energy:

$$U^B\left(y', y; \boldsymbol{\theta}^B\right) = \boldsymbol{\theta}^B\left(y', y\right). \tag{6.2}$$

6.2.1.3 Character Overlap

A pair of neighboring spans may either overlap or have a gap between them. In the case of overlap, a simple energy term is added:

$$U^O_{n,r}\left(\boldsymbol{\theta}^O\right) = \theta^O_{n,r}, \tag{6.3}$$

depending on how many pixels overlap—from n to r—between the spans. Using this information, we allow character bounding boxes to overlap (as in the example `fi` ligature of Figure 6.3), but the degree of overlap allowed is soft and flexible.

6.2.1.4 Character Gap

As stated above, a pair of neighboring spans may also have a gap between them. For instance, in Figure 6.3, the character `i` is separated from `g` by a few pixels.) In this case, the gap is scored by a learned compatibility function

$$U^G_{n,r}\left(\mathbf{x}; \boldsymbol{\theta}^G\right) = \sum_{i=n}^{r} \boldsymbol{\theta}^G \cdot F_i\left(\mathbf{x}\right), \tag{6.4}$$

which is a sum of the scores for each index (column) considered to be a gap. Similar to the character appearance model, image features from and neighboring the index i are used with learned linear parameters to score the compatibility of the hypothesis that the span from n to r is a gap.

6.2.1.5 Lexicon Information

Our model also features a parameter that will allow a bias for character sequences that compose a lexicon word,

$$U^W\left(\theta^W\right) = \theta^W. \tag{6.5}$$

When calculating the total score (as detailed in the next section), this term will only be included in portions that are part of a lexicon word.

6.2.2 Model Inference

The recognition task can be thought of as finding the segmentation and corresponding labeling that maximizes a total (summed) score. The constituents of this score—the exponent of the exponential model—were described in the previous section. Here we describe how to find the segmentation and labeling that gives the best overall score. The inference process can be accomplished in a model with or without the use of a lexicon. Omitting lexicon information is simpler, so we begin by describing a model without it.

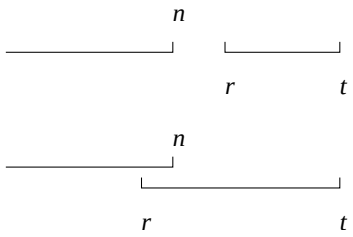


Figure 6.4. Illustration of dynamic programming variables n, r, and t, and their ranges from character gap (top) to character overlap (bottom).

6.2.2.1 Lexicon-Free Parsing

We can build up a two-dimensional dynamic programming table that finds an optimal parse. Let $S(t, y)$ be the optimal score for a span ending at index t with character y. If $t = 1$ is the first index of the sequence, we can iteratively build up the table via the following recurrence relation

$$S(t, y) = \begin{cases} \max_{n,r,y'} S(n, y') + P(n, r, t, y', y, 0), & t > 0 \\ 0 & t = 0 \\ -\infty & t < 0 \end{cases}, \tag{6.6}$$

where $P(n, r, t, y', y, 0)$ represents the additional parse score for adding a segment that starts at r, ends at t, and consists of character y, while the previous character y' ended at n. The last argument 0 indicates that the additional character is not forming part of a lexicon word. The maximization occurs for acceptable ranges of n and r, based on the limits of character gaps and overlaps. Figure 6.4 illustrates how n and r vary and relate to the segments for a given t.

The additional parse score P is thus composed of the energy terms detailed in the previous section. It is very important not to bias the total score for parses with differing number of segments. Therefore, we add simple multiplicative weights that assign the character appearance and bigram energies to every index point along their respective spans. Weight $W^A(r, t)$ is simply the width of the character span, while $W^B(m, n, r, t)$ assigns the bigram energy U^B to every index point along the two spans, from from the start of the first span (m to n, covered by y') to the end of the second span (r to t, covered by y).[1] The gap score does not require such bias prevention because each component of a gap is scored individually, rather than as a unit. The total additional parse score is

$$\begin{aligned} P(n, r, t, y', y, w) &= U^A_{r,t}(y, r, t) * W^A(r, t) \\ &\quad + (1 - w) * U^B(y', y) * W^B(m, n, r, t) \\ &\quad + w * U^L * W^B(m, n, r, t) + U^O_{n,r} + U^G_{n,r}, \end{aligned} \tag{6.7}$$

where m is the beginning of the previous span. The total score for a given segmentation and labeling is the sum of all the P terms, which would be the value inside the

[1] Bigram scores get double-counted on every index except those for the first and last characters; we correct this bias as well in our implementation.

exponential of the corresponding Markov probability model. Since we are only finding the most likely parse, the normalization term Z does not need to be calculated.

We note that there are only character scores for segments from r to t that correspond to a set of possible quantized character widths. The final, optimal parse can of course be found by storing the values of n, r, and y' that yield the maxima at each step and tracing them back to the beginning. We especially note that no word segmentation must be done; it is automatic since a space is among the characters to be recognized.

6.2.2.2 Lexicon-Based Parsing

When incorporating a lexicon (and thereby allowing a bias for strings to be recognized as lexicon words), the dynamic programming problem becomes more complex. First consider the case of a one-dimensional segment that is assumed to be just one word. In this case, we simply constrain the dynamic program to find parses corresponding to a given lexicon word, and then find the word that scores the highest. The dynamic programming can be interleaved with a trie traversal to speed the processing of this type of parsing by reusing information[74, 52].[2] For every leading substring of the lexicon, the optimal score is stored for that substring parse ending at t. Thus, leading substrings shared among words do not need to be re-parsed for each word.

In practice, we cannot assume that the array we are trying to recognize contains just one word, and this complicates things significantly. A word may start or end at any point in the line. To handle this, we will maintain a parallel set of dynamic programming tables, one for non-lexicon parses like before, and another using the lexicon.

A graph representation of the finite state machine for an example can be seen in Figure 6.5. Every arc junction represents the ending index t, and the dynamic programming maximum is calculated over all the incoming arcs. The maximum may be either from the characters (bottom) governed by a bigram language model or from the lexicon-constrained words (top).

Like before, one table is built by optimally adding characters using the parse score $P(n, r, t, y', y, w)$ and corresponds to the best parse for a particular character y ending at point t. These are the incoming arcs from the bottom half of the graph in Figure 6.5. Our other dynamic programming table $W(t)$ corresponds to the best score for a lexicon word ending at t. These are the incoming arcs rom the top half of Figure 6.5. The important thing is how these two tables are related. The optimal character given by $S(t, y)$ may have been preceded by a lexicon word, and the optimal *word* calculated by $W(t)$ may have been preceded by a character that is not from a lexicon word. The two are linked via dependent recurrence relations.

With a lexicon, the primary recurrence for $S(t, y)$ has two cases. When $y \neq \sqcup$, the parse score is given by Eq. (6.6), the sum of the best previous score plus the score for adding the new segment. When the new y is a space, the end of a character string

[2] A "trie" is a prefix tree in which a character is associated with every node, and the descendents of any node have a common prefix given by the values on the path from the root (the empty string) to that node.

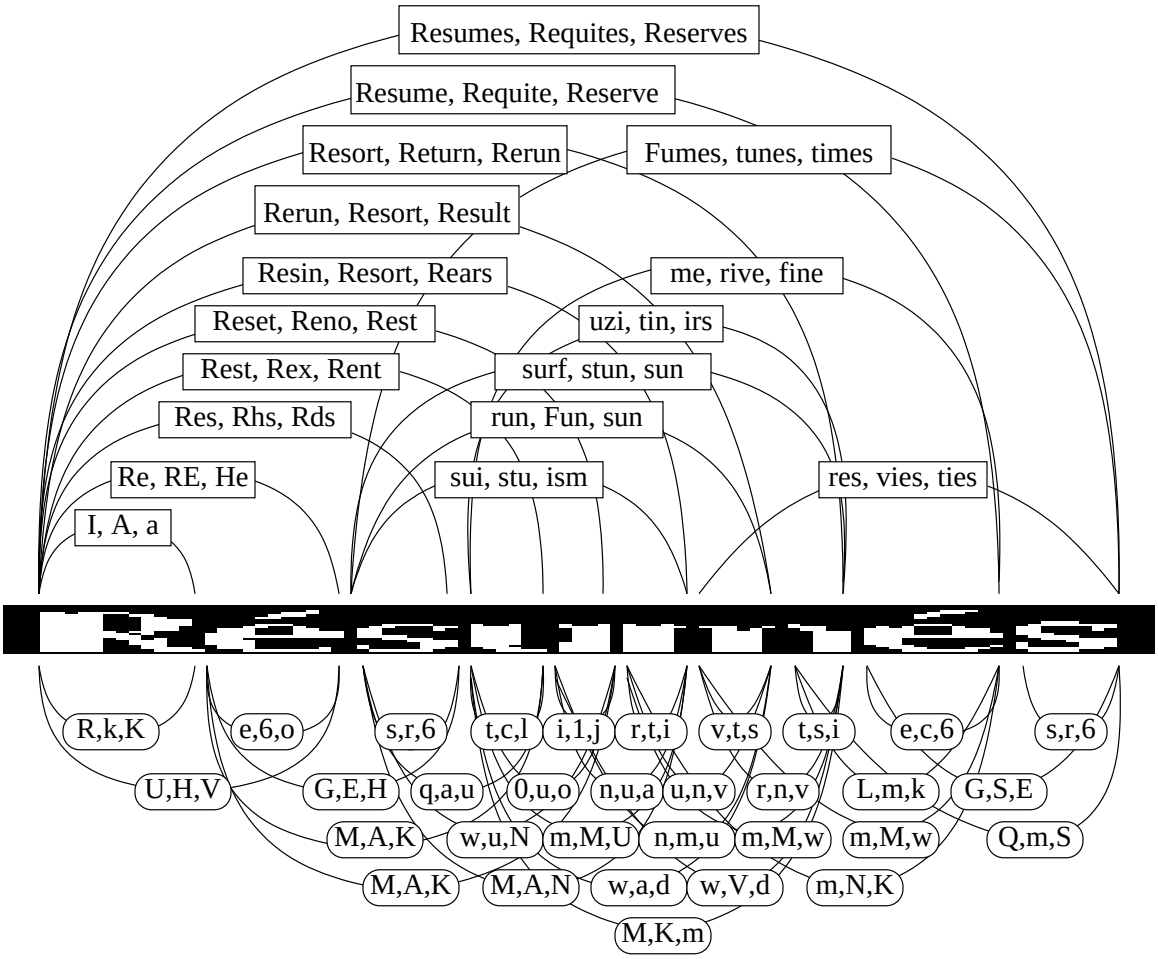


Figure 6.5. Partial finite state word graph for recognition and segmentation. Top (square) boxes contain the top three lexicon words for a span, and the bottom (rounded) boxes contain the top three characters for a small span for a lexicon-free parse. Crossing over from characters to words or going from one word state to another implies a space was parsed. Spaces are optional between characters.

is signalled, and this string may either be a lexicon word or not. In this case, the dynamic program must determine whether the optimal parse is to accept the previous lexicon word S_ℓ or to take the non-lexicon parse ending before the space $S_{\overline{\ell}}$:

$$\begin{aligned} S(t, \sqcup) &= \max\left\{S_\ell(t), S_{\overline{\ell}}(t)\right\}, && (6.8)\\ S_\ell(t) &= \max_{n,r} W(n) + P(n, r, t, \widehat{y}_n, \sqcup, 0) && (6.9)\\ S_{\overline{\ell}}(t) &= \max_{n,r,y'} S(n, y') + P(n, r, t, y', \sqcup, 0) && (6.10) \end{aligned}$$

where $\widehat{y}_n$ is the last character of the word parse ending at n and all other terms are as before.

The lexicon word-based relation is similar to the original table $S(t, y)$, but with an extra layer of complexity:

$$W(t) = \max_n S_\ell(n, \sqcup) + B(n, t). \tag{6.11}$$

Thus, the score $W(t)$ builds upon the previous scores S_ℓ, but adds an additional term $B(n, t)$ that corresponds to the optimal score for any lexicon word ending at t, with the previous portion of parsed text ending at n with a space. We must now say how this term is calculated.

The lexicon word score B is calculated very similarly to the original S of Eq. (6.6). However, instead of allowing any arbitrary sequence of neighboring characters $y'y$, the string is constrained to be a particular lexicon word. Thus, for the kth lexicon word, we can define a score for a parse of the word up to the ith character, y_i^k, that ends at location t, where the beginning of the word is preceded by a space that ends at n:

$$C(n, t, i, k) = \max_{m,r} C(n, m, i-1, k) + P\left(m, r, t, y_{i-1}^k, y_i^k, 1\right). \tag{6.12}$$

As before, the term P is a score for a parse of a particular character including the appearance model, and gap/overlap scores. However, the language model is altered since the character is now part of a (hypothesized) lexicon word. The score for neighboring characters that are constituents of a lexicon word thus replace the bigram score entirely, as indicated by the argument $w = 1$. Now, the $B(n, t)$ term of Eq. (6.11) is the highest valued complete parse of all lexicon words over the span from n to t,

$$B(n, t) = \max_k C\left(n, t, \left|\mathbf{y}^k\right|, k\right), \tag{6.13}$$

where $i = \left|\mathbf{y}^k\right|$ is the length of the kth word.

6.2.2.3 Computational Complexity

This approach begets a problem of scale. Although the complexity is linear in the number of lexicon words and the length of the component to be parsed, we have proposed a method that maintains a hypothesis of every possible word beginning at every possible location. This cross product of possibilities is impossibly slow in practice, so therefore approximations must be introduced.

The complexity of the lexicon-free model is easy to analyze. It is $O(rnct)$, where

$$\begin{aligned} r &\equiv \text{maximum character width} \\ n &\equiv \text{maximum gap} \\ c &\equiv \text{size of alphabet} \\ t &\equiv \text{length of input.} \end{aligned}$$

In practice, all are essentially constant parameters of the model (not the input), with the exception t, but it is important to consider them in contrast with the lexicon-based model. A simple worst-case analysis gives a complexity of $O(rnuwt)$, where terms are as above, adding

$$\begin{aligned} u &\equiv \text{number of nodes in lexicon trie} \\ w &\equiv \text{maximum word width.} \end{aligned}$$

Once again, u and w are really built-in properties of the model that do not vary with the input, so they are "constants" in the final analysis. However, they end up being rather large constants that are non-negligible in practice. The term u is the number of unique leading substrings of lexicon words. For two words that share the same leading substrings, the optimal parse of that substring can be shared. This sharing reduces the potential complexity in the lexicon we use by 75%—the number of nodes in the lexicon trie is roughly 25% of the number of characters in the lexicon. The uw term (which corresponds to and greatly dominates the c of the original model) indicates that for every location t, an optimal parse of every subword is calculated, up to the maximum word width. Of course, there are some span widths for which a subword is too short, due to character width and gap limit, and some span widths for which a subword is too long, due to character width and overlap limits. Thus, this simple analysis fails to account for these, but the picture is still clear—without some sort of approximation, this method is impractical, because $uw \gg c$. In the next section, we outline how, for a given span (up to the maximum word width), we can limit the number of nodes in the lexicon trie that are parsed.

6.2.2.4 Sparse Lexicon-Based Parsing

To keep the number of hypotheses small, we eliminate words from consideration based on the relative score of all subword parses over a particular span. Perhaps the most natural view of the table $C(n, t, i, k)$ is as the score for the subparse of all words, where the word k is fixed and the segmentation locations n and t are varied. This is most natural when we are seeking the best segmentation for a given word, which is the view promoted by Equations (6.11) and (6.12). However, the table is not only a competition among parses for a given subword, but it may also be viewed as a competition among subwords for a given span. This view considers fixing n and t while the subwords vary with i and k. This view allows us to eliminate from further consideration unlikely subwords on the span from n to t.

For a fixed span from n to t, the term $C(n, t, i, k)$ then represents the energy in an exponential model, where the optimal subword parse varies with the arguments i

and k. This joint space over i and k represents every possible sub-word parse for the span, most of which are extremely unlikely. To speed the processing that is necessary to eventually find the optimal region word parse $B(n,t)$ calculated from C, we would like to eliminate these unlikely candidates.

Such pruning is a form of beam search common to such dynamic programming algorithms for speech recognition [49]. In Chapter 4, we used a divergence bound to prune individual character hypotheses for lexicon-based recognition when segmentations were given. Here, we will use the same idea to prune unlikely subwords. Using the optimal parse score, we define the probability distribution over lexicon subwords as

$$p(i,k \mid n,t,\mathbf{x},\boldsymbol{\theta},I) \propto \exp\{C(n,t,i,k)/T\}, \tag{6.14}$$

where $T = n-t+1$ is the size of the span. This scaling factor is important to capture the "average" score for each index/pixel, otherwise the cumulative differences between scores is extremely large for very long spans, tantamount to a delta function for the probability distribution.

With (6.14), we have a few options for eliminating some subwords from further consideration in an approximate Viterbi beam search. For a given n and t, every entry for i and k that is eliminated further eliminates all words that build upon that subword parse and no longer need to be considered. With fewer calculations of the subword score (6.12) to be made, there are fewer complete words to score (6.13). This is essentially a greedy breadth first beam search strategy. Our options include the information-theoretic KL divergence criterion used in the lexicon-based recognition of Chapter 4, a simple N-best list, or a threshold on the ratio of a candidate and the highest probability for a region. These are subsequently described in more detail.

KL Divergence Beam We may assign a distribution $q(i,k)$ with a maximal number of zero entries, subject to a bound on the KL divergence between p and q (cf. Equation (4.15) of Section 4.2.4.2 on page 60). While the scores C do not change, the non-zero entries of q constitute the beams that continue to be explored.

N-Best List A standard strategy for Viterbi beam search is to sort the scores C for a given region (n and t), keeping the N best subword parses (i and k).

Probability Ratio A common dynamic alternative to the N-best list is to threshold the ratio of the best score for a region and an alternative candidate,

$$\frac{p}{\widehat{p}} > \tau, \tag{6.15}$$

where $\widehat{p}$ represents the highest value of the probability (6.14) for a fixed region, specified by n and t, and p is an alternative value of i and k for the same region.

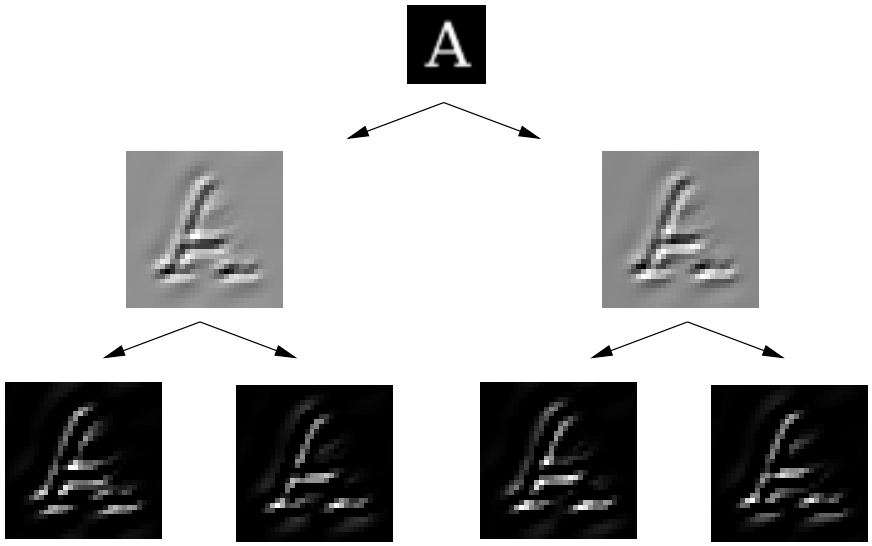


Figure 6.6. A character image (top) is transformed by the steerable pyramid, with even and odd outputs of one orientation channel (middle) each rectified into positive and negative components (bottom).

6.3 Features and Pre-Processing

In this section we describe in some detail the image features that are used for recognition, as well as the processing that transforms a detected image region into a one-dimensional representation suitable for our dynamic programming algorithm.

6.3.1 Image Features

As in earlier chapters, we use the steerable pyramid as the basis of our representation. However, we are now doing segmentation, and image phase information will be important so we no longer use the magnitude of the quadrature pairs, but instead use a rectification step that facilitates better performance in the linear model (6.1). This rectification separates positive and negative values of each of the even and odd filters into separate components. That is, each original value x is replaced with two transform functions, f^+ and f^-:

$$\begin{aligned} f^+(x) &= \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \\ f^-(x) &= \begin{cases} -x & x < 0 \\ 0 & x \geq 0 \end{cases} . \end{aligned} \tag{6.16}$$

Example filter outputs and rectified versions are illustrated in Figure 6.6. At each window being recognized as some character, these four features are used as the vector $F_{r,t}(\mathbf{x})$ of Eq. (6.1).

Since the images are of varying contrast and brightness, we must normalize the filter responses. We use the same strategy described in Section 5.3, reviewed here. At

each location, all the rectified filter response values for every pixel in an $a \times a$ window surrounding the location are normalized to a unit ℓ_2 norm. After normalization, the responses are clipped at a threshold (0.2 in our experiments), and the same values are re-normalized. These re-normalized values for each filter response are then kept for the center location. The normalization strategy implies that a different window is used to normalize each location.

6.3.2 Pre-Processing

Approximate text baselines can be detected using the classifier described in the previous chapter. At every pixel in an image, a detector gives the probability of the pixel being centered on a character of a particular scale (font size). This yields a set of windows (one corresponding to each detection) containing text of a (roughly) known size. Furthermore, because the characters in the training data have a baseline located at a certain row of the window, the baseline is roughly known as well. No top down information is fed back to the classifier, so the detection probabilities may be thresholded to yield the window candidates.

At each pixel detected as text, a probabilistic character classifier outputs a score for a combined character identity and approximate width at that location. In addition to the 62 characters, the score includes a "space" character as well (along with the space width). This is important for both bigram modeling and word segmentation. The classifier also includes a "gap" class that is not an intra-word (i.e., ASCII) space but is simply a result of the type layout (e.g., kerning).

We assume that we are recognizing a one-dimensional string of text, but not that it is necessarily horizontal or even linear. Because the detections generate a text region, and the characters may not even necessarily have a true base "line," we must transform the set of character scores (which may be thought of as residing at each detected pixel), into a one-dimensional representation. Roughly, this proceeds as follows:

- Horizontally connect any disconnected areas (generally regions with intra-word spacing larger than the detection window size)
- For each resulting connected component
  - Find the unique set of columns for all pixels from the component (assuming a horizontal orthography)
  - These columns become indices into a one-dimensional array representation of the recognition problem for that component
  - For each column in the component
    - Assign the score for each hypothesis (character, width, space, gap, etc.) to be the maximum value of that hypothesis over the component's rows for that column

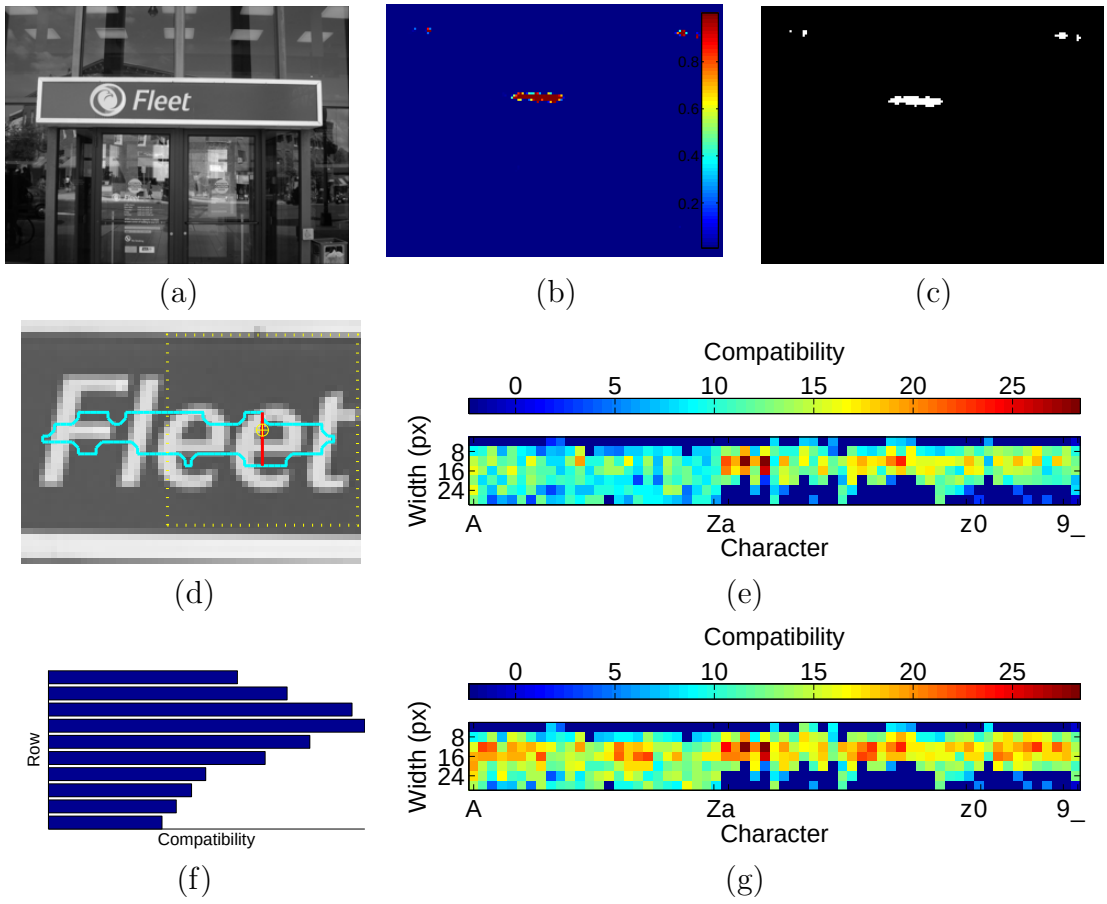


Figure 6.7. An image (a) is put through a text detector that outputs the probability of text centered at each pixel (b), which is subsequently thresholded (c). Each pixel in the detection region (d), with center pixels outlined by solid cyan contour, is used as the center for a character recognition score. Here, the pixel marked with a yellow $\oplus$ in (d) is the center of a dashed yellow box that is input to the compatibility function Eq. (6.1), yielding a score for each character and quantized width (e). For the column marked with a solid red line (d), the score for a particular hypothesis (here, character e and width 12 pixels) is measured over each row (f), and the maximum compatibility is used as the score for the hypothesis in that column. The maximization result for all hypotheses in the column are shown in (g).

This procedure is outlined visually in Figure 6.7.

Next, we perform a course binarization of the image in each connected component mentioned above. This step is not strictly necessary, but it speeds recognition by limiting the number of cut points in the segmentation. We use a variant of Niblack's binarization algorithm, which uses a local threshold at each pixel. The threshold is determined by the mean plus some fraction of the standard deviation of a window centered at the pixel. This assumes the foreground is lighter than the background. We currently provide the text polarity to the system, but this analysis is fairly straightforward to achieve automatically. The fraction of the standard deviation used to calculate the local threshold is the same across all images, except in the following case. If the resulting binarization yields a hypothesized binary character component that is wider than the assumed maximum character width, the threshold is (recursively) increased *for only that component* until it yields components that meet this width criterion. Finally, these character components are examined, and the start (leftmost column) and end (rightmost column) of each component are used as anchors for *possible* cut points in our one-dimensional segmentation problem. Note that we assume there is an over-segmentation of the characters, so that components may be combined, but not split. This makes solving the dynamic program faster because fewer options for n, r, and t need to be considered, while also solving the problem of split binarized characters highlighted in Figure 6.2.

We now have a one dimensional problem where for each span or segment of an array, there is a score, $U^A_{r,t}$, for that span as a particular character—assuming the span corresponds to one of the quantized widths of the character classifier. There is also a score, $U^G_{n,r}$ at each array entry for considering it as part of a intra-word/inter-character gap.

6.4 Experiments

Here we present experimental validation of our model and inference techniques on our sign data. First, the training and evaluation data are described, and then we detail the procedures used for both model training and evaluation. We conclude with the experimental results, analysis, and discussion.

6.4.1 Experimental Data

Training and testing data are drawn from the same sources as in Chapter 4, with a few minor modifications.

Sign Evaluation Data 10 of the 95 regions are eliminated because they contain punctuation characters that our classifier is not currently trained to recognize. Furthermore, the size of the training and recognition text is made more difficult, being scaled to a 25 pixel font size (roughly 12.5 pixel x-height).

Synthetic Font Training Data The 62 characters are rendered in 934 fonts and left-right centered in a square window of size $3 * (128 \times 128)$. As in the data for

Figure 6.8. Synthetic font training data. TOP: Full images with borders used for calculating wavelet features and contrast normalization. BOTTOM: Final windows used for training.

Chapter 5, neighboring characters were sampled from bigrams learned from our English text corpora and placed with uniform random kerning/tracking. No distortion transformation was applied, but the same random contrast, brightness, and Gaussian noise are added. In addition, we randomly add solid borders neighboring the text, with an appearance probability of 0.7 (0.1) for a top/bottom (left/right) edge, with the location sampled from $l \sim \text{Beta}\,(\alpha = 2, \beta = 3)$ and scaled from the range $[0, 1]$ up to the distance from the image edge to the character. We also subject the resulting image to an additive linear bias field with random slope $m \sim \exp\,(\lambda = 1e4)$ pixels and uniform direction angle. As mentioned above, the images are scaled down by a factor of four, for a font height of 25 pixels. The proper amount of Gaussian blurring was performed prior to downsampling to avoid aliasing. After the features are calculated and normalized (as described in §6.3.1), the character images are cropped to the center 32×32 window. Example images are shown in Figure 6.8. Space and gap images were generated by choosing a space or gap width, and then randomly selecting neighboring characters from a given font, which are placed along the baseline with the appropriate distance so that the gap is centered.

Text Corpora The same text is used as in Chapter 4.

Lexicon Because our model does not account for word frequency in its bias, we reduce the lexicon to words up to the 50th percentile, including only proper names and upper case words in addition to the standard list. Furthermore, since there are many two letter words that do not seem to us to warrant inclusion

(e.g., `cs`, `Bk`, `Kr`, `Nb`, `pd`, `Tl`), we only use these up to the 10th percentile of frequency.

6.4.2 Experimental Procedure

In this section we outline the procedures used for training and evaluating our model.

6.4.2.1 Model Training

The model parameters $\boldsymbol{\theta} = \left[\begin{array}{ccccc} \boldsymbol{\theta}^A & \boldsymbol{\theta}^B & \boldsymbol{\theta}^O & \boldsymbol{\theta}^G & \boldsymbol{\theta}^W \end{array} \right]$ are learned from the data outlined above or established by a simple manual procedure. While typical parameter estimation procedures in joint discriminative models require labeled data that includes all the information at once, we use the decoupled and piecewise training methods described earlier to learn the compatibilities individually. The only exception is that $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^G$ are estimated together, since they involve using the same features to explain competing hypotheses for the same regions.

Appearance and Gap Model The parameters $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^G$ are trained on 934 fonts. The set is split in half first so that cross-validation may be used to select the value of the hyperparameter α for the Laplacian prior, as in the appearance model of Chapter 4. We quantize the width of training characters (and subsequent parse spans) to the values $\{4, 8, 12, 16, 20, 24, 32\}$ pixels. Space characters have all but the smallest and largest widths, and training gaps are 1-3 pixels wide, although only features from the center column of pixels is used in the compatibility.

Bigram Model As in Chapter 4, we use piecewise training for estimating the bigram model weights $\boldsymbol{\theta}^B$. The signs in our data tend to occur in all-capital letters more frequently than text in our corpora. For that reason, we train a case-insensitive bigram model for same-case transitions (so that the weight for `AB` is the same as `ab`). The case-sensitive values are used for all intra-case transitions.

Overlap Model The overlap weights are a truncated quadratic. Since the character widths are quantized in four pixel increments, the largest error in this quantization is 2 pixels on each side. Thus, we allow a two pixel overlap without penalty. The quadratic coefficients were established so that the penalty would be at a scale comparable to that of a fraction of the appearance compatibilities (roughly one-third for the largest overlap of seven pixels).

Lexicon Model We set lexicon intra-word character transition weight θ^W to twice the median bigram weight for all bigrams over words in our lexicon.

6.4.2.2 Model Evaluation

To focus testing on the ability of our model to recognize text, we manually provide rough detection windows corresponding to those that would be provided by the text detector outlined in Chapter 5. This eliminates egregious false positives and false negatives, but is by no means a perfect localization. Because the baseline is only approximately and imprecisely specified, we must still perform the score maximization that is part of the linearization described in Section 6.3.2.

Using 85 text regions consisting of 183 words and 1144 characters we report the character recognition error of our system using the Levenstein edit distance. The word error is given by the ISRI OCR performance tookit (OCRtk) program `wordacc`[3]. The program ignores number strings, however, so we manually fix the program output for the five signs in our data containing numbers.

We compare the results of our model to those of commercially available OmniPage 15 software. Both the original image and a binarized version that is the result of Niblack's binarization algorithm (described above) are input to the software. Our model can be operated in an "open vocabulary mode" without a lexicon, in which case only Equation (6.6) is used for inference. Alternatively, a "closed vocabulary" mode forces words to be from the lexicon if we ignore the non-lexical parses. The mixed vocabulary mode is the full model presented above, and it recognizes words both in and out of the lexicon.

Upon inspection, some of the signs in the evaluation data contain fonts that, though normalized for font size (e.g., height), otherwise have character bounding box aspect ratios that are not present in our training data; they are either narrower or wider. Ideally, we would have such fonts in our training data. As a compromise, we hypothesize an unknown aspect ratio. Running the recognition/parser at five different horizontal scales, $\left\{\frac{1}{2}, \frac{\sqrt{2}}{2}, 1, \sqrt{2}, 2\right\}$, we keep whichever has the highest final score $S\left(t_{\mathsf{end}}, y\right)$.

6.4.3 Experimental Results

Figure 6.9 evaluates the utility of the various beam search strategies, displaying the error versus average recognition time (per horizontal pixel). Each strategy has a different parameter (ϵ for KL divergence, N for N-best, and τ for the probability ratio), and these parameters are varied to increase or decrease the size of the beams during search. As the beam size increases, more options are considered and it is less likely that a correct word will have been dropped prematurely, but it will take more processing time to maintain the list of word candidates.

The overall results are collected in Figure 6.10. The mixed vocabulary mode, drawing on words both in and out of the lexicon, yields roughly a 13% error reduction over open and closed vocabulary modes. For best performance, the commercial software requires the input to be binarized before being processed. When we optimize for the unknown aspect ratio of the font we reduce our error by nearly 15%, which is

[3] `http://www.isri.unlv.edu/ISRI/OCRtk`

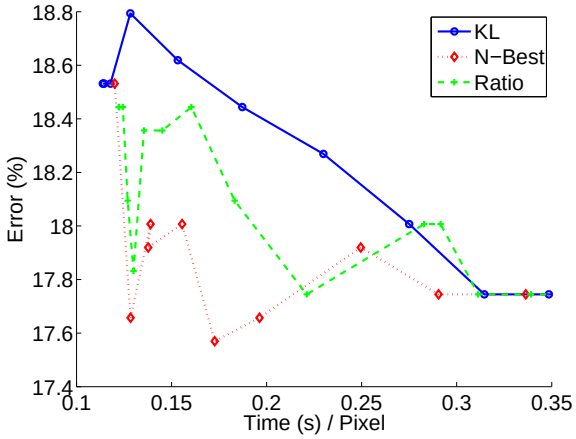


Figure 6.9. Comparison of beam search strategies as ϵ, N, and τ are varied.

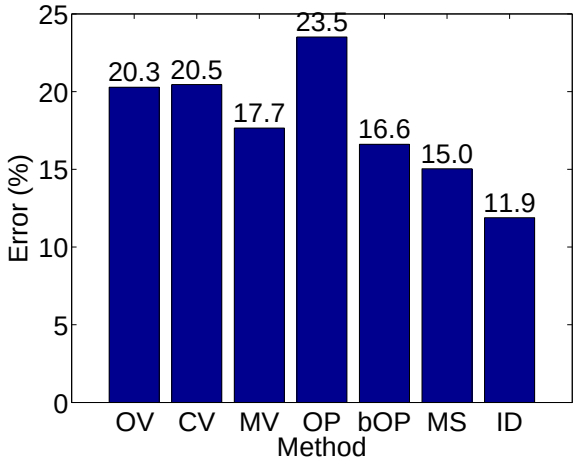


Figure 6.10. Comparison of recognition methods: our model's open vocabulary (OV), closed vocabulary (CV), and mixed vocabulary (MV) modes; OmniPage software results on image input (OP) and binarized input (bOP); our multiscale approach (MS) and hypothesized ideal (ID).

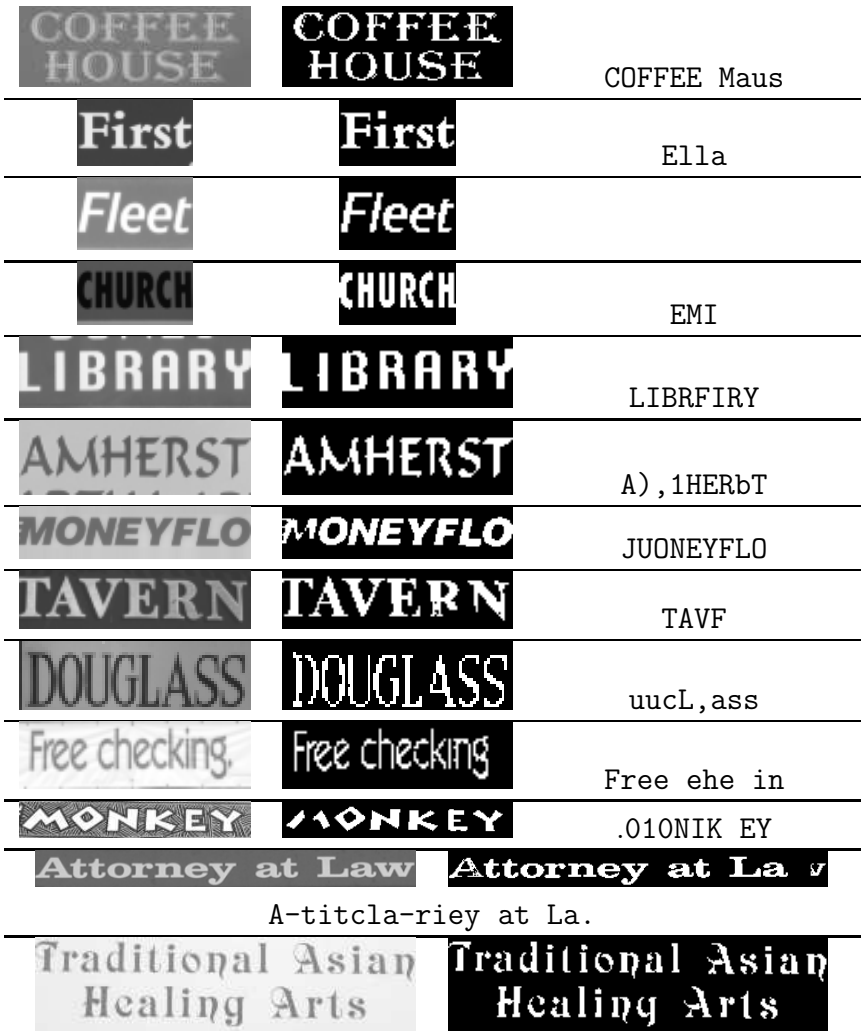


Figure 6.11. Example images correctly read by our model and the corresponding binarization and OmniPage output.

also a 10% error reduction over OmniPage. This approach to handling narrow and thin characters does indeed fix many errors, but it also introduces some when the highest scoring parse occurs at a scale other than 1. In the ideal case, if adding such characters to the training data only improved the results and did not make them worse, our error rate would drop by 33%.

Examples that our model reads correctly, along with the corresponding binary input to OmniPage and its output are in Figure 6.11. We also probe the behavior of our model in the face of lowered resolution, as shown in Figure 6.12. Two input images are subject to low-pass filtering before being recognized by our system. Similarly, we binarize these blurred images before passing them to OmniPage. In this case, the binary images are not used to limit anchor points for segmentation in our model—all indices are considered as cut points in the blurred images.

σ	Image	Output	Binarized	OmniPage
0.00		Resumes		Resumes
0.66		Resumes		Resumes
1.33		Resumes		WM/ I WS
1.66		Resumes		
2.00		Resumes		krauts
2.66		tttA wms		11.-4her

σ	Image	Output
0.00		Jeffery Amherst
0.66		Jeffery Amherst
1.33		Jeffery Amherst
1.66		lit wryA An memo
2.00		An wrrA Am wme
2.66		wow worm wim

σ	Binarized	OmniPage
0.00		Je eryAmherst
0.66		JefferyAmherst
1.33		JONryAmhent
1.66		GkilaryAmhent
2.00		.WlervAmMnt
2.66		Jo 116 fp Ak 11_Se-

Figure 6.12. Example recognition comparison at lower resolutions. The input images are low-pass filtered by a Gaussian of scale σ px.

Table 6.1. Comparison of word errors for the multiscale approach of our model and the binarized images input to OmniPage .

Method	Word Error
MS	31.2%
bOP	26.8%

Table 6.1 compares the word error between the systems.

6.4.4 Discussion

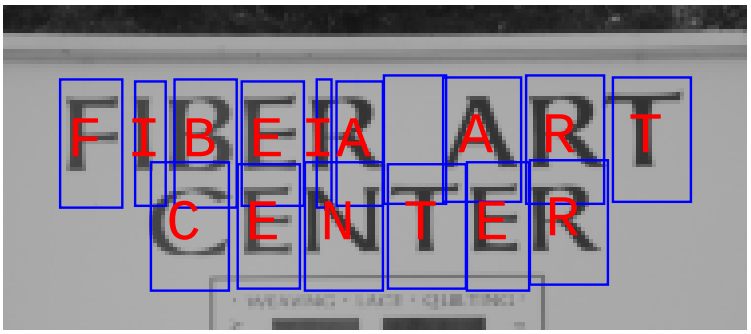
Surprisingly, the KL divergence-based beams, though adaptive and globally aware of the score distribution for a span, do not perform as well as N-best for this task. One reason for this could be that we have somewhat ham-handedly imposed a distribution on a set of optimal scores, whereas the original KL beam search used proper probability distributions from forward-backward message passing in a linear chain model. Ultimately, it seems the N-best method is best for our model and data, while the ratio threshold, which measures the difference between optimal and other candidate scores, is a good adaptive measure. We estimate that very little accuracy is lost by using the beam search approximation because the performance in Figure 6.9 levels out for parameter values with much longer processing times not shown on the graph. Thus, errors being made are likely not due to prematurely eliminated word hypotheses.

We can correctly recognize signs with a wide variety of fonts. As shown in Figure 6.11, many of these frustrate even a top commercial system. This could be due to a variety of reasons, but it seems the main one is a brittleness with respect to font diversity and noise. Despite the fact that they rhyme, the prediction of `Maus` for `HOUSE` is perplexing, as are the next three results. The unusual `A` in `LIBRARY` seems to be mistaken for an `FI` ligature. It seems the `M` in `AMHERST` is entirely too odd and gets recognized as miscellaneous punctuation. Even without the aid of a lexicon, our system recognizes this correctly. The `TAVERN` sign is interesting. Whereas the `E` character is broken up after binarization so that missing bottom the leg it does indeed look like an `F`, the subsequent two characters are ignored. While our model has the advantage of being forced to interpret the entire region, it seems odd that these characters (which are just blobs to the software), having roughly the same height and width as the others, are overlooked. We do note that the next few examples (`DOUGLASS`, `Free checking`, `MONKEY`, and `Attorney at Law`) are not recognized by our system at the original aspect ratio, but undergo the horizontal scaling optimization. However, we have no way of using a similar process for OmniPage since it does not report any sort of "confidence" that might compare to our optimal parse score.

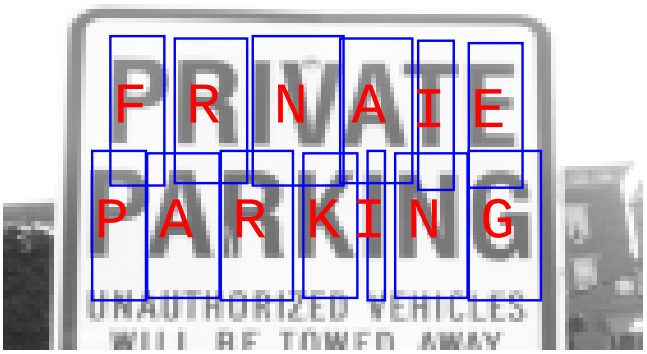
Our model also has potential for handling recognition and word segmentation at lower resolutions than it was trained upon. We do not know how OmniPage handles word segmentation, but it does fail to place a space between the two words in Figure 6.12 (bottom), even at full resolution. As the resolution of images is lowered, we imagine it will be even harder to perform a word segmentation prior to recognition with unusual kerning patterns such as these.

While the word error of the our model compared to OmniPage seems larger (Table 6.1), most of the difference is due to numbers being recognized incorrectly. Due to the nature of our corpus, numbers appear very infrequently and so there is a bias against them in the bigram model. Such errors would likely be fixed with better bigram statistics for numbers.

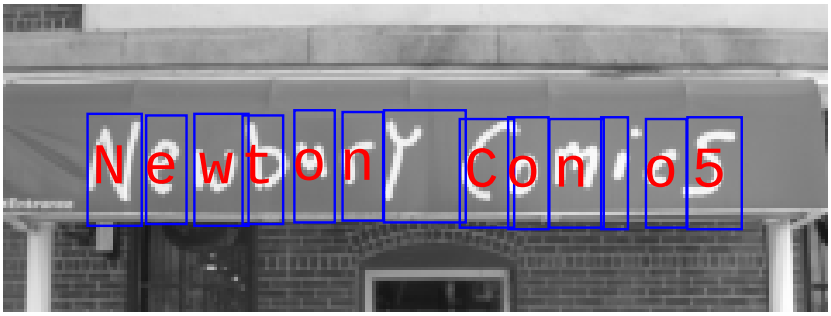
(a)



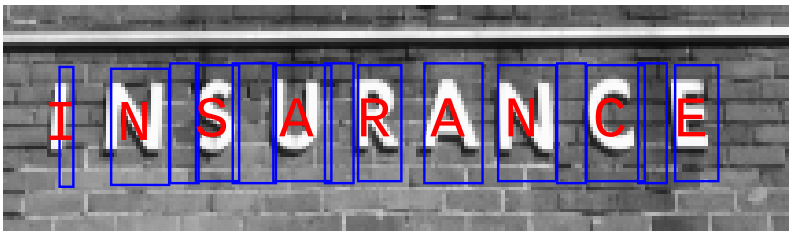
(b)



(c)



(d)



(e)

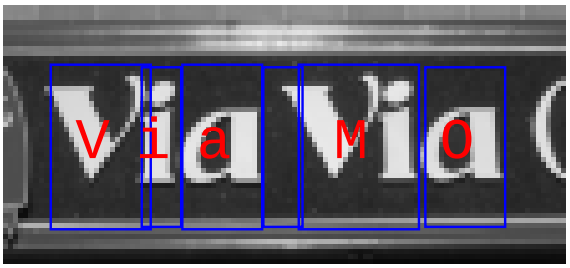


Figure 6.13. Examples of failed segmentation and/or recognition. Blue boxes indicate character bounds, and the predicted character appears centered in red. Interword spaces are marked by empty boxes.

Of course, our model has room for improvement. Figure 6.13 contains examples of some of our system failures. We have used a limited lexicon to avoid biasing recognition toward very infrequent words. As a result, the word "FIBER" is omitted from the lexicon, and gets misrecognized in Fig. 6.13(a). The system instead prefers to split the R into IA. An integrated trigram model would probably fix this problem; it is correctly recognized when the word is in the lexicon. One possibility for improvement would be to use a word-specific lexicon weight, rather than the generic θ^W, so that infrequent words might have a much lower bias but could still be included.

Another failure is that of the character appearance model. Because the discriminant (6.1) is linear in the features, certain relations cannot be captured. For instance, in Fig. 6.13(b) the leading P is recognized as an F. Presumably, the weights on the vertically oriented filters in the upper-right portion of the character bounding box for an "F" are not strong enough to prevent the confusion. Alternatively, many training instances of the F could have very close neighbors, so that verticle edges in that region are not any more inconsistent with F than they are consistent with P. More importantly, the image features are not modeled jointly. Thus, the T being predicted as I happens because the crossbar of the T in a serif font could be the same as the top of an I. Only by checking the consistency of both can this be avoided. A quadratic discriminant could solve this.

The space character model also has issues that need addressing. In Fig. 6.13(c), an actual character is covered by an inter-word space. While image features are input to this discriminant, it is not sufficient to prevent such errors. A more problematic issue with the spaces has to do with the distribution of their sizes within a query. Other than a rather generous limit on the size of a gap between characters within a word, there is nothing to prevent such gaps from being spaces, or vice-versa, save for the interpretation. Thus, it is primarily the neighboring characters (e.g., resulting bigrams or words), that determine whether such sparse image features are interpreted as gaps or spaces. In Fig. 6.13(d), several rather large gaps are confused with spaces. Although there would be a great increase in computational complexity due to a large change in the Markov properties of our model, one might imagine a Markov chain on the size of both inter-character gaps and inter-word spaces.

Finally, in order to allow a relatively straight-forward dynamic programming solution, we have ignored the font adaptivity proven useful in Chapter 4. Thus, we see in Fig. 6.13(e) that while the the first Via is recognized correctly, a seemingly identical instance is missed entirely. Finding ways to include this information, even if only in approximation, could help eliminate such errors.

We also note that this performance has been achieved with a limited amount of training data. The recognition accuracy of the model on the training data is very high. Therefore, we believe the parametric capacity of the model to recognize complex characters in many fonts is great. By employing methods that can increase the amount of training fonts and character examples, we believe that the practical, absolute performance of our model would be dramatically improved.

6.4.5 End-to-End Demonstration

In this section we describe and give a qualitative analysis of an end-to-end system for scene text detection and recognition. For detection, we use a model like the one described in the previous chapter. Since detection is a categorization-only task, we use the top 50 features selected for detection only. The template-based features are used exclusively for implementational simplicity; it is easier to calculate a subset of these features than subsets of the texture statistic features. The two class model is trained on the character and background data described in Chapter 5 (§5.4.1). For recognition, we use precisely the same model and pre-processing described in this chapter.

The text detection model is scale-specific; therefore, we run the detector on several scaled-down versions of the input image. Four scales per octave are used, and an appropriate Gaussian blur is applied before a downsampling operation with bicubic interpolation. Since the detector has no model of spatial context for continuity, we apply a moderate horizontal dilation to connect nearby text regions and pick up any missing characters on the edges. After dilation, we discard any text regions below a certain width to eliminate isolated false positives. This yields a set of candidate text regions; any detected pixel in these regions is a candidate for being centered on a character (as was shown for a single scale in Figure 6.7((d)). An example image, the multiscale detection responses and the corresponding regions are shown in Figure 6.14.

Our recognition algorithm is subsequently run on each of these candidate regions with the image scaled appropriately before recognition features are computed. If candidate regions of different scales overlap, the region with the highest per-pixel average parse score is taken as the final result. Calculating the per-pixel average is a normalization that makes scores comparable across scales. Figure 6.15 shows example recognitions and scores for the multiscale detections of Figure 6.14.

In the quantitative experiments of this chapter, we have shown that even under perfect text region detection, our recognition algorithm performs better than the commercial OmniPage software. Here, we demonstrate that good text detection is also important for producing accurate results. Figure 6.16 shows some example scene images from our data as well as the ICDAR robust reading competition [72]. In Figures 6.17 and 6.18 and Table 6.2, we qualitatively compare our end-to-end system to OmniPage, using the raw image as input to both.

6.5 Contributions

It has long been known in the handwriting recognition literature that it is necessary for character recognition to be combined with character segmentation, e.g., [64]. Furthermore, the space between words is not consistently larger than that between characters. While most approaches focus on word segmentation prior to recognition [77, 47], some have combined candidate word segmentations to find the best recognition [90]. This is analogous to continuous speech recognition, where there is very

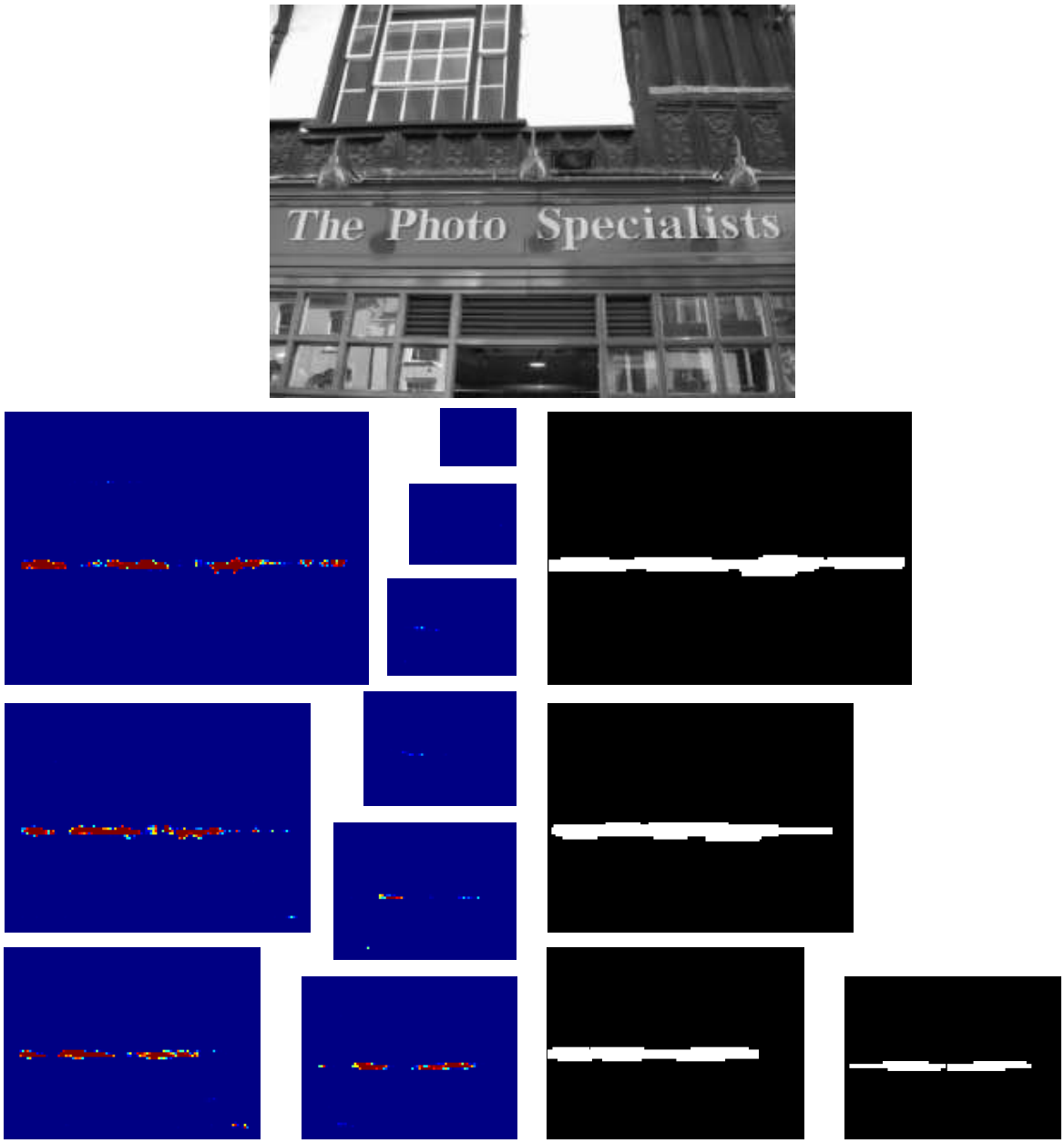
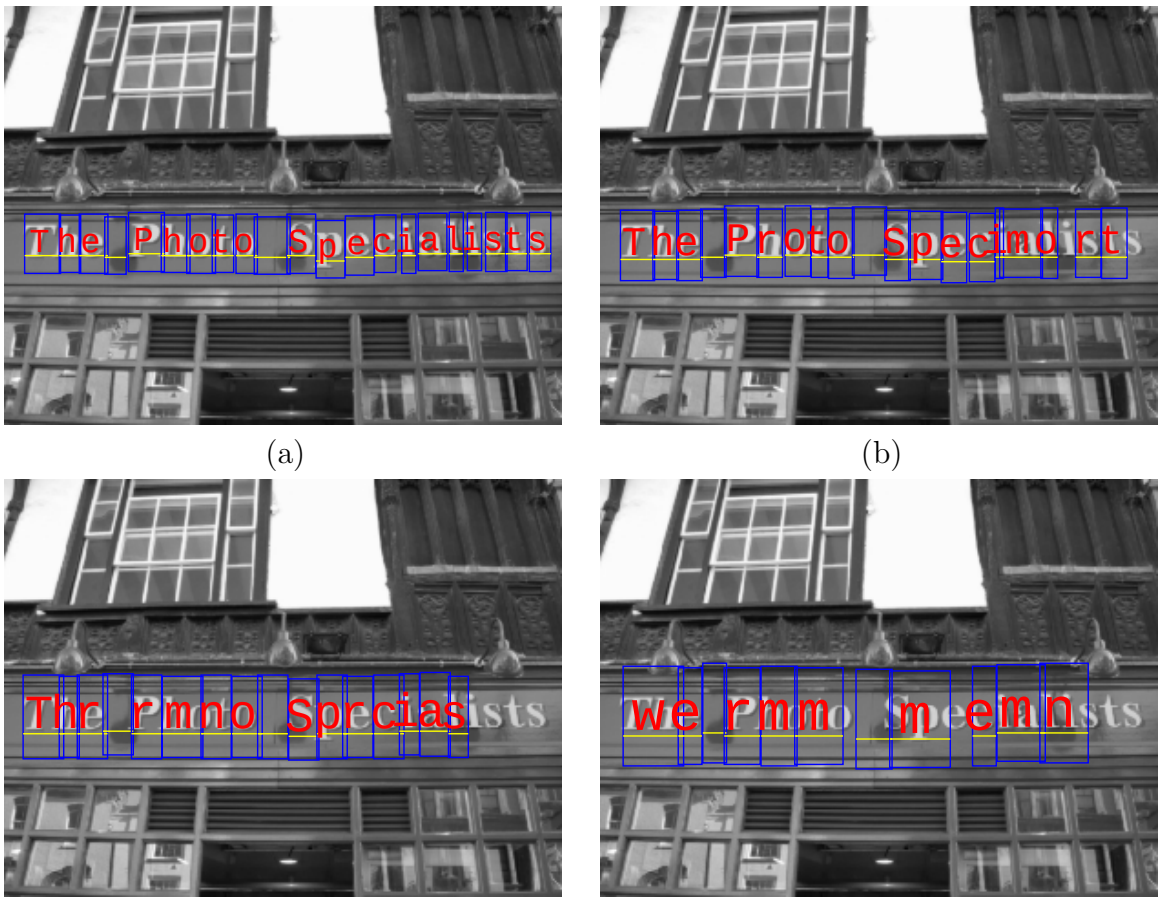


Figure 6.14. Example image (top) with multiscale text detection probabilities (left) and resulting detected regions (right).



(a) (b) (c) (d)

Panel	Scale	Output	Score
(a)	2^{0}	`The Photo Specialists`	28.97
(b)	$2^{-\frac{1}{4}}$	`The Proto Specimort`	28.29
(c)	$2^{-\frac{1}{2}}$	`Thr rmno Sprcias`	26.24
(d)	$2^{-\frac{3}{4}}$	`we rmm memn`	24.36

Figure 6.15. Recognition output on the candidate regions detected at multiple scales. The highest scoring output is the final result.

(a) (b) (c) (d) (e) (f)

Figure 6.16. Example scene images used for qualitative comparison.

(a) (b) (c) (d) (e) (f)

Figure 6.17. Example output of our end-to-end detection and recognition system.

(a) (b) (c) (d) (e) (f)

Figure 6.18. Example screen shots of detected text regions (solid yellow outlines) from OmniPage.

Table 6.2. Comparison of recognition results between our end-to-end system and OmniPage with raw image input. Panels refer to the images of Figures 6.16-6.18. The error measure is edit distance.

Panel	Correct Output	No. Chars.
(a)	The Photo Specialists	21
(b)	LLOYDS BANK	11
(c)	Fleet	5
(d)	MILL ANTIQUES	13
(e)	No footway for 450 yds	22
(f)	FREE DELIVERY	13

Panel	Our Output	No. Errors
(a)	The Photo Specialists	0
(b)	LLOYDS BANK	0
(c)	Fleet	0
(d)	MILL ANTIQUM	2
(e)	M o fo o f we y l IP m wi	18
(f)	FIREE DELIVEp	3

Panel	OmniPage Output	No. Errors
(a)	ï I taw aim 1717 ïïïï Ñal ti7Tir-TWITE.A	36
(b)	0	11
(c)	43 010 Fleet ri	10
(d)	MILL ANTIQUES J L7:"? 4,	11
(e)	ïïïïï%ï"4 "ferNirsiNee-s""esase".ïï^ noThoesnespor. a 'qpseNoï sÑÑ■ssseeseÑcee, esses sisso%ïï for 450 yds No footway) Ht. t	108
(f)	Ma OS	12

little signal to indicate word boundaries, and many possibilities must be considered [49].

Much prior work has been lexicon driven, but to robustly recognize a variety of inputs, words from outside the lexicon must be allowed. However, previous work acknowledging this has assumed both word and character segmentations, or ignored more powerful back-off models, such as character bigrams [138]. While some scene text readers use lexicons in a post-processing stage [7], integrating lexicon processing with recognition generally yields better results, as shown in Chapter 4.

The primary contribution of this chapter is in applying the techniques used for integrating character segmentation and recognition to word segmentation recognition, while retaining the ability to recognize out of lexicon words. This is especially important for scene text, which can feature many uncommon words (street names), as well as numbers (addresses) which are not in typical lexicons. While Jacobs et al. [52] can recognize low resolution words, their work is based upon the assumption of prior word segmentation. We have shown that, even in low resolution, our model can integrate these processes where prior word segmentation would likely fail.

We also believe we have contributed a technique for preventing bias in number of segments (characters) used to parse and recognize a string. This is a common problem and many normalize the total score by using the number of segments. Such a strategy prevents long strings with poor character scores from additively beating short strings with high character scores. However, we argue that it is not the right scheme. After all, we are optimizing over both segmentation and recognition. It is not how well a label explains a particular span (a given portion of the segmentation), but how well that label explains all the components of that span (the components being pixel column indices in our model). Thus, we ensure that every parse is competing on the same scale by normalizing using the "area" explained by each portion of the parse (in this case, the length of a span). In our formulation, a very large (wide) segment with a low score could have a heavier impact than in earlier methods, where it may only marginally affect an average score over a handful of segments. In short, we prefer to maximize the per-pixel normalized score, rather than the per-segment normalized score.

Though our model is moderately complex, it is rendered usable by some standard approximation methods. While others have used the KL beam for prediction and training using forward-backward message passing [89, 30], we have attempted to adapt it for the Viterbi algorithm and found our adaptation lacking for this particular environment. Perhaps in situations where N-best with a relatively small N is not appropriate (as it seems to be here), the more dynamic KL approach could give better results.

6.6 Conclusions

We have presented a model that can correctly and flexibly recognize scene text under a variety of conditions including unusual fonts, low resolution, and non-lexicon words. In achieving a greater total character accuracy, it performs better than com-

mercial document recognition software, which has been used in previous scene text readers. By integrating a lexicon to aid recognition, we can improve our performance, but we also allow non-lexicon words when warranted by the data. In addition, we need not perform prior word segmentation, because word recognition is integrated with the segmentation process, just like character segmentation and recognition. Because no binarization is required, we can recognize characters at lower resolution. Our robust probabilistic model is straightforward to train on data and requires little to no hand-tuning of parameters.

Once again, we have demonstrated a model that integrates several levels of processing which would be prone to error if done in isolation.

CHAPTER 7

SUMMARY AND CONCLUSIONS

This thesis has provided models that integrate the processes involved in reading—bottom-up detection, spatial context, segmentation, recognition, font-consistency, top-down lexicon information—and shown how more extensive communication between these processes during testing *and* training can improve results. In this, our final chapter, we briefly summarize what we have done, what it means, and to what directions it may point.

Our detection model eliminates the need for heuristic layout rules by learning the parameters for spatial context. We eliminate the "peephole" view of the sliding window approach and unify the entire process under a probabilistic model. Our first recognition framework brings together several information sources, most notably character similarity and a lexicon, into a seamlessly unified model. We have also shown how training a detector and recognizer together can improve accuracy and reduce computation time by promoting feature utility overlap and forcing interpretations earlier. Finally, our robust reader needs only rough text detection windows and completes both word and character segmentation in a recognition-driven process.

In solving the problems involved in teaching a computer to read—from detection, to segmentation and recognition—we have promoted approaches that utilize as much relevant information as possible. For detection, it was the spatial configuration of text regions. For recognition, it was character similarity, language statistics, or a lexicon. For the intersection of detection and recognition, each was allowed to influence the other in determining the features available for the discrimination task. In all of these cases, we have demonstrated that using relevant information in pipelined stages would or could not perform as well as the integrated approach.

The general contribution of the thesis is in its theme: unified information processing. Integration makes a marked difference in the accuracy of scene text reading, and some text is impossible to read without it. Our specific contributions include:

- a text detector that utilizes spatial dependencies to find text of all scales and orientations with a single classification,
- a scale-specific text detector whose features are shared with a recognizer for greater speed and accuracy, and
- two robust recognizers that enforce font consistency and use top-down lexicon knowledge to drive recognition and segmentation.

In addition to the models, we have contributed the training and inference techniques to facilitate their practical use. These technical contributions include:

- a marginal likelihood for partially labeled data,
- a simple trained function for determining character similarity,
- the use of sparse belief propagation for graphical models with cycles,
- a joint discriminative likelihood for multi-task feature selection, and
- an appropriate scheme for preventing bias in the number of segments with our recognition-driven segmentation algorithm

For decades, document recognition systems have focused on optimizing individual components of the problem, such as zoning (finding coherent text regions), word segmentation, and isolated character recognition. Indeed, when development began and much work needed to be done to make these systems practical and operable, this was a necessary and fruitful approach. However, as 99.5% recognition accuracy on a laser-printed document in the Times New Roman font becomes the norm, the document recognition community will face the challenge of journal archives with complex fonts and layouts, camera-captured document images with noise and low-resolution, and scene images with text in arbitrary colors, layouts, and world orientations. While these drastically more complex problems will undoubtedly require recognition features that are as intricate and massively parallel as biological machinery and training experience on par with that of humans, we believe that the unified approach to training and recognition will be no less necessary to achieve human-level performance. In many of these tough problems, there is just not enough discriminating local information to make bottom-up, pipelined commitments without inevitably making errors. Conversely, without consideration of the information content in lower-level processes, post-processing with top-down information is unlikely to fix major problems either. Indeed, the unified approach will become increasingly necessary to advance the state of the art and push computational systems toward human capabilities.

That two-hundred years ago, the first known character recognition system was developed to aid the visually impaired is telling. Whether one views the contributions from that patent to this thesis as a matter of teaching computers to read or as giving a limited form of sight to the blind, the progress of the last six decades has been remarkable. The advent of machine learning has had an undeniably positive impact upon reading systems. Our hope is that more extensive training and robust, integrated reasoning processes—such as those presented in this thesis—will bring great developments over the next six decades. Indeed, we believe that by the end of the *next* two-hundred years, computers should indeed be accomplished grammar-school level readers.

BIBLIOGRAPHY

[1] Agarwal, Shivani, Awan, Aatif, and Roth, Dan. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 11 (2004), 1475–1490.

[2] Baird, Henry S., and Nagy, George. Self-correcting 100-font classifier. In *Proc. of SPIE: Document Recognition* (1994), Luc M. Vincent and Theo Pavlidis, Eds., vol. 2181, pp. 106–115.

[3] Bapst, Frédéric, and Ingold, Rolf. Using typography in document image analysis. In *Electronic Publishing, Artistic Imaging, and Digital Typography* (1998), vol. 1375 of *Lecture Notes in Computer Science*, pp. 240–251.

[4] Bargeron, David, Viola, Paul, and Simard, Patrice. Boosting-based transductive learning for text detection. In *Proc. Intl. Conf. on Document Analysis and Recognition* (2005), pp. 1166–1171.

[5] Bazzi, Issam, Schwartz, Richard, and Makhoul, John. An omnifont open-vocabulary OCR system for English and Arabic. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 6 (1999), 495–504.

[6] Beal, Matthew J. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, University College London, London, 2003.

[7] Beaufort, R., and Mancas-Thillou, C. A weighted finite-state framework for correcting errors in natural scene OCR. *Proc. Intl. Conf. on Document Analysis and Recognition 2* (2007), 889–893.

[8] Belongie, S., Malik, J., and Puzicha, J. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 4 (2002), 509–522.

[9] Berger, Adam L., Della Pietra, Stephen A., and Della Pietra, Vincent J. A maximum entropy approach to natural language processing. *Computational Linguistics 22*, 1 (1996), 39–71.

[10] Bernstein, Elliot Joel, and Amit, Yali. Part-based statistical models for object classification and detection. In *Proc. Conf. on Computer Vision and Pattern Recognition* (2005), pp. 734–740.

[11] Bledsoe, W. W., and Browning, I. Pattern recognition and reading by machine. In *Proc. of Eastern Joint Computer Conf.* (1959), pp. 225–232.

[12] Blum, Avrim, and Langley, Pat. Selection of relevant features and examples in machine learning. *Artificial Intelligence 97*, 1-2 (1997), 245–271.

[13] Boyd, Stephen, and Vandenberghe, Lieven. *Convex Optimization*. Cambridge University Press, 2004.

[14] Brakensiek, Anja, Willett, Daniel, and Rigoll, Gerhard. Improved degraded document recognition with hybrid modeling techniques and character n-grams. In *Proc. Intl. Conf. on Pattern Recognition* (2000), vol. 4, pp. 438–441.

[15] Breuel, Thomas M. Classification by probabilistic clustering. In *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing* (2001), vol. 2, pp. 1333–1336.

[16] Breuel, Thomas M. Character recognition by adaptive statistical similarity. In *Proc. Intl. Conf. on Document Analysis and Recognition* (2003), vol. 1, pp. 158–162.

[17] Buntine, W., and Weigend, A. Bayesian back-propagation. *Complex Systems 5* (1991), 603–643.

[18] Carbonetto, P., de Freitas, N., and Barnard, K. A statistical model for general contextual object recognition. In *Proc. European Conf. on Computer Vision* (2004), vol. 1, pp. 350–362.

[19] Caruana, Rich. Multitask learning. *Machine Learning 28*, 1 (1997), 41–75.

[20] Chen, Datong, Odobez, Jean-Marc, and Bourlard, H. Text detection and recognition in images and video frames. *Pattern Recognition 37*, 3 (2004), 595–608.

[21] Chen, Xiangrong, and Yuille, Alan L. Detecting and reading text in natural scenes. In *Proc. Conf. on Computer Vision and Pattern Recognition* (2004), pp. 366–373.

[22] Chen, Xilin, Yang, Jie, Zhang, Jing, and Waibel, Alex. Automatic detection and recognition of signs from natural scenes. *IEEE Transactions on Image Processing 13*, 1 (2004), 87–99.

[23] Chen, Xilin, Yang, Jie, Zhang, Jung, and Waibel, Alex. Automatic detection of signs with affine transformation. In *Proceedings of the 2002 IEEE Workshop on Applications in Computer Vision (WACV2002)* (Dec. 2002), pp. 32–36.

[24] Clark, P., and Mirmehdi, M. Combining statistical measures to find image text regions. In *Proc. Intl. Conf. on Pattern Recognition* (2000), vol. 1, pp. 1450–1453.

[25] Daugman, John G. Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing 36*, 7 (1988), 1169–1179.

[26] Deng, Da, Chan, K.P., and Yu, Yinglin. Handwritten Chinese character recognition using spatial Gabor filters and self-organizing feature maps. In *Proc. Intl. Conf. on Image Processing* (1994), vol. 3, pp. 940–944.

[27] Dietterich, Thomas G., Lathrop, Richard H., and Lozano-Perez, Tomas. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence 89*, 1-2 (1997), 31–71.

[28] Edwards, Jaety, and Forsyth, David. Searching for character models. In *Advances in Neural Information Processing Systems (NIPS)* (2006), Y. Weiss, B. Schölkopf, and J. Platt, Eds., pp. 331–338.

[29] Ezaki, Nobuo, Bulacu, Marius, and Schomaker, Lambert. Text detection from natural scene images: towards a system for visually impaired persons. In *Proc. Intl. Conf. on Pattern Recognition* (2004), vol. 2, pp. 683–686.

[30] Feng, Shaolei, Manmatha, R., and McCallum, Andrew. Exploring the use of conditional random field models and HMMs for historical handwritten document recognition. In *Conf. on Document Image Analysis for Libraries* (2006), pp. 30–37.

[31] Ferreira, Silvio, Garin, Vincent, and Gosselin, Bernard. A text detection technique applied in the framework of a mobile camera-based application. In *Proc. of Workshop on Camera-based Document Analysis and Recognition* (2005).

[32] Fox, Colin, and Nicholls, Geoff. Exact MAP states and expectations from perfect sampling: Greig, Porteous and Seheult revisited. In *Proc. Twentieth International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering* (2001), pp. 252–263.

[33] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics 39* (1980), 139–202.

[34] Gao, Jiang, and Yang, Jie. An adaptive algorithm for text detection from natural scenes. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition* (December 2001), vol. 2, pp. 84–89.

[35] Gao, Jiang, Yang, Jie, Zhang, Ying, and Waibel, Alex. Text detection and translation from natural scenes. Tech. Rep. CMU-CS-01-139, Carnegie Mellon University, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 2001.

[36] Garcia, C., and Apostolidis, X. Text detection and segmentation in complex color images. In *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing* (June 2000), vol. 4, pp. 2326–2330.

[37] Geman, Stuart, and Geman, Donald. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 6 (Nov 1984), 721–741.

[38] Glantz, Herbert T. On the recognition of information with a digital computer. *Journal of the Association for Computing Machinery 4*, 2 (1957), 178–188.

[39] Grieg, D.M., Porteous, B.T., and Seheult, A.H. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society 51*, 2 (1989), 271–279.

[40] Hammersley, J.M., and Clifford, P. Markov field on finite graphs and lattices. Unpublished, 1971.

[41] He, Zuming, Zemel, Richard S., and Carreira-Perpiñán, Miguel Á. Multiscale conditional random fields for image labeling. In *Proc. Conf. on Computer Vision and Pattern Recognition* (2004), pp. 695–702.

[42] Hillel, Aharon Bar, Hertz, Tomer, and Weinshall, Daphna. Efficient learning of relational object class models. In *Proc. Intl. Conf. on Computer Vision* (2005), vol. 2, pp. 1762–1769.

[43] Hillel, Aharon Bar, and Weinshall, Daphna. Subordinate class recognition using relational object models. In *Advances in Neural Information Processing Systems (NIPS)* (2007), B. Schölkopf, J. Platt, and T. Hoffman, Eds., pp. 73–80.

[44] Ho, Tin Kam, and Nagy, George. OCR with no shape training. In *Proc. Intl. Conf. on Pattern Recognition* (2000), vol. 4, pp. 4027–4030.

[45] Hobby, John D., and Ho, Tin Kam. Enhancing degraded document images via bitmap clustering and averaging. In *Proc. Intl. Conf. on Document Analysis and Recognition* (1997), vol. 1, pp. 394–400.

[46] Hong, Tao, and Hull, Jonathan J. Improving OCR performance with word image equivalence. In *Symposium on Document Analysis and Information Retrieval* (1995), pp. 177–190.

[47] Huang, Chen, and Srihari, Sargur N. Word segmentation of off-line handwritten documents. In *Proc. Document Recognition and Retrieval* (Jan 2008).

[48] Huang, Gary, Learned-Miller, Erik, and McCallum, Andrew. Cryptogram decoding for optical character recognition. Tech. Rep. UM-CS-2006-045, University of Massachusetts-Amherst, Computer Science Research Center, University of Massachusetts, Amherst, MA 01003-4601, 2006.

[49] Huang, Xuedong, Acero, Alex, and Hon, Hsiao-Wuen. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development.* Prentice Hall PTR, 2001.

[50] Hull, Jonathan J. Incorporating language syntax in visual text recognition with a statistical model. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 12 (1996), 1251–1256.

[51] Jaakkola, T. S., and Jordan, M. I. Bayesian logistic regression: a variational approach. *Statistics and Computing 10* (2000), 25–37.

[52] Jacobs, Charles, Simard, Patrice Y., Viola, Paul, and Rinker, James. Text recognition of low-resolution document images. In *Proc. Intl. Conf. on Document Analysis and Recognition* (2005), pp. 695–699.

[53] Jain, A.K., and Bhattacharjee, S. Text segmentation using Gabor filters for automatic document processing. *Machine Vision Applications 5* (1992), 169–184.

[54] Jeon, B.W., and Landgrebe, D.A. Classification with spatio-temporal interpixel class dependency contexts. *IEEE Transactions on Geoscience and Remote Sensing 30*, 4 (July 1992), 663–672.

[55] Jones, Mark A., Story, Guy A., and Ballard, Bruce W. Integrating multiple knowledge sources in a Bayesian OCR post-processor. In *Proc. Intl. Conf. on Document Analysis and Recognition* (1991), pp. 925–933.

[56] Jurie, Frederic, and Triggs, Bill. Creating efficient codebooks for visual recognition. In *Proc. Intl. Conf. on Computer Vision* (2005), pp. 604–610.

[57] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science 220*, 4598 (1983), 671–680.

[58] Kopec, Gary E., and Lomelin, Mauricio. Document-specific character template estimation. In *SPIE Intl. Symposium on Electronic Imaging* (1996), pp. 14–26.

[59] Kornai, András. Language models: where are the bottlenecks? *AISB Quarterly 88* (1994), 36–40.

[60] Kschischang, F.R., Frey, B.J., and H.-A.Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory 47*, 2 (Feb. 2001), 498–519.

[61] Kumar, Sanjiv, and Hebert, Martial. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proc. Intl. Conf. on Computer Vision* (2003), vol. 2, pp. 1150–1157.

[62] Kusachi, Yoshinori, Suzuki, Akira, Ito, Naoki, and Arakawa, Kenichi. Kanji recognition in scene images without detection of text fields. In *Proc. Intl. Conf. on Pattern Recognition* (2004), vol. 1, pp. 457–460.

[63] Lafferty, John, McCallum, Andrew, and Pereira, Fernando. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Intl. Conf. on Machine Learning* (2001), Morgan Kaufmann, San Francisco, CA, pp. 282–289.

[64] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation 1*, 4 (1989), 541–551.

[65] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE 86*, 11 (Nov 1998), 2278–2324.

[66] Lee, Dar-Shyang. Substitution deciphering based on HMMs with applications to compressed document processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 12 (2002), 1661–1666.

[67] Li, Huiping, Doermann, David, and Kia, Omia. Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing 9*, 1 (2000), 147–156.

[68] Li, Stan Z. *Markov Random Field Modeling in Image Analysis*, second ed. Computer Science Workbench. Springer-Verlag, Tokyo, 2001.

[69] Liang, Jian, Doermann, David, and Li, Huiping. Camera-based analysis of text and documents: a survey. *International Journal on Document Analysis and Recognition 7*, 2–3 (2005), 84–104.

[70] Lienhart, Rainer. Automatic text recognition for video indexing. In *Proc. of ACM International Conf. on Multimedia* (1996), pp. 11–20.

[71] Lowe, David G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[72] Lucas, S. M., Panaretos, A., Sosa, L., Tang, A., Wong, S., and Young, R. ICDAR 2003 robust reading competitions. In *Proc. Intl. Conf. on Document Analysis and Recognition* (2003), vol. 2, pp. 682–687.

[73] Lucas, Simon M. Text locating competition results. In *Proc. Intl. Conf. on Document Analysis and Recognition* (2005), pp. 80–85.

[74] Lucas, Simon. M., Patoulas, Gregory, and Downton, Andy C. Fast lexicon-based word recognition in noisy index card images. In *Proc. Intl. Conf. on Document Analysis and Recognition* (2003), vol. 1, pp. 462–466.

[75] Manjunath, B.S., and Ma, W.Y. Texture features for browsing and retrieval of data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 9 (1996), 837–842.

[76] Manmatha, R., Han, Chengfeng, Riseman, E. M., and Croft, W. B. Indexing handwriting using word matching. In *Proc. Intl. Conf. on Digital Libraries* (New York, NY, USA, 1996), pp. 151–159.

[77] Manmatha, R., and Rothfeder, Jamie L. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 8 (2005), 1212–1225.

[78] Mathis, Charles, and Breuel, Thomas. Classification using a hierarchical Bayesian approach. In *Proc. Intl. Conf. on Pattern Recognition* (2002), vol. 4, pp. 103–106.

[79] McCallum, Andrew. Efficiently inducing features of conditional random fields. In *Proc. Conf. on Uncertainty in Artificial Intelligence* (2003), pp. 403–410.

[80] McQueen, Malcolm, and Mann, Samuel. A language model based optical character recognizer (OCR) for reading incidental text. In *Proc. of the National Advisory Committee on Computing Qualifications* (2000), pp. 207–212.

[81] Minka, Thomas P. *A Family of Algorithms for approximate Bayesian inference.* PhD thesis, MIT, Cambridge, MA, 2001.

[82] Mundy, Joseph L., and Strat, Tom, Eds. *IEEE Worskshop on Context-Based Vision* (June 1995).

[83] Mutch, Jim, and Lowe, David G. Multiclass object recognition with sparse, localized features. In *Proc. Conf. on Computer Vision and Pattern Recognition* (2006), pp. 11–18.

[84] Nagy, G., and G. Shelton, Jr. Self-corrective character recognition system. *IEEE Transactions on Information Theory 12*, 2 (April 1966), 215–222.

[85] Nagy, G., Seth, S., and Einspahr, K. Decoding substitution ciphers by means of word matching with application to OCR. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 5 (1987), 710–715.

[86] Nagy, George. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 1 (2000), 38–62.

[87] Ohya, Jun, Shio, Akio, and Akamatsu, Shigeru. Recognizing characters in scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 2 (1994), 214–220.

[88] Ozcanli, Ozge C., Tamrakar, Amir, and Kimia, Benjamin B. Augmenting shape with appearance in vehicle category recognition. In *Proc. Conf. on Computer Vision and Pattern Recognition* (2006), vol. 1, pp. 935–942.

[89] Pal, Chris, Sutton, Charles, and McCallum, Andrew. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing* (2006), vol. 5, pp. 581–584.

[90] Park, Jaehwa, Govindaraju, Venu, and Srihari, Sargur N. Efficient word segmentation driven by unconstrained handwritten phrase recognition. In *Proc. Intl. Conf. on Document Analysis and Recognition* (1999), pp. 605–608.

[91] Perkins, Simon, Lacker, Kevin, and Theiler, James. Grafting: fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research 3* (2003), 1333–1356.

[92] Petkov, N., and Kruizinga, P. Computational model of visual neurons specialised in the detection of period and aperiodic oriented visual stimuli: bar and grating cells. *Biological Cybernetics 76* (1997), 83–96.

[93] Portilla, Javier, and Simoncelli, Eero P. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision 40*, 1 (2000), 49–71.

[94] Qi, Yuan, Szummer, Martin, and Minka, Thomas P. Bayesian conditional random fields. In *Proc. Intl. Workshop on Artificial Intelligence and Statistics* (2005).

[95] Quattoni, Ariadna, Collins, Michael, and Darrel, Trevor. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems (NIPS)* (2005), vol. 17.

[96] Rath, T.M., and Manmatha, R. Word spotting for historical documents. *International Journal on Document Analysis and Recognition 9*, 2-4 (2007), 139–152.

[97] Rayner, Keith, and Pollatsek, Alexander. *The Psychology of Reading*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[98] Rehling, John A. *Letter Spirit (Part Two): Modeling Creativity in a Visual Domain*. PhD thesis, Indiana University, July 2001.

[99] Rice, S., Nagy, G., and Nartker, T. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, 1999.

[100] Riseman, E. M., and Hanson, A.R. A contextual postprocessing system for error correction using binary n-grams. *IEEE Transactions on Computers C-23* (May 1974), 480–493.

[101] Sarawagi, Sunita, and Cohen, William W. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems (NIPS)* (Cambridge, MA, 2005), Lawrence K. Saul, Yair Weiss, and Léon Bottou, Eds., MIT Press, pp. 1185–1192.

[102] Schantz, Herbert F. *The History of OCR.* Recognition Technologies Users Association, 1982.

[103] Serre, Thomas, Wolf, Lior, and Poggio, Tomaso. Object recognition with features inspired by visual cortex. In *Proc. Conf. on Computer Vision and Pattern Recognition* (2005), vol. 2, pp. 994–1000.

[104] Shen, Huiying, and Coughlan, James. Finding text in natural scenes by figure-ground segmentation. In *Proc. Intl. Conf. on Pattern Recognition* (2006), vol. 4, pp. 113–118.

[105] Shepard, David H. United States patent #2,663,758: Apparatus for reading, 1953.

[106] Shi, Hongwei, and Pavlidis, Theo. Font recognition and contextual processing for more accurate text recognition. In *Proc. Intl. Conf. on Document Analysis and Recognition* (1997), pp. 39–44.

[107] Silapachote, Piyanuch, Weinman, Jerod, Hanson, Allen, Weiss, Richard, and Mattar, Marwan A. Automatic sign detection and recognition in natural scenes. In *IEEE Workshop on Computer Vision Applications for the Visually Impaired* (June 2005).

[108] Simard, Patrice Y., Steinkraus, Dave, and Platt, John C. Best practices for convolutional neural networks applied to visual document analysis. In *Proc. Intl. Conf. on Document Analysis and Recognition* (2003), vol. 2, pp. 958–963.

[109] Simoncelli, Eero P., and Freeman, William T. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Proc. Intl. Conf. on Image Processing* (1995), vol. 3, pp. 444–447.

[110] Strat, T.M., and Fischler, M.A. Context-based vision: Recognizing objects using information from both 2-D and 3-D imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 10 (Oct. 1991), 1050–1065.

[111] Sutton, Charles, and McCallum, Andrew. Piecewise training of undirected models. In *Proc. Conf. on Uncertainty in Artificial Intelligence* (2005), pp. 568–575.

[112] Sutton, Charles, McCallum, Andrew, and Rohanimanesh, Khashayar. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research 8* (Oct. 2007), 693–723.

[113] Thillou, Céline, Ferreira, Silvio, and Gosselin, Bernard. An embedded application for degraded text recognition. *Eurasip Journal on Applied Signal Processing 13* (2005), 2127–2135.

[114] Torralba, Antonio, Murphy, Kevin P., and Freeman, William T. Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. Conf. on Computer Vision and Pattern Recognition* (2004), vol. 2, pp. 762–769.

[115] Torralba, Antonio, Murphy, Kevin P., and Freeman, William T. Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems (NIPS)* (2005), pp. 1401–1408.

[116] Torralba, Antonio, Murphy, Kevin P., and Freeman, William T. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 5 (2007), 854–869.

[117] Torralba, Antonio, Murphy, Kevin P., Freeman, William T., and Rubin, Mark A. Context-based vision system for place and object recognition. In *Proc. Intl. Conf. on Computer Vision* (2003), IEEE Computer Society, pp. 273–280.

[118] Tu, Zhuowen, Chen, Xiangrong, Yuille, Alan L., and Zhu, Song-Chun. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision 63*, 2 (2005), 113–140.

[119] Ullman, Shimon, Sali, Erez, and Vidal-Naquet, Michel. A fragment-based approach to object representation and classification. In *International Workshop on Visual Form* (2001), no. 2059 in LNCS, pp. 85–102.

[120] Veeramachaneni, Sriharsha, and Nagy, George. Adaptive classifiers for multisource OCR. *International Journal on Document Analysis and Recognition 6*, 3 (2003), 154–166.

[121] Viola, P., and Jones, M.J. Robust real-time face detection. *International Journal of Computer Vision 57*, 2 (May 2004), 137–154.

[122] Wang, L., and Pavlidis, T. Direct gray-scale extraction of features for character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 10 (1993), 1053–1067.

[123] Weinman, Jerod. Data-dependent spatial context for computer vision with conditional Markov fields. Tech. Rep. UM-CS-2005-052, University of Massachusetts-Amherst, Computer Science Research Center, University of Massachusetts, Amherst, MA 01003-4601, Sept. 2005.

[124] Weinman, Jerod, Hanson, Allen, and McCallum, Andrew. Sign detection in natural images with conditional random fields. In *IEEE Intl. Workshop on Machine Learning for Signal Processing* (September 2004), pp. 549–558.

[125] Weinman, Jerod J., and Learned-Miller, Erik. Improving recognition of novel input with similarity. In *Proc. Conf. on Computer Vision and Pattern Recognition* (June 2006), pp. 308–315.

[126] Weinman, Jerod J., Learned-Miller, Erik, and Hanson, Allen. Fast lexicon-based scene text recognition with sparse belief propagation. In *Proc. Intl. Conf. on Document Analysis and Recognition* (Sept 2007), pp. 979–983.

[127] Welch, J.R., and Salter, K.G. A context algorithm for pattern recognition and image interpretation. *IEEE Transactions on Systems, Man, and Cyberneics 1*, 1 (January 1971), 24–30.

[128] Williams, Peter M. Bayesian regularization and pruning using a Laplace prior. *Neural Computation 7* (1995), 117–143.

[129] Winkler, Gerhard. *Image Analysis, Random Fields, and Markov Chain Monte Carlo Methods*, second ed. Springer-Verlag, Berlin, 2003.

[130] Winn, J., Criminisi, A., and Minka, T. Object categorization by learned universal visual dictionary. In *Proc. Intl. Conf. on Computer Vision* (2005), pp. 1800–1807.

[131] Wolf, Christian, Jolion, Jean-Michel, and Chassaing, Francoise. Text localization, enhancement and binarization in multimedia documents. In *Proc. Intl. Conf. on Pattern Recognition* (2002), vol. 4, pp. 1037–1040.

[132] Wu, Victor, Manmatha, R., and Riseman, Edward M. Finding text in images. In *Proc. Intl. Conf. on Digital Libraries* (1997), pp. 3–12.

[133] Wu, Victor, Manmatha, Raghavan, and Riseman, Edward M. Textfinder: An automatic system to detect and recognize text in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 11 (1999), 1224–1229.

[134] Xi, Jie, Hua, Xian-Sheng, Chen, Xiang-Rong, Wenyin, Liu, and Zhang, Hong-Jiang. A video text detection and recognition system. In *Proc. of IEEE International Conf. on Multimedia and Expo* (2001), pp. 222–225.

[135] Yamaguchi, Takuma, and Maruyama, Minoru. Character extraction from natural scene images by hierarchical classifiers. In *Proc. Intl. Conf. on Pattern Recognition* (2004), vol. 2, pp. 687–690.

[136] Yedidia, Jonathan S., Freeman, William T., and Weiss, Yair. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory 51*, 7 (2005), 2282–2312.

[137] Zhang, Dong-Qing, and Chang, Shih-Fu. Learning to detect scene text using a higher-order MRF with belief propagation. In *IEEE Workshop on Learning in Computer Vision and Pattern Recognition* (2004), vol. 06, pp. 101–108.

[138] Zhang, DongQing, and Chang, Shih-Fu. A Bayesian framework for fusing multiple word knowledge models in videotext recognition. In *Proc. Conf. on Computer Vision and Pattern Recognition* (2003), vol. 2, pp. 528–533.

[139] Zhou, Yaqian, Wenb, Fuliang, Wu, Lide, and Schmidt, Hauke. A fast algorithm for feature selection in conditional maximum entropy modeling. In *Proc. Conf. on Empirical Methods in Natural Language Processing* (2003), pp. 153–159.