Learning shape priors with neural networks

by

Simon Safar

B.S. (Budapest University of Technology and Economics) 2010
M.S. (Budapest University of Technology and Economics) 2010

A thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science
in

Electrical Engineering and Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, MERCED

Committee in charge:

Professor Ming-Hsuan Yang, Chair
Professor Sungjin Im
Professor Shawn Newsam

Fall 2014

Learning shape priors with neural networks

The thesis of Simon Safar is approved, and it is acceptable
in quality and form for publication on microfilm and electronically:

Prof. Ming-Hsuan Yang, Chair	Date
Prof. Sungjin Im	Date
Prof. Shawn Newsam	Date

University of California, Merced

Fall 2014

Abstract

Learning shape priors with neural networks

by

Simon Safar

Master of Science in Electrical Engineering and Computer Science

University of California, Merced

Professor Ming-Hsuan Yang, Chair

We propose two methods for object segmentation by combining learned shape priors with local features. The first, Max-Margin Boltzmann Machines, learns shapes in an unsupervised way, followed by a joint refinement using features extracted from the image, using max-margin methods. Second, we investigate the feasibility of another approach, based on deep learning and patchwise output mask refinement. As a way to further improve results, we also present an application of structured learning to learn Graph Cut based segmentation mask smoothing.

We conduct experiments on datasets containing diverse images of three classes of objects, showing promising results. We also discuss both qualitative and quantitative results extensively and point out both the strengths and shortcomings of the above approaches.

Professor Ming-Hsuan Yang
Thesis Committee Chair

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I'm superbly grateful to my advisor, Prof. Ming-Hsuan Yang, for the advice, help and encouragement throughout my studies at UC Merced. Ming-Hsuan always had time for all of us, no matter the circumstances, and I have learned a great lot from him not only on the technical details but also on the way of thinking every good researcher should have.

I would like to thank members of our lab, especially Jimei Yang, for the many fruitful discussions and collaboration, without which most of this thesis wouldn't exist. I also highly appreciate the time and work that members of my committee, Prof. Sungjin Im and Prof. Shawn Newsam, put into reviewing this text.

Further thanks is due to Yujia Li for providing code for their paper to run comparative experiments, to the CVPR reviewers for their feedback on our work on Max-Margin Boltzmann Machines, and to NSF, for supporting part of this work through by CAREER Grant #1149783 and IIS Grant #1152576.

Simon Sáfár

Contact Information

Address: Room 311, S&E 2, UC Merced, CA 95343
Phone: (209) 417-5822
E-mail: ssafar@ucmerced.edu
Webpage: https://eng.ucmerced.edu/people/ssafar

Education

University of California, Merced, CA, USA
Ph.D. student, EECS — August 2012 - present
- Advisor: Ming-Hsuan Yang

Budapest University of Technology and Economics, Budapest, Hungary
B.S. and M.S., Electrical Engineering — September 2004 - June 2010
- Advisor: dr. Bela Pataki
- Final grade: 4.66 out of 5

Publications

Jimei Yang, Simon Safar, and Ming-Hsuan Yang, *Max-Margin Boltzmann Machines for Object Segmentation*, accepted by the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

Research and Development Experience

University of California, Merced
Merced, CA, USA — August 2012 - Present
- **Max-Margin Boltzmann Machines for Object Segmentation**
  We used structured output prediction combined with Restricted Boltzmann Machines as a shape prior to segment specific object classes from images.
- **Object Tracking by Detection of Multiple Parts**
  In this project, we experimented with combining detections from multiple object parts to obtain robustness against occlusions, predicting position and scale changes in a Hough voting framework.
- **Movement-based Saliency in Videos using Temporal Superpixels**
  To segment (and co-segment) videos, this project aimed for localizing interesting regions based on their movement relative to other parts of the frame. We used the Temporal Superpixel framework to obtain trajectories to analyze.

Google Inc.
Mountain View, CA, USA — Jun 2014 - September 2014
Software Engineering Intern, Multimedia Content Analysis team

Google Inc.
Mountain View, CA, USA — May 2013 - August 2013
Software Engineering Intern, Shopping

Ericsson Hungary Ltd.
Budapest, Hungary — May 2011 - July 2012
Software developer, Mobile Media Gateway (mostly C++ and Java).

Dolphio Consulting Ltd.
Budapest, Hungary — January 2009 - May 2011
- **Various implementations using CUDA**
  Designed and wrote code for Singular Value Decomposition, Linear Regression (including QR decomposition) and a BP Neural Network framework. Also accelerated an existing Linear Programming solver with a specialized sparse Cholesy decomposition algorithm running on the GPU.

- **Stereo camera system for passenger counting**
  Optimized code of a stereo camera based passenger counter system, ported components to a DSP, cleaned up the theory behind camera calibration and improved the GTK+-based camera calibration tool.
- **Further development tasks**
  Includes a partial rewrite of a dual-tree complex wavelet transform algorithm and writing a framework for a company logo detection system using SIFT and SURF descriptors.

Budapest University of Technology and Economics
Budapest, Hungary — September 2004 - June 2010

- **Implementation of RBMs**
  As my thesis work, I implemented Restricted Boltzmann Machines based on the work on G. Hinton and his colleagues, aiming for the reproduction of their results on handwritten digit and texture recognition.
- **Texture Recognition in X-Ray images**
  Used various descriptors to classify areas of mammography images and try to locate suspicious areas. Exact borders of possible lesions were located using a descriptor based on the Radial Gradient Index.

Teaching Experience

University of California, Merced, CA, USA
Teaching Assistant — August 2012 - Present

- **CSE031 - Introduction to Computer Science and Engineering II.**
- **CSE021 - Introduction to Computing II.**
- **CSE021 - Introduction to Computing I.**

Dolphio Consulting Ltd., Budapest, Hungary
Mentor, Internship Program — January 2009 - June 2010

- **Sign Language Recognition**, for a group of 3 interns
- **Sorting Algorithms on GPUs**, for a group of 2 interns
- **Jersey Number Recognition in Sports Videos**
- **Dimensionality Reduction on GPUs**

Relevant coursework and test results

Courses

- Advanced Topics in Computer Vision
- Optimization
- Knowledge-Based Architectures (Machine Learning)
- Three-Dimensional Vision Systems
- Data Mining Algorithms
- Probability Theory
- Artificial Intelligence

GRE results

- Math: 170/170 (99th percentile)
- CS Subject Test: 870/900 (98th percentile)

Awards

- Best Improvement Award, Media Gateway Development department, Ericsson Hungary, 2012 (in a group of 3, for a testing tool).
- Chancellor's Graduate Fellowship, UC Merced, 2012.
- Honorary prize, Scientific Conference for Undergraduate Students (with a work "Detecting and Evaluating Lesions on Mammography Images"), 2008.

- 2nd place, National Scientific Conference for Undergraduate Students, co-author (with a work “Test of an Optimal Electoral System”), 2008.
- 2nd place, Instrumentation Technology Competition, Faculty of Electrical Engineering and Computer Science, BUTE, 2006.
- 1st place, Eötvös Loránd National Physics Contest, 2005.
- 11th place, National High School Mathematics Contest (OKTV), 2004.

Programming Skills

Proficiency with Python, Matlab, C/C++, Java.
Has experience with Javascript, PHP, C#, OpenCV, CUDA and Common Lisp.

Chapter 1

Introduction

An important area of research in computer vision is coming up with methods to understand what can be seen in an image. For example, is it a picture of a bird? [3] However, in addition to identifying image contents, we are also often interested in their location within it.

To pinpoint locations, we might use relatively more coarse outputs, such as drawing a bounding box around the objects we recognized. This task is known as "object detection" in vision. For many applications, such a location is sufficiently accurate: among other examples, it is used by driver assistance systems to warn drivers of pedestrians; also, to analyze large amounts of security camera footage in terms of movement of people.

However, in other cases, we might need to know of every pixel whether it belongs to the foreground or the background. Image editing is such an example: to automatically remove occluding people from an otherwise perfectly captured scene, we need to know exactly which pixels belong to the tourists who stepped into the picture in the wrong moment. Or we might take robotics: to grasp something using our manipulators, we need to know its exact shape. In other words, we need to segment the image into a foreground and a background region.

To perform segmentation, we might use several visual properties of image areas to tell apart foreground from background. For example, color and texture can be highly distinctive in many cases; grass is often green and sky regions are often blue. Edge information is also useful: large, uniform areas usually belong to the same category, with strong, distinctive edges separating different ones.

Of course, none of these cues is perfectly reliable. Nothing prevents clothing items or some birds from being the same shade of blue as the sky, which also often changes its color. Edges might be confusing in cluttered scenes, and also, other image features that we can extract also have their limitations. All this is often further complicated by our target being occluded by other entities. As a way to ameliorate this problem, probabilistic methods are commonly used, fusing these pieces of information to come up with a reasonable guess for the truth. The more cues we can find, the better guesses we will get.

In this thesis, we are concerned with a smaller subset of what we could be looking for in an image: objects. Unlike generic types of material, like water or road surfaces often found in scene parsing tasks, objects have a specific location and well-defined boundaries. The points described above still stand though: birds, for example, might have a great variety of colors and texture, and might be hiding in other complicated structures, like trees.

However, fortunately, objects do have an extra property that can be exploited for segmentation purposes: shape. Even if the exact boundaries are unclear even for humans when looking at the image from up close, we can still draw the boundary accurately knowing that the object in question should be generally bird-shaped. In fact, we can even recognize objects based on nothing else but their outline. By building a system that learns and uses shapes, we can overcome the difficulties presented by the ambiguity of local-only features.

The approaches in this thesis all use the starting point of having a more-or-less accurate bounding box available, and are aiming for an accurate foreground-background segmentation. Such a bounding box can be obtained from using using class-specific object detectors (e.g. R-CNN [4]).

In this work, we study several key questions on exploiting shape information:

- What shape representation should we use?
- How is shape information to be learned from training data?
- How can we use this model in the segmentation process itself?

Of course, one possible answer to all these questions is not to represent shape at all, or only do so in a significantly simplified manner. For example, approaches based on Conditional Random Fields with pairwise terms make the simple assumption that physical objects tend to be laid out continuously in space, making any two mask pixels likely to have the same label (both foreground or both background). Despite their simplicity, these methods perform reasonably well in practice. We give an overview of these along with more advanced approaches along with our own methods in the following chapters; they will serve as a baseline for performance comparisons.

We can also use unsupervised learning to learn the possible global shapes of an object class' members, along with that of their main parts, an approach taken by Li *et al.* in [1]. Nevertheless, if we also use signals from image features to guide all components of such a model, we get more accurate representations. Chapter 2 presents such an approach: after unsupervised pre-training using Boltzmann machines, we learn the relation of image features to possible shapes in a max-margin framework.

Another possibility is to substitute the sophisticated unsupervised machinery with the power of deep learning. Convolutional networks can also provide a way to learn not only a global representation of shape but also extend it to local patterns, a capability that helps us deal with objects undergoing significant shape changes (such as legs of horses). In Chapter 3, we describe such an architecture, showing how global shape information can be integrated into the segmentation process, and compare the performance of this method to that of global-only models.

Finally, we return to our starting point, Conditional Random Fields, with a different approach. Instead of only applying them as a refinement step, like in Chapter 2, we learn the CRF parameters as an integral part of a deep network. Chapter 4 explains how this can be done using structured learning.

Our experiments show that we can obtain significant improvements compared to state-of-the-art methods. We conclude the thesis with Chapter 5, interpreting the results, drawing conclusions and mapping out possible future research directions.

Chapter 2

Max-Margin Boltzmann Machines

2.1 Introduction

Object segmentation[1] can be formulated as a structured output problem that involves making predictions collectively over correlated output labels $\mathbf{y} \in \mathcal{Y}$ from input observations $\mathbf{x} \in \mathcal{X}$. One of the core issues in structured output prediction problems is how to represent complex output variable interrelations effectively while carrying out inference and learning efficiently.

In Markov Random Fields (MRFs), output structures are represented by pairwise and high-order potential functions $p(\mathbf{y}) = \prod_{\mathbf{y}_i \subset \mathbf{y}} \phi(\mathbf{y}_i)/\mathbf{Z}$ where $\mathbf{Z}$ is the partition function. The prediction from the observations $\mathbf{x}$ to the labels $\mathbf{y}$ is usually realized in the conditional models $p(\mathbf{y}|\mathbf{x})$, i.e., Conditional Random Fields (CRFs) [6], which allow flexible use of various long-range features from observations $\mathbf{x}$. Pairwise potentials [7], although admitting efficient inference, can only capture limited local structure, such as smoothness and edges. High-order potentials are able to capture long-range interactions between pixel labels through bottom-up segmentation [8], pattern-based priors [9], [10]. Beyond the generic high-order priors, the ObjCut algorithm [11] introduces category-specific object models into MRFs and has shown good segmentation performance on articulated objects. In ObjCut, the hidden variables of pictorial structures encode the positions of object parts, but their interactions with pixel labels are manually designed.

[1]This chapter was previously published as [5] and is reproduced with permission from the authors.

Alternatively, Restricted Boltzmann Machines (RBMs) render more flexible models for structured output representation that learn high-order interrelations through a joint distribution of labels and a set of hidden (latent) variables $\mathbf{h}$ in $p(\mathbf{y}, \mathbf{h})$. By omitting lateral connections in a single layer, RBMs admit efficient inference and sampling from conditional probabilities. When operating with a small number of training samples, layered architectures [12] have been shown more effective in terms of model expressiveness and learning efficiency. Eslami et al. [13] propose a two-layer Boltzmann Machine $p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2)$ (where $\mathbf{h}^1$ and $\mathbf{h}^2$ denote hidden variables in two layers) for modeling object shapes (ShapeBMs), and apply it onto object segmentation in a generative model $p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{x})$ [14].

In this chapter, we present a general class of Conditional Boltzmann Machines (CBMs) for object segmentation in the form of $p(\mathbf{y}, \mathbf{h}|\mathbf{x})$ and $p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2|\mathbf{x})$. In addition to the connections from image to labels, our models also include the connections from the image to hidden variables, which allows direct shape inference from image features. Based on layer-wise conditional independence of BMs, we derive a simple but efficient Iterated Conditional Modes [15] algorithm for maximum a posteriori (MAP) inference.

Learning with CRFs and CBMs is challenging as it requires handling exponentially large numbers of output combinations in data-dependent partition functions. Approximate learning algorithms are easily trapped in local optima, thereby limiting their generalization performance. Another line of research for structured output prediction is developed on max-margin formulations [16]–[18], that facilitates model generalizability to unseen test data. This technique has been applied to CRFs for object segmentation [7], [19]. In a similar spirit, we propose a max-margin formulation of CBMs, referred as MMBMs, and develop an online Concave-Convex Procedure (CCCP) [20] algorithm for learning efficiently with hidden variables. Note that large margin BMs have been proposed in [21] with a focus on theoretical analysis while our max-margin method is proposed for training a particular class of CBMs with applications to object segmentation. We investigate the effects of four kinds of margin functions on discriminative training, and demonstrate the importance of combined hidden and visible margin functions. We study two variants of MMBMs with a single hidden layer $p(\mathbf{y}, \mathbf{h}|\mathbf{x})$ as well as two hidden layers $p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2|\mathbf{x})$, and compare them with two state-of-the-art models: superpixel based CRFs [22] and Compositional High Order Pattern Potentials [1]. We carry out experiments on the Weizmann horse [23], Penn-Fudan pedestrian [24] and Caltech-UCSD birds 200 [25] datasets. Experimental results show that the proposed MMBMs perform better than existing methods both quantitatively and qualitatively.

2.2 Related work

Recent work [1], [26] on object segmentation realizes the power of Boltzmann Machines to represent high-order interactions in combining RBMs with CRFs. Li et al. [1] combine pairwise, data-dependent potentials with a one-layer RBM prior in CRFs (referred as Compositional High Order Pattern Potentials (CHOPPs) in Figure 2.1(b)), and show the relationship between the marginalized RBM free energy and high-order potentials [9]. Kae et al. [26] augment CRFs with an RBM shape prior in a two-layer model for image labeling. Their lower layer has nodes for every superpixel of the image, with pairwise weights connecting them. The labels for this layer are then pooled into a raster structure, enabling them to use a RBM to provide shape priors. Another attempt is to combine Deep Boltzmann Machines shape prior with a variational segmentation model [27], showing the effectiveness of strong shape priors for simple object segmentation. In all of the above approaches, the only inference pathway between the image features $\mathbf{x}$ and the hidden variables $\mathbf{h}$ representing shapes leads through the labels assigned to image pixels $\mathbf{y}$ while the shape only works as a prior. To perform inference and learning, the hidden variables are usually marginalized through an EM-like procedure. The shape information is thus not fully explored. In contrast, our MMBM models introduce connections between hidden variables $\mathbf{h}$ and image features $\mathbf{x}$, which enables a more efficient MAP inference procedure and thus max-margin learning.

2.3 Models

In this section, we first introduce two variants of Boltzmann Machines, RBMs and ShapeBMs, for modeling object shapes, and then describe the proposed conditional models and the maximum a posteriori inference algorithm.

2.3.1 Boltzmann Machines

Given a labeled image of an object, we represent the mask as a set of visible variables $\mathbf{y} \in \{0,1\}^n$. RBMs use one layer of hidden variables $\mathbf{h} \in \{0,1\}^m$ to capture global dependencies between visible variables (See Figure 2.1(a))

$$p(\mathbf{y}, \mathbf{h}) = \exp(-E(\mathbf{y}, \mathbf{h}))/\mathbf{Z}, \tag{2.1}$$

where $\mathbf{Z}$ is the partition function. RBMs do not have lateral connections within visible and hidden layers so that the energy function takes the form,

$$E(\mathbf{y}, \mathbf{h}) = -\mathbf{y}^\top \mathbf{W}\mathbf{h} - \mathbf{b}^\top \mathbf{y} - \mathbf{c}^\top \mathbf{h}, \tag{2.2}$$

parametrized by $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{W}$. One attractive property of RBMs is that visible variables are conditionally independent given hidden variables and vice versa. The conditional probability of each variable is essentially the sigmoid function $\sigma(y) = 1/(1+\exp(-y))$,

$$p(y_i = 1|\mathbf{h}) = \sigma(\sum_j w_{ij} h_j + b_i), \tag{2.3}$$

$$p(h_j = 1|\mathbf{y}) = \sigma(\sum_i w_{ij} y_i + c_j), \tag{2.4}$$

which facilitates efficient inference.

Although RBMs have the capacity of modeling complex distributions, they require a large set of hidden variables and numerous training examples. For object segmentation, it is labor intensive to collect a large number of training examples with ground truth masks, and challenging to train RBMs with a large set of variables. It is, however, possible to ameliorate this problem by considering the spatial structure of images. Eslami et al. [13] propose a particular form of Boltzmann Machine with two hidden layers $p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2)$ (referred as ShapeBM) for object shape modeling. The first layer of hidden variables $\mathbf{h}^1$ is partitioned into several disjoint subsets $\{\mathbf{h}^1_k = \mathbf{h}^1(\mathbf{J}_k)\}_{k\in\mathbf{G}}$ of same size m^1_k, where $\mathbf{J}_k \in \{0,1\}^m$ denotes the subset indexing. Each of them has a restricted receptive field and only connects to a local patch of the object mask. The local patches $\{\mathbf{y}_k = \mathbf{y}(\mathbf{I}_k)\}_{k\in\mathbf{G}}$ have the same size n_k and they overlap each other along the boundaries, where $\mathbf{I}_k \in \{0,1\}^n$ denotes the patch index. Therefore, the pairwise potentials between visible variables $\mathbf{y}$ and the first layer hidden variables $\mathbf{h}^1$ can be represented by $\sum_{k\in\mathbf{G}} \mathbf{y}_k^\top \mathbf{W}^1_k \mathbf{h}^1_k$. Furthermore, different patches can share the same weights $\mathbf{W}^1 = \mathbf{W}^1_k, k \in \mathbf{G}$. The second layer of hidden variables $\mathbf{h}^2$ connects to all the variables $\mathbf{h}^1$ of the first layer. Similar to RBMs, there are no lateral connections between variables within any single layer. The energy function can be thus described by

$$\begin{aligned}\mathbf{E}(\mathbf{y},\mathbf{h}^1,\mathbf{h}^2) = & -\sum_{k\in\mathbf{G}} \mathbf{y}_k^\top \mathbf{W}^1 \mathbf{h}_k^1 - \mathbf{b}^\top \mathbf{y} - \\ & \mathbf{c}^{1\top}\mathbf{h}^1 - \mathbf{h}^{1\top}\mathbf{W}^2\mathbf{h}^2 - \mathbf{c}^{2\top}\mathbf{h}^2.\end{aligned} \tag{2.5}$$

The pairwise term of the first layer can be rewritten in the same form as RBMs by some matrix manipulation:

$$\begin{aligned}&\sum_{k\in\mathbf{G}} \mathbf{y}_k^\top \mathbf{W}^1 \mathbf{h}_k^1 = \mathbf{y}^\top \tilde{\mathbf{W}}^1 \mathbf{h}^1, \\ &\text{where } \tilde{\mathbf{W}}^1(\mathbf{I}_k,\mathbf{J}_k) = \mathbf{W}^1.\end{aligned} \tag{2.6}$$

Due to its structure, ShapeBM uses much fewer parameters than conventional two-layer RBMs [12], thereby facilitating efficient learning for smaller datasets. The pairwise term $\mathbf{y}^\top\tilde{\mathbf{W}}^1\mathbf{h}^1$ models the compatibility between pixels and parts while the term $\mathbf{h}^{1\top}\mathbf{W}^2\mathbf{h}^2$ defines the possible configuration of parts. Thus, when an unit of $\mathbf{h}^1$ is activated, a template stored in $\mathbf{W}^1$ is selected to enforce the group of pixels to obey a binary image pattern. Also, when an unit of $\mathbf{h}^2$ is activated, it triggers a particular configuration of parts (due to varying pose or viewpoint). The ShapeBM architecture also enjoys the property of conditional independence $p(\mathbf{y}|\mathbf{h}^1), p(\mathbf{h}^1|\mathbf{y},\mathbf{h}^2)$ and $p(\mathbf{h}^2|\mathbf{h}^1)$, although exact inference is not tractable for this model.

2.3.2 Conditional Boltzmann Machines

While generative RBMs and ShapeBMs are capable of modeling object shape priors, it is still challenging to efficiently infer a binary object mask $\mathbf{y}$ from an image $\mathbf{x}$. Intuitively, we can construct a fully generative model for object images and their binary masks $p(\mathbf{y},\mathbf{x})$ such that object shape can be inferred from an image by the conditional distribution $p(\mathbf{y}|\mathbf{x})$, and an image generated from a shape mask by $p(\mathbf{x}|\mathbf{y})$. As an example, Eslami et al. [14] present a generative multinomial joint model of appearance (object images) and shape (parts-based segmentation).

Nevertheless, constructing a joint model of object images and shape masks poses significant difficulties as the conditional distribution of images given the shape masks are intrinsically multi-modal and full of ambiguities. In order to estimate the object mask $\mathbf{y}$ from an image $\mathbf{x}$, we instead propose to directly train the conditional models $p(\mathbf{y},\mathbf{h}|\mathbf{x})$ for RBMs (MMBM1, Figure 2.1(c)) and

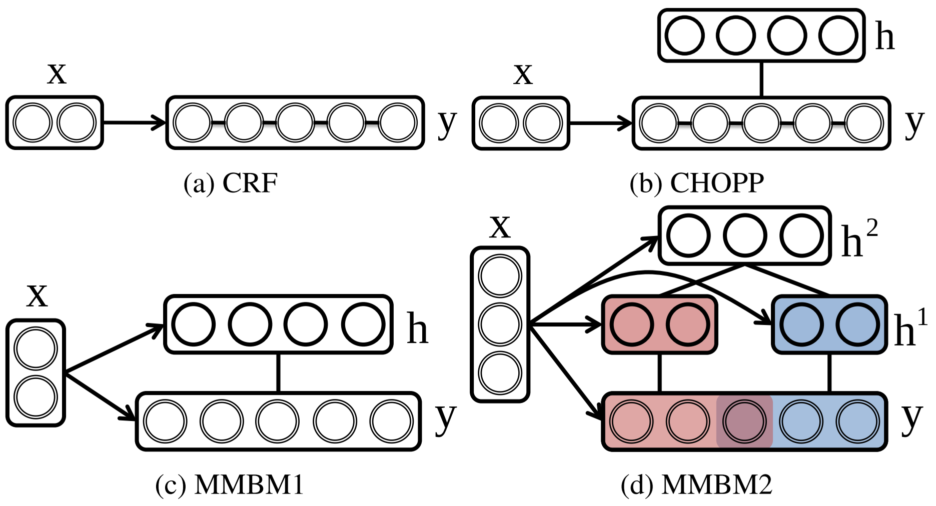


Figure 2.1. Comparing graphical models of MMBMs ((c) and (d)) with pairwise CRF (a) and CHOPP [1]. The edges mean full connections between two layers. In (d), the connections between $\mathbf{y}$ and $\mathbf{h}^1$ only involve the variables of the same color.

$p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2|\mathbf{x})$ for ShapeBMs (MMBM2, Figure 2.1(d)). In these conditional models, the activations of variables depend on the observations or image features, so the energy function of $p(\mathbf{y}, \mathbf{h}|\mathbf{x})$ can be represented by

$$\mathbf{E}(\mathbf{y}, \mathbf{h}, \mathbf{x}) = -\mathbf{y}^\top \mathbf{W}\mathbf{h} - \mathbf{h}^\top(\mathbf{V}^1\mathbf{x}^1 + \mathbf{c}) - \mathbf{y}^\top(\mathbf{V}^0\mathbf{x}^0 + \mathbf{b}), \tag{2.7}$$

while the energy function of $p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2|\mathbf{x})$ takes the form,

$$\begin{aligned}\mathbf{E}(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2, \mathbf{x}) = &-\mathbf{y}^\top \tilde{\mathbf{W}}^1\mathbf{h}^1 - \mathbf{h}^{1\top}\mathbf{W}^2\mathbf{h}^2 \\ &- \mathbf{h}^{1\top}(\mathbf{V}^1\mathbf{x}^1 + \mathbf{c}^1) - \mathbf{h}^{2\top}(\mathbf{V}^2\mathbf{x}^2 + \mathbf{c}^2) - \mathbf{y}^\top(\mathbf{V}^0\mathbf{x}^0 + \mathbf{b}).\end{aligned} \tag{2.8}$$

In the above equations, $\mathbf{x}^0$ represents low-level image features that indicate foreground and background assignments. The variable $\mathbf{x}^1$ represents features of object parts and $\mathbf{V}^1$ contains templates of object parts. The variable $\mathbf{x}^2$ describes the holistic object features, and $\mathbf{V}^2$ is composed of object templates of different poses and viewpoints. In these two models, we connect the observations $\mathbf{x}$ to both visible and hidden layers, which enables the direct inference pathway from image features to shapes.

2.3.3 MAP Inference

Given a set of image features $\mathbf{x}$, the most likely configuration of $\mathbf{y}$ is computed from

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}|\mathbf{x}). \tag{2.9}$$

In the proposed MMBM with single hidden layer, the marginal distribution $p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{h}} p(\mathbf{y}, \mathbf{h}|\mathbf{x})$ can be represented by its free energy form $\exp(-F(\mathbf{y}, \mathbf{x}))/\mathbf{Z})$, and

$$\begin{aligned} -F(\mathbf{y}, \mathbf{x}) =& \mathbf{y}^\top(\mathbf{V}^0\mathbf{x}^0 + \mathbf{b}) + \\ & \sum_j \log(1 + \exp(c_j + \mathbf{y}^\top \mathbf{W}_{\cdot j} + \mathbf{V}^1_{j\cdot}\mathbf{x}^1)). \end{aligned} \tag{2.10}$$

where $\mathbf{W}{\cdot j}$ and $\mathbf{W}_{j\cdot}$ denote j-th column and row of $\mathbf{W}$, respectively. As the partition function $\mathbf{Z}$ is constant given $\mathbf{x}$, the MAP inference in (2.9) is exactly equivalent to optimizing the free energy function

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} -F(\mathbf{y}, \mathbf{x}). \tag{2.11}$$

Note that the free energy $F(\mathbf{y}, \mathbf{x})$ is not a linear function of $\mathbf{y}$, and we need to take gradients to find the optimal $\hat{\mathbf{y}}$. However, the analytic free energy is not available in the MMBM with two hidden layers. We instead optimize the variational upper bound of log-likelihood $\log p(\mathbf{y}|\mathbf{x})$ using the EM algorithm in spirit similar to techniques that have been effectively applied to training generative BMs. However, in MMBMs, both visible $\mathbf{y}$ and hidden variables are conditioned on input variables $\mathbf{x}$. The conditional distributions $p(\mathbf{y}|\mathbf{h}^1, \mathbf{x})$, $p(\mathbf{h}^1|\mathbf{y}, \mathbf{h}^2, \mathbf{x})$ and $p(\mathbf{h}^2|\mathbf{h}^1, \mathbf{x})$ are likely highly peaked, if not unimodal, and thus they can be approximated by optimizing

$$\{\hat{\mathbf{y}}, \hat{\mathbf{h}}^1, \hat{\mathbf{h}}^2\} = \arg\max p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2|\mathbf{x}). \tag{2.12}$$

Similar to the block Gibbs sampling method, the independent property of conditional distributions induces an efficient Iterated Conditional Modes (ICM) algorithm (See Algorithm 1). The ICM algorithm also provides a good approximate solution to the free energy optimization problem in (2.11) and (2.10) for single layer MMBMs. Essentially, the second term in (2.10) can be approximated by

$$\begin{aligned} & \sum_j \log(1 + \exp(c_j + \mathbf{y}^\top \mathbf{W}_{\cdot j} + \mathbf{V}^1_{j\cdot}\mathbf{x}^1)) \approx \\ & \max_{\mathbf{h}}(\mathbf{c}^\top \mathbf{h} + \mathbf{y}^\top \mathbf{W}\mathbf{h} + \mathbf{h}^\top \mathbf{V}^1\mathbf{x}^1), \end{aligned} \tag{2.13}$$

which can be solved by the ICM algorithm.

Algorithm 1 MAP inference by the ICM algorithm.

```
1: Initialize $\mathbf{h}^1$
2: while do not converge do
3:     $\mathbf{h}^2 \leftarrow \max p(\mathbf{h}^2|\mathbf{h}^1, \mathbf{x})$
4:     $\mathbf{y} \leftarrow \max p(\mathbf{y}|\mathbf{h}^1, \mathbf{x})$
5:     $\mathbf{h}^1 \leftarrow \max p(\mathbf{h}^1|\mathbf{y}, \mathbf{h}^2, \mathbf{x})$
6: end while
```

2.4 Learning

Given a training set of object image-mask pairs $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$, we learn MMBMs for object segmentation. As the proposed learning algorithm can be applied to both MMBMs with single ($p(\mathbf{y}, \mathbf{h}|\mathbf{x}; \omega)$) or two hidden layers ($p(\mathbf{y}, \mathbf{h}^1, \mathbf{h}^2|\mathbf{x}; \omega)$), we denote the MMBM by a general form $p(\mathbf{y}, \mathbf{H}|\mathbf{x}; \omega)$ where $\mathbf{H} = \mathbf{h}$ for one single hidden layer or $\mathbf{H} = \{\mathbf{h}^1, \mathbf{h}^2\}$ for two hidden layers, and $\omega = \{\mathbf{W}^{1,2}, \mathbf{V}^{0,1,2}, \mathbf{c}^{1,2}, \mathbf{b}\}$ are the model parameters. The MMBMs consist of image-independent and image-dependent parts. We first initialize the image-independent part by generative pre-training, and then reformulate the joint learning problem into a max-margin optimization task which is solved effectively by a CCCP algorithm.

2.4.1 Pre-training

Generative pre-training $p(\mathbf{y}, \mathbf{H})$ is of crucial importance for the MMBM models. It provides the MMBM models with proper regularization between output and hidden variables, and feeds sensible hidden variables to the discriminative learning in the following stage. By omitting image-dependent components, the MMBM with one single hidden layer reduces to the RBM while the one with two hidden layers reduces to the ShapeBM. We thus can utilize the generative training algorithms of these methods. Indeed, the general training procedure of BMs requires minimizing the differences between the data-dependent and model-dependent expectations. We train the RBM by minimizing contrastive divergence [28]. For the ShapeBM, each layer is greedily trained.

Algorithm 2 Stochastic Gradient Descent algorithm for max-margin learning MMBMs.

```
1: Set $t = 0$, initialize $\omega_0, \alpha_0$ and define $\gamma$
2: while $t < T$ do
3:     Randomly select a training instance $(\mathbf{x}_i, \mathbf{y}_i)$
4:     Solve (2.16): $\mathbf{H}_i^* \leftarrow \max_{\mathbf{H}}[-E(\mathbf{y}_i, \mathbf{H}, \mathbf{x}_i; \omega_t)]$
5:     Solve (2.17): $\hat{y}_i, \hat{\mathbf{H}}_i \leftarrow \max_{\mathbf{y},\mathbf{H}}[-E(\mathbf{y}, \mathbf{H}, \mathbf{x}_i; \omega_t) + \Delta(\mathbf{y}, \mathbf{y}_i, \mathbf{H}, \mathbf{H}_i^*)]$
6:     Update $\omega_{t+1} \leftarrow (1 - \alpha_t\gamma)\omega_t + \alpha_t(\frac{\partial E(\hat{y}_i, \hat{\mathbf{H}}_i, \mathbf{x}_i; \omega)}{\partial \omega} - \frac{\partial E(\mathbf{y}_i, \mathbf{H}_i^*, \mathbf{x}_i; \omega)}{\partial \omega})$
7:     Decrease $\alpha_t$
8: end while
```

2.4.2 Max-Margin Learning

To generate accurate prediction on test images, we seek for the parameters ω that assign training labels $\mathbf{y}_i$ a greater than or equal log-likelihood of any other labeling $\mathbf{y}$ for instance i,

$$\log p(\mathbf{y}_i, \mathbf{H}|\mathbf{x}_i; \omega) \geq \log p(\mathbf{y}, \mathbf{H}|\mathbf{x}_i; \omega), \forall \mathbf{H}, \forall \mathbf{y}, \forall i. \tag{2.14}$$

We can cancel the partition function $\mathbf{Z}$ for both sides of (2.14), and express the constraints by energies,

$$-E(\mathbf{y}_i, \mathbf{H}, \mathbf{x}_i; \omega) \geq -E(\mathbf{y}, \mathbf{H}, \mathbf{x}_i; \omega), \forall \mathbf{H}, \forall \mathbf{y}, \forall i. \tag{2.15}$$

We refer the left term of (2.15) as data-dependent energy and the right term as model-dependent energy. Since the number of constraints in (2.15) is exponentially large, we look for the hidden variables $\mathbf{H}_i^*$ that best explain the training instance $(\mathbf{x}_i, \mathbf{y}_i)$ in the data-dependent energy

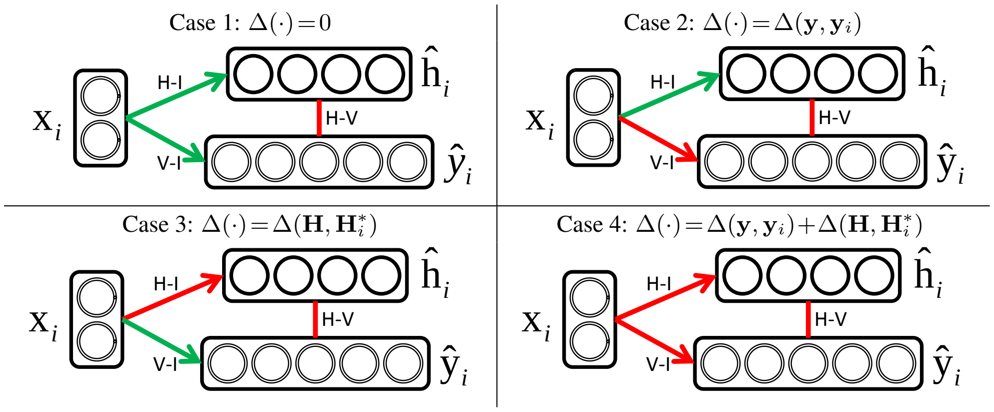
$$\mathbf{H}_i^* = \arg\max_{\mathbf{H}} -E(\mathbf{y}_i, \mathbf{H}, \mathbf{x}_i; \omega). \tag{2.16}$$

For the model dependent energy, we compute the best prediction from $\mathbf{x}_i$ by augmenting an energy margin $\Delta(\mathbf{y}, \mathbf{y}_i, \mathbf{H}, \mathbf{H}_i^*)$,

$$\{\hat{y}_i, \hat{\mathbf{H}}_i\} = \arg\max_{\mathbf{y},\mathbf{H}} -E(\mathbf{y}, \mathbf{H}, \mathbf{x}_i; \omega) + \Delta(\mathbf{y}, \mathbf{y}_i, \mathbf{H}, \mathbf{H}_i^*). \tag{2.17}$$

These two decoding problems (2.16) and (2.17) can be solved efficiently by the ICM algorithm in Algorithm 1 where the only difference is to initialize with a random $\mathbf{H}$.

To deal with noisy training image data, we relax the margin constraints by introducing slack variables ξ_i. Thus, we formulate the MMBM learning with the following max-margin objective



Case 1: $\Delta(\cdot) = 0$
Case 2: $\Delta(\cdot) = \Delta(\mathbf{y}, \mathbf{y}_i)$
Case 3: $\Delta(\cdot) = \Delta(\mathbf{H}, \mathbf{H}_i^*)$
Case 4: $\Delta(\cdot) = \Delta(\mathbf{y}, \mathbf{y}_i) + \Delta(\mathbf{H}, \mathbf{H}_i^*)$
x_i
$\hat{h}_i$
$\hat{y}_i$
H-I
H-V
V-I

Figure 2.2. Comparing margin functions. Cases 1-4 illustrate learning single-layer MMBMs with four kinds of margin functions. The particular margin functions induce the red connections between any two layers dominate the energy loss during learning while leaving the green connections unoptimized. Best viewed in color.

function,

$$\begin{aligned} &\min_{\omega} \frac{\gamma}{2}\|\omega\|^2 + \sum_i \xi_i, \quad \text{s.t.} \\ -E(\mathbf{y}_i, \mathbf{H}_i^*, \mathbf{x}_i; \omega) \geq &\max_{\mathbf{y},\mathbf{H}} [\Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{H}_i^*, \mathbf{H}) - E(\mathbf{y}, \mathbf{H}, \mathbf{x}_i; \omega)] \\ &- \xi_i, \xi_i \geq 0, \forall i, \\ \text{where} \quad &\mathbf{H}_i^* = \arg\max_{\mathbf{H}} - E(\mathbf{y}_i, \mathbf{H}, \mathbf{x}_i; \omega). \end{aligned} \tag{2.18}$$

This formulation is equivalent to minimizing the loss function,

$$\begin{aligned} \min_{\omega} \frac{\gamma}{2}\|\omega\|^2 + \sum_i \max_{\mathbf{y},\mathbf{H}} [\Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{H}_i^*, \mathbf{H}) - E(\mathbf{y}, \mathbf{H}, \mathbf{x}_i; \omega) + \\ E(\mathbf{y}_i, \mathbf{H}_i^*, \mathbf{x}_i; \omega)]. \end{aligned} \tag{2.19}$$

To optimize the loss function (2.19), we initialize the parameters ω_0 with pre-trained $\mathbf{W}^{1,2}$, $\mathbf{c}^{1,2}$, $\mathbf{b}$ and random matrices $\mathbf{V}^{1,2}$. We develop a stochastic gradient descent algorithm (See Algorithm 2) for optimizing (2.19) by applying the Concave-Convex Procedure [20]. Note that it is easy to compute the gradients of energy functions with respect to ω as both data energy $E(\hat{y}_i, \hat{\mathbf{H}}_i, \mathbf{x}_i; \omega)$ and model energy $E(\mathbf{y}_i, \mathbf{H}_i^*, \mathbf{x}_i; \omega)$ are linear functions of parameters ω given fixed hidden and output variables.

Comparing Margin Functions. Choosing a proper margin penalty function $\Delta(\cdot)$ is crucial to effective learning. Taking the single layer MMBM as an example, we find its energy function consists of three components: hidden-visible interaction (H-V), hidden-image interaction (H-I) and visible-image interaction (V-I), which correspond to the three kinds of edges in the graphical model of MMBMs,

$$E(\mathbf{y}, \mathbf{h}, \mathbf{x}) = -\underbrace{\mathbf{y}^\top \mathbf{W} \mathbf{h}}_{\text{H-V}} - \underbrace{\mathbf{h}^\top (\mathbf{V}^1 \mathbf{x}^1 + \mathbf{c})}_{\text{H-I}} - \underbrace{\mathbf{y}^\top (\mathbf{V}^0 \mathbf{x}^0 + \mathbf{b})}_{\text{V-I}}, \tag{2.20}$$

We analyze four cases of $\Delta(\cdot)$ (See Figure 2.2) and evaluate their performance in the experiments.

Case 1: $\Delta(\cdot) = 0$. If we set $\Delta(\cdot) = 0$, then the loss function in (2.19) reduces to the perceptron loss used in [29]. As the data-dependent and model-dependent energies remain the same form, there exist several possibilities that can explain the perceptron loss, considering the potential combinations of three components. For example, learning with $\Delta = 0$ may end up with a strong H-V component but weak H-I and V-I components, as the H-V component is pre-trained. This result is clearly deficient for prediction.

Case 2: $\Delta(\cdot) = \Delta(\mathbf{y}, \mathbf{y}_i)$. If we set $\Delta(\cdot) = \Delta(\mathbf{y}, \mathbf{y}_i)$, then the loss function in (2.19) is closely related to the one used in latent Structured SVM [20]. The energy margin $\Delta(\mathbf{y}, \mathbf{y}_i)$ only depends on $\mathbf{y}$ so that the V-I component will be better constrained to dominate the energy loss between the data energy and the augmented model energy. Considering the pre-trained H-V component, we may obtain strong H-V and V-I components but a weak H-I component. However, the H-I and V-I components take different input features and should be complementary to each other. The unoptimized H-I component very likely constrains the model generalizability to unseen data.

Case 3: $\Delta(\cdot) = \Delta(\mathbf{H}, \mathbf{H}_i^*)$. If we set $\Delta(\cdot) = \Delta(\mathbf{H}, \mathbf{H}_i^*)$, then the loss function in (2.19) indirectly corresponds to the output through hidden variables. That is, the H-V component functions as clustering. The margin on hidden variables essentially encourages the H-I component to correctly predict the cluster labels $\mathbf{H}_i^*$, i.e., the hidden variables that best explain the training instance $(\mathbf{x}_i, \mathbf{y}_i)$. Thus, the energy difference is likely dominated by the H-I and H-V components, which leaves the V-I component unoptimized. This approach has the same generalizability problem as Case 2.

Case 4: $\Delta(\cdot) = \Delta(\mathbf{y}, \mathbf{y}_i) + \Delta(\mathbf{H}, \mathbf{H}_i^*)$. Based on the above analysis, we use $\Delta(\cdot) = \Delta(\mathbf{y}, \mathbf{y}_i) + \Delta(\mathbf{H}, \mathbf{H}_i^*)$ as the margin penalty function. Since $\Delta(\mathbf{y}, \mathbf{y}_i)$ and $\Delta(\mathbf{H}, \mathbf{H}_i^*)$ are absorbed into the V-I component and H-I component, respectively, all three components are optimized during learning.

2.5 Experiments

2.5.1 Datasets

Penn-Fudan Pedestrians This dataset [30] consists of 170 images with bounding box annotations and ground truth foreground-background segmentation masks. The images all include one or more pedestrians. For our experiments we extracted 423 patches, each adjusted to include one person only. We resize the patches to an uniform size of 32×64 pixels, cropping the original image so that we can keep the original aspect ratio while resizing them.

In order to increase the number of training and test samples, we subsequently mirror all patches, resulting in 846 samples, some of which with severe occlusions. We then select 400 samples for training and use the rest for tests. The training-test split is done randomly except for keeping the original images in the same set as their mirrored pairs.

Weizmann Horses. This dataset [23] contains 328 horse images, with a high variability of poses and scales. Before processing, we resize every image to 128x128, padding images with different aspect ratios with mirrored versions of the image itself. To get comparable results to [1], we calculate 32x32 foreground-background segmentation masks with all of our models. Also, we use their training-test split (into 200 training and 128 test images).

Caltech-UCSD Birds 200. The dataset [25] includes 6033 images of 200 bird species, each image usually including one dominant bird in the scene. The images are annotated with a bounding box and a coarse-grained segmentation mask. As the accuracy of this isn't sufficient to evaluate our segmentation methods, we manually annotate these images with accurate masks (available on the website `https://eng.ucmerced.edu/people/jyang44`). We crop 6033 bird patches and the corresponding segmentation masks from bounding boxes, and resize the image patches to 128×128 pixels. We use the same training/test partition as in [25], i.e., 3000 samples for training and the rest for tests.

2.5.2 Implementations

Architectures. For the MMBM with a single hidden layer (MMBM1) and RBM, we use 500 hidden units $\mathbf{h} \in \{0,1\}^{500}$. For the MMBM with two hidden layers (MMBM2), we use 500 hidden units in the first layer $\mathbf{h}^1 \in \{0,1\}^{500}$, and 200 hidden units in the second layer $\mathbf{h}^2 \in \{0,1\}^{200}$. For the birds and the horses, each mask is partitioned into 2×2 four patches $\{\mathbf{y} = \vee \mathbf{y}_k, k = 1, \ldots, 4, \mathbf{y}_k \in \{0,1\}^{36\times 36}\}$ with 8 pixels overlapping between adjacent patches, such that each part is connected to 125 hidden units in the first layer $\mathbf{h}^1$. For the pedestrians, we also use four patches $\{\mathbf{y}=\vee \mathbf{y}_k, k=1, \ldots, 4, \mathbf{y}_k \in \{0,1\}^{22\times 32}\}$ but in a 4x1, vertical organization with 14 pixel overlaps between neighbors.

Features. One of the advantages of the proposed method is that it can handle a diverse set of features: local descriptors can be connected to the visible layer while features covering larger image areas are better suited as conditionals for one of the hidden layers.

For MMBM1, we use two sets of features: $\mathbf{x}^0$ for the visible and $\mathbf{x}^1$ for the hidden layer. For $\mathbf{x}^0$, we first segment the image into superpixels using the gPb algorithm [22]. For each superpixel, we compute dense SIFT, color and contour histograms. The histograms of densely sampled SIFT words are computed by using a codebook of size 512 and the locality-constrained linear coding method [31]. The color histograms of RGB values are computed from a codebook of size 128, and finally, the contour histograms are computed from the oriented gPb edge detector responses [22]. For per-pixel visible features we simply use those of the superpixel containing the pixel in question.

For the hidden layers of MMBM1, we use the HOG descriptors for the entire input image as $\mathbf{x}^1$. For MMBM2, the features $\mathbf{x}^2$ for the top layer is calculated the same way, while for the middle layer feature vector $\mathbf{x}^1$ we use the HOG descriptors for the four patches.

Training. For the MMBM1, we run 2000 epochs with 100-sample mini-batches in the generative training phase (RBM training). For the MMBM2, we run 2000 epochs for the first layer pre-training in the generative training phase (ShapeBM training) and 1000 epochs for the second layer pre-training. In addition, we run 5 cycles in the max-margin training phase in both cases. We set the learning rate $\alpha_0 = 0.001$ and the constant $\gamma = 0.01$ for all the experiments. The MATLAB source code and the labeled datasets will be made available for research purposes.

Baseline. We study two discriminative models for comparison: a superpixel based CRF model using bottom-level features $\mathbf{x}^0$ and Compositional High Order Potentials (CHOPPs) [1]. For the CRF model $p(\mathbf{y}|\mathbf{x}^0)$, we use the implementation in [1].

For CHOPPs, we used the code provided by the authors for the inference but we didn't get the same results, likely due to differences in our unary / pairwise potential generation code. To make the comparison fair, in the experiments we used the same unary features as in our MMBM implementation instead. As Table 2.2 shows, this improved their results compared to the original published in [1].

Unlike the combined RBM-CRF models of [1] and [26], our model doesn't have pairwise weights in the visible layer. For a better comparison with these models, we also ran Graph Cut on the output mask, using the probabilities given by the model as unary potentials and a pairwise term taken from [32], based on the magnitude of the gradients of color channels. We report results for both the original and refined masks.

2.5.3 Results

We use two metrics for performance evaluation: the average pixel accuracy (AP) of foreground and background classification and the foreground intersection-over-union score (IoU) of the entire test set [2]. We first present segmentation results on the Penn-Fudan Pedestrians in Table 2.1. Overall, the MMBM1 (76.92% IoU, Case 4) and MMBM2 (77.30% IoU, Case 4) outperform the CRF (68.35% IoU) and CHOPPs (71.33% IoU) algorithms. The results show that the MMBMs are effective models for object segmentation by integrating image features and a strong shape prior. Also, as the last two rows of the table indicate, introducing pairwise constraints further improves results.

Our results for Weizmann horses are shown in Table 2.2. Again, both the advantage of augmenting the loss function with multiple margins and the benefits of using a two-layered architecture are demonstrated. Also, comparisons using different margin functions (Cases 1-4) for the MMBM1 model demonstrate the importance of a max-margin formulation with multiple margins for output prediction. By using margin functions (MMBM1 Cases 2-4), we obtain 19% AP improvement and more than 30% IoU improvement over the non-margin (perceptron loss) algorithm in Case 1 of the

[2]The IoU score is defined as $\frac{|\mathbf{Y} \bigcap \hat{\mathbf{Y}}|}{|\mathbf{Y} \bigcup \hat{\mathbf{Y}}|}$, where $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ are the sets of ground truth and predicted foreground pixels.

Table 2.1. Results on the Penn-Fudan Pedestrians dataset.

		AP	IoU
CRF		84.87	68.35
CHOPPs [1]		86.55	71.33
MMBM1	Case 1	82.66	64.80
	Case 2	85.27	69.20
	Case 3	83.35	65.78
	Case 4	89.91	76.92
MMBM2		89.74	77.30
MMBM1 Case 4 w/ GC		90.42	77.97
MMBM2 Case 4 w/ GC		**90.77**	**79.42**

Table 2.2. Results on the Weizmann Horses dataset.

		AP	IoU
CRF		87.46	67.44
Bo and Fowlkes [24]		77.2	N/A
CHOPPs [1]		88.67	71.60 (69.90 in [1])
MMBM1	Case 1	70.59	38.01
	Case 2	85.87	62.97
	Case 3	85.37	59.35
	Case 4	89.43	69.59
MMBM2		89.80	72.09
MMBM1 Case 4 w/ GC		90.62	74.12
MMBM2 Case 4 w/ GC		**90.71**	**75.78**

MMBM1. The best results for the MMBM1 (89.43% AP, 69.59% IoU) from Case 4 indicate that the combining multiple margin functions $\Delta(\cdot) = \Delta(\mathbf{y}, \mathbf{y}_i) + \Delta(\mathbf{H}, \mathbf{H}_i^*)$ alleviates degenerating effects by providing stronger constraints. Our results on other datasets also strengthen this observation. The two-layer hierarchical hidden architecture also helps generating better results than a single hidden layer, as shown in the Case 4 of MMBM2 (89.80%AP, 72.09% IoU) over MMBM1.

In addition to the comparison to CRF and CHOPPs, for this dataset we also added the results from [24]. Their aim was to identify body parts and got the foreground-background segmentation as a byproduct.

Finally, the segmentation results on the Caltech-UCSD Birds 200 dataset are presented in Table 2.3. Different from pedestrians and horses, this dataset has large shape variations but more distinct appearances (e.g., color, textures). Thus, the appearance-based CRF model performs less well. Similar is the case of CHOPP: as its hidden nodes are not directly connected to image features, so they can only refine and correct the shape of results that are mostly right just based on

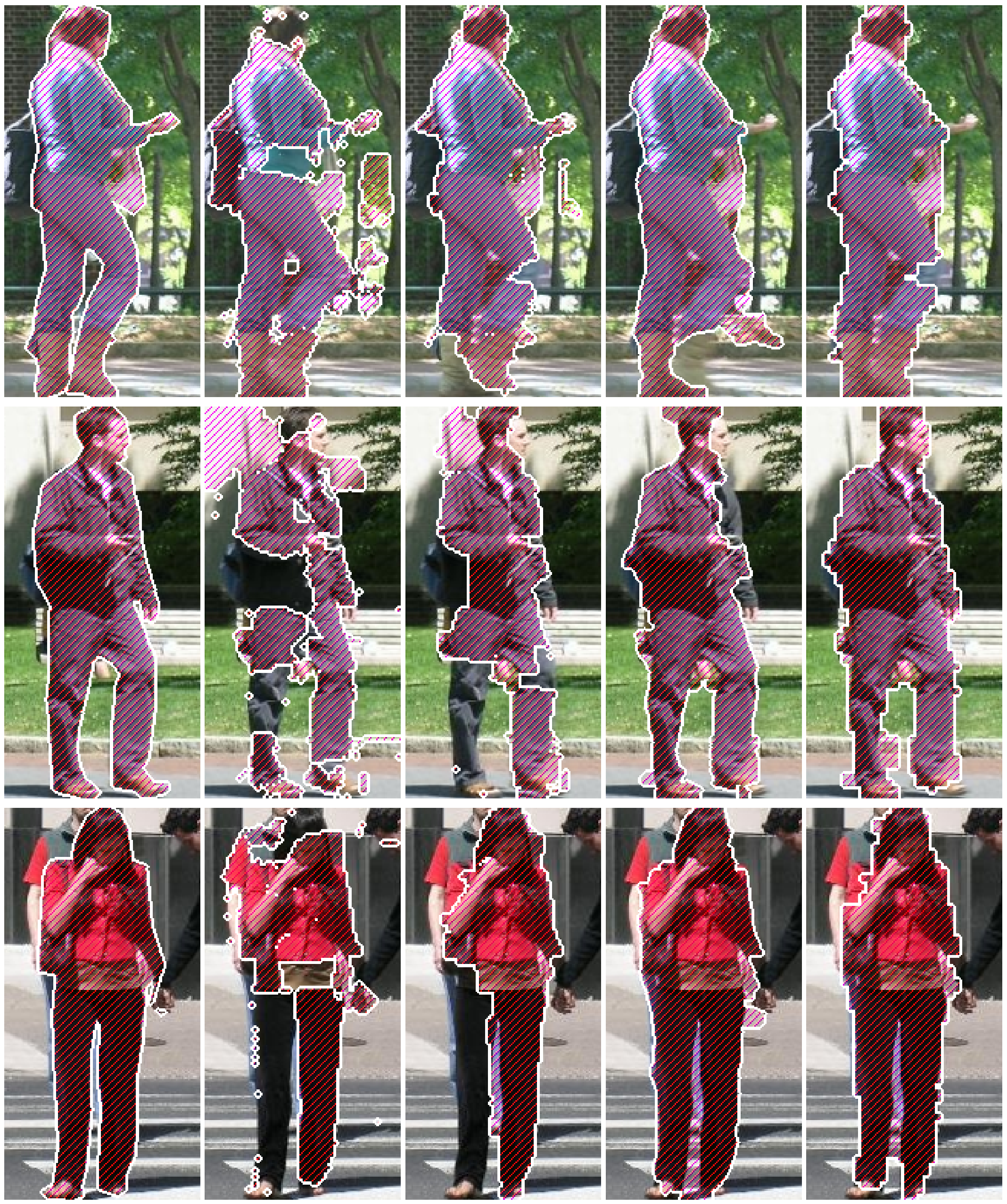

(a) GT (b) CRF (c) CHOPP (d) MMBM1 (e) MMBM2

Figure 2.3. Qualitative results on the Penn-Fudan Pedestrians, where segmentation results (shown with white contours) are overlaid with the input images.

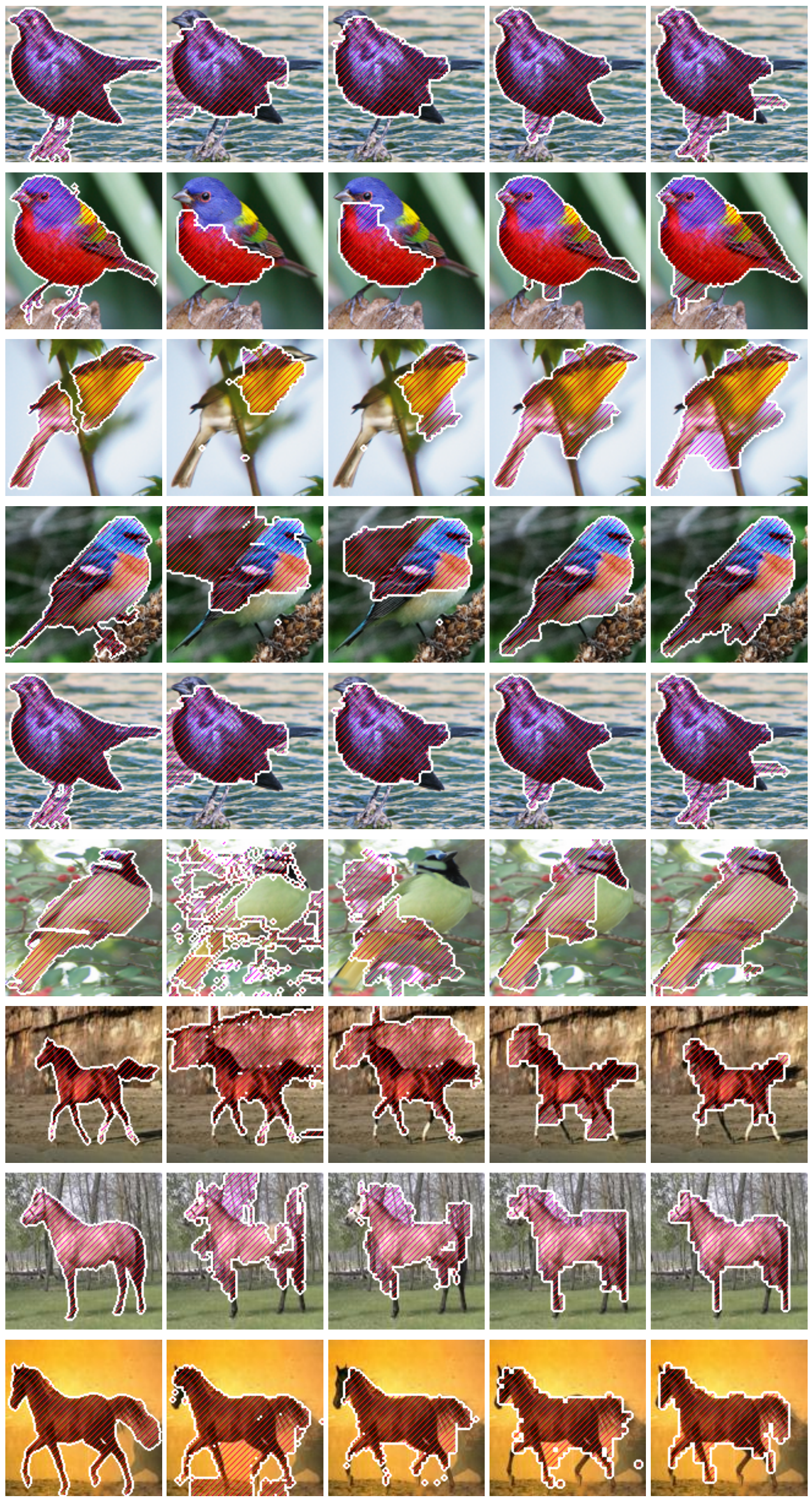


Figure 2.4. Qualitative results on the Caltech-UCSD Birds and Weizmann horses where segmentation results (shown with white contours) are overlaid with the input images.

Table 2.3. Results on the Caltech-UCSD Birds 200 dataset.

		AP	IoU
CRF		83.50	38.45
CHOPPs [1]		74.52	48.84
MMBM1	Case 1	80.96	60.37
	Case 2	87.73	72.45
	Case 3	75.73	63.22
	Case 4	88.07	72.96
MMBM2		86.38	69.87
MMBM1 Case 4 w/ GC		**90.42**	**75.92**
MMBM2 Case 4 w/ GC		90.77	72.40

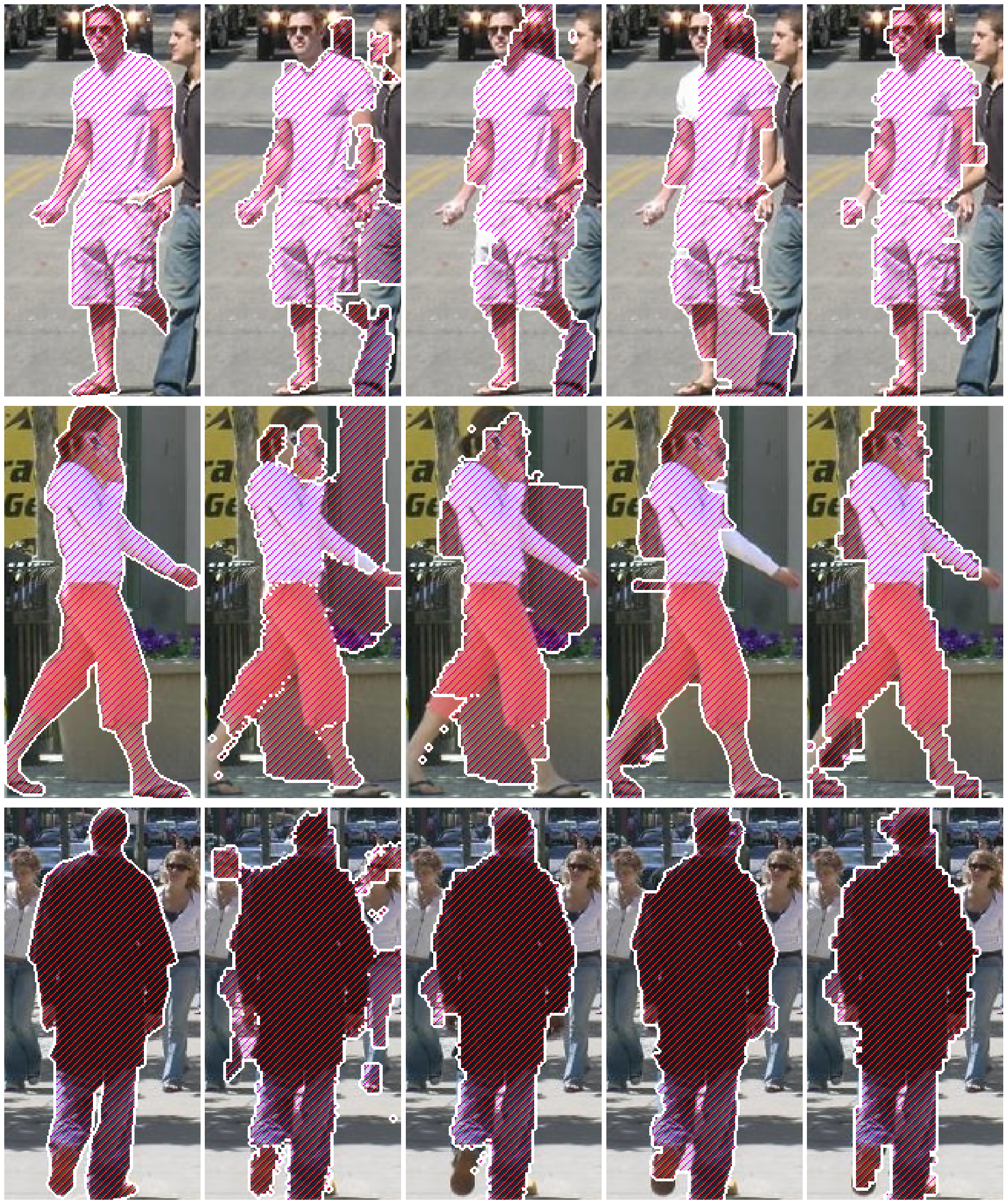
local, visible-layer features, which is hard to accomplish on this dataset. In contrast, the features-to-hidden connections in the MMBM models make it possible to exploit global shape information even without reliable local features. The results, similar to the observations on the other two datasets for evaluating different margin functions demonstrate the significance of max-margin formulation and combining margin functions (Case 4). In the bird data, we observe better performance by using just one hidden layer compared to using the two-layered MMBM2 model. A possible reason is that while the weight replication for the four windows in MMBM2 is beneficial when given a small number of training samples (such as for horses and pedestrians), but for larger datasets we can learn a better prior using simple architectures (RBMs) with more parameters from the data.

We present some qualitative results in Figure 2.3, from which we can see more directly the importance of features-to-hidden connections for shape prediction. For example, CRF finds the most colorful parts of birds, which is corrected by CHOPP to be shaped more birdlike, but it's only MMBMs that discover the entire bird well.

2.6 Conclusions

In this chapter, we propose MMBMs for structured output prediction problems and investigate two variants of MMBMs with single and two hidden layers for object segmentation. Instead of using BMs as shape priors, we build connections between input observations with hidden variables that opens an inference pathway from image features to object shapes. We derive a simple yet efficient ICM algorithm for MAP inference. We formulate MMBMs with a max-margin objective function for discriminative training, and discuss four margin functions as well as their effects on learning performance. The results on horses, pedestrians, birds datasets show that our algorithms perform favorably against the state-of-the-art methods.

In experiments, we have found that the pairwise edge potentials can after all improve the segmentation quality, given the predicted shapes from our models.



(a) GT (b) CRF (c) CHOPP (d) MMBM1 (e) MMBM2

Figure 2.5. More results on the Penn-Fudan Pedestrians, where segmentation results (shown with white contours) are overlaid with the input images.

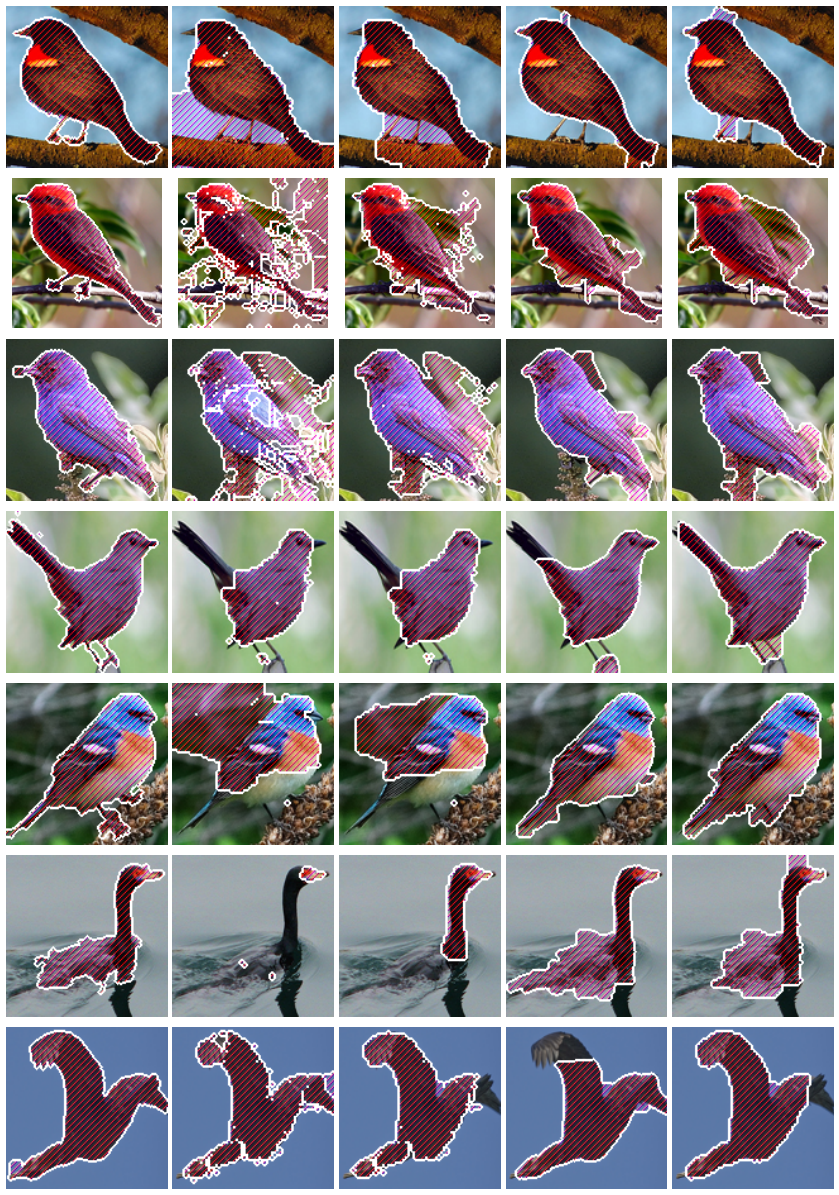


Figure 2.6. More results on the Caltech-UCSD Birds where segmentation results (shown with white contours) are overlaid with the input images.

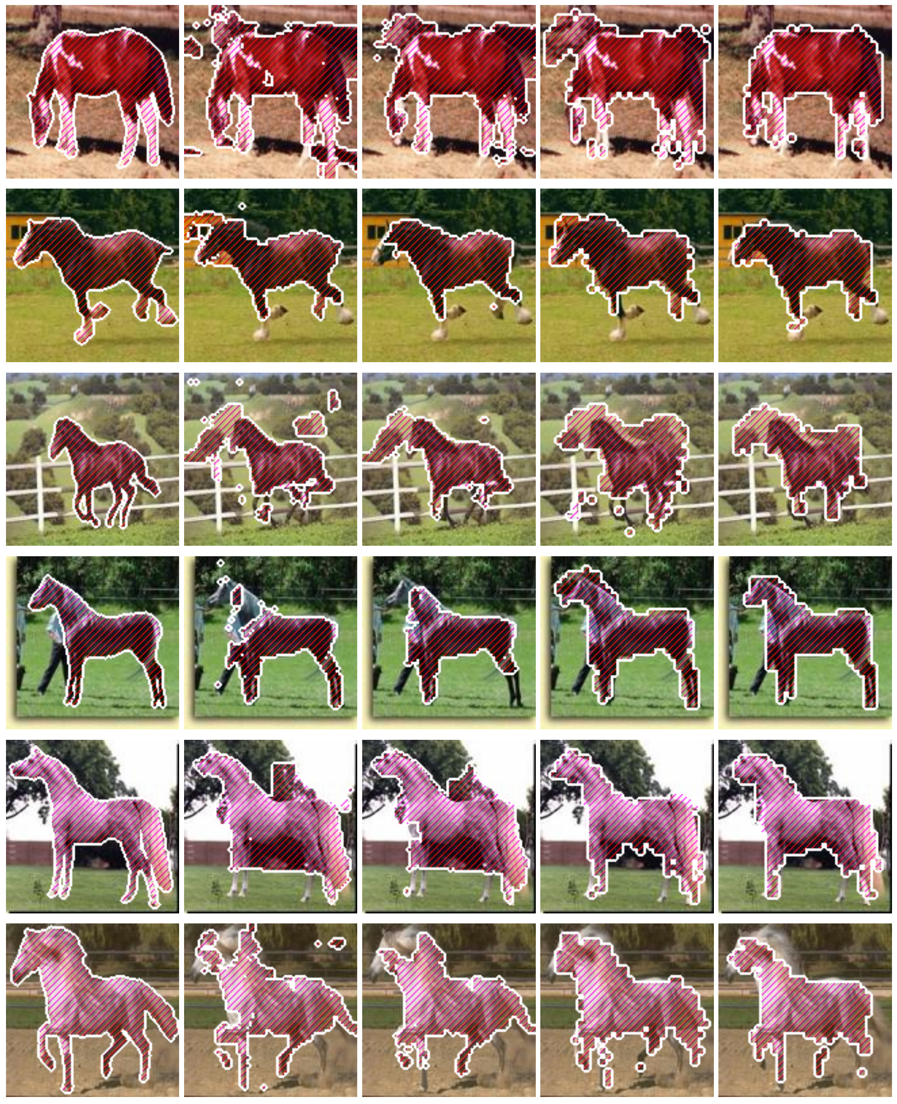


Figure 2.7. More results on Weizmann horses where segmentation results (shown with white contours) are overlaid with the input images.

Chapter 3

Global Masks with Convolutional Refinement

3.1 Introduction

Recent research (for example, the deep convolutional architecture presented by Krizhevsky *et al*. in [33]) has shown that convolutional neural networks present an efficient way of extracting features from images, with the represented complexity increasing towards the upper layers. CNNs present a good alternative to traditional features, such as color histograms, HOG or SIFT, as a basis of further processing. Accordingly, a considerable amount of work has been done on various applications, including recognition, detection [4], segmentation [27] and image super-resolution [34].

In this chapter, we present a simple but effective method of solving the same segmentation task as in Chapter 2, with a different toolset. Instead of hand-crafted features, we are using a network following the architecture presented in [33], pre-trained on the ImageNet dataset. Using these, we learn the connection between shapes, pixel masks and image features in a fully supervised way.

Thus, our architecture can be understood as consisting of two components: image analysis and mask synthesis. The analysis side of the network processes raw visual information and determines, for example, the general shape of the object in question. These features are being used by the synthesis side to construct and refine a pixel-wise segmentation mask.

Typically, nodes in the earlier layers of the analysis side represent simpler visual features, such as edges or colors, while the ones in later layers indicate the presence of more complicated structures, such as human faces or legs of horses. However, this presents a tradeoff: by collecting visual information from larger receptive fields, layers late in the network lose location information that would be essential for accurate segmentation.

In contrast to the feature extraction stage, our synthesis network uses the opposite architecture. First, a coarse, global mask is inferred, using inputs from the later, more high-level layers of the analysis network. This is then used as one of the inputs for the more fine-grained second layer, providing the final output. For this step, we are using the early-stage visual information from the analysis side, to better localize parts of the mask.

3.2 Related work

One of the works inspiring the approach described in this chapter is [34] by Dong *et al.* It is taking the problem of super-resolution, a problem whose solutions traditionally made use of sparse coding, dictionary learning and patch matching, techniques that might feel substantially different from the approach taken by deep learning (that is, "represent everything by either convolutional or fully-connected layers"). Nevertheless, the paper presents a network that successfully formulates the patch matching process in the form of convolutional and fully connected layers, with excellent results. This readily raises the question whether the same could be done in the area of image segmentation?

A possible example to follow is that of the structured forest-based edge detector of [35] by Dollár *et al.* Although it addresses a slightly different problem, one of their intermediate stages in fact does involve foreground-background mask patches. Of course, they do not deal with the question of object shapes, as their aim is general edge detection. Nevertheless, it is a good starting point, if not for shape priors but for their refinement, a task for which last chapter's MMBM model didn't exhaust all improvement possibilities.

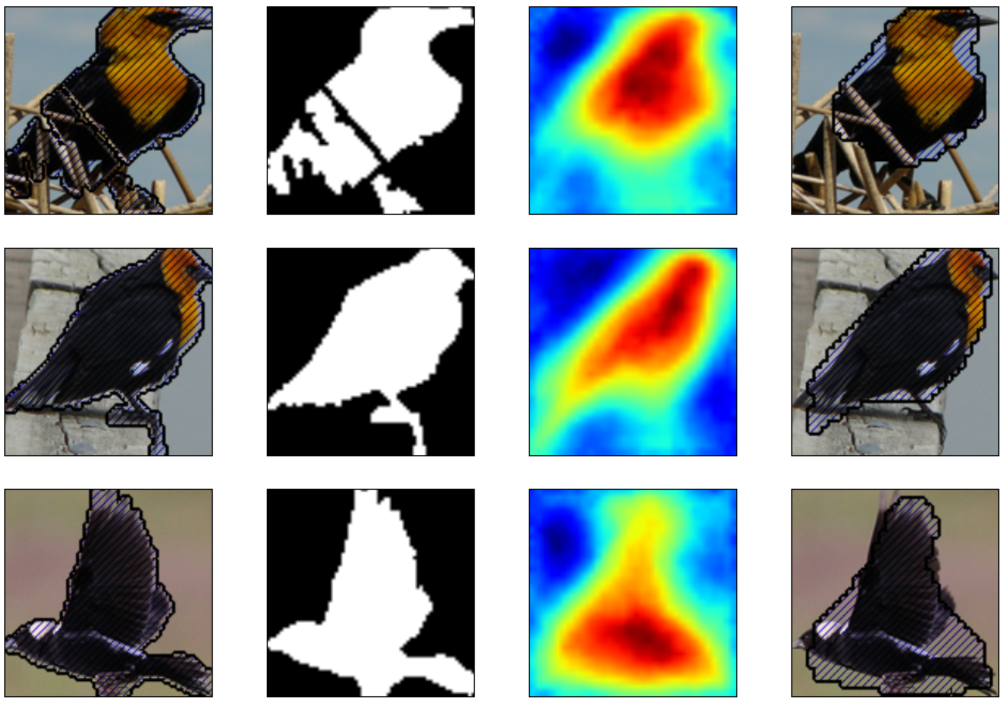

(a) Ground truth (b) GT mask (c) Global prior (d) Global result

Figure 3.1. Example results from the global model on the Caltech-UCSD Birds dataset.

3.3 Architecture

3.3.1 Shapes

When looked at from an architecture point of view, ignoring the details of training and inference, the one hidden layer network presented in Chapter 2 has a surprisingly simple structure. The feature vector $\mathbf{x}$ is obtained using HOG, color and shape histograms, then fed to a fully connected layer. The hidden nodes are in turn connected to an output raster, producing the binary output mask. This suggests the question: how does a network with the same structure network perform with a simple, backpropagation-based, SGD training process?

According to our first experiments, the answer is that although it does not perform as well, it is still comparable in performance, even though both the training time and the amount of custom code are significantly smaller than of its more sophisticated version.

As it can be seen on Figure 3.2, the inputs of this network are obtained as one of the convolutional outputs of the analysis stage. Given the image $\mathbf{I}$, we feed it to the pre-trained convolutional layers C_1 to C_5. We mostly leave the architecture of [33] unchanged, always using the last output from the certain layer (e.g. if there was a normalization component, we take its input, if not, we use the pooling output instead, etc.). It is this way we obtain the analysis outputs $\mathbf{f}_1$ to $\mathbf{f}_5$. Since these feature detectors (especially the early ones) are unlikely to be dataset-specific, we are using a set of weights pretrained on ImageNet throughout our experiments, kept static throughout the learning process.

The original architecture performs pooling after the first layer, resulting in the feature maps $\mathbf{f}_1$ having half the resolution of what would be needed eventually. Thus, we also perform the same pooling operation with a stride of 1 instead of 2. To compensate for the change of resolution, local response normalization is done on this result not with a kernel size of 5 but 9, resulting in the higher-resolution first layer feature maps $\mathbf{f}^{(1\mathrm{HR})}$.

Due to its simplicity, unlike Chapter 2, this approach still does not make use of several pieces of additional information. It does not perform explicit, class-agnostic image segmentation or edge detection at all, with its only way of representing the input being the several hundred hidden nodes before the output layer, only capable of representing global information. Due to this, it cannot represent small local variations in shape, resulting in the blurriness of output masks observable on Figure 3.1.

3.3.2 Patches

To make up for this shortcoming, we borrow the approach of Dong *et al.*'s deep superresolution approach [34]. Conceptually, we are learning to associate different patterns of output mask patches with image features, first describing them with local, linear features, then adding a nonlinear second layer to predict the actual output center pixel at that location.

As for the implementation, this step is also performed by convolutional layers. The patch-match component consists of two of them. The first uses a 5×5 kernel to collect local information, followed by a ReLU and a local response normalization layer. Latter is operating over 5×5 windows, dividing values by $(1 + (\alpha/n) \sum_i x_i^2)^\beta$, where $\alpha = 0.0001$ and $\beta = 0.75$ are parameters, $n = 25$

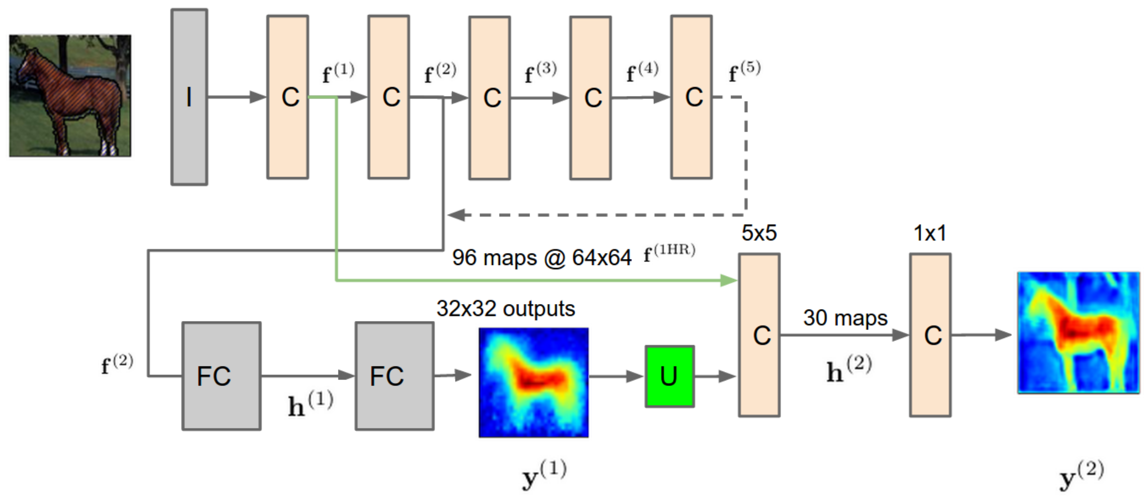


Figure 3.2. Architecture of our segmentation network, with the image analysis stage on the top and the mask synthesis stage at the bottom. “C” denotes convolutional layers, “U” is an upscaling operation, while “FC” stands for “fully connected”.

is the number of pixels in the receptive field and the x_i the output values from the preceding ReLU layer.

Above operations result in the feature maps $\mathbf{h}^{(2)}$, with 30 feature maps. This is summarized by a local linear transformation, described by a 1×1 convolution, providing the refined mask output $\mathbf{y}^{(2)}$.

Unfortunately, the flexibility of local patch matching comes at a cost: if we base our output only on local features, the result might be able to model local edges well, but it would not be aware of the global context and object shape. Therefore, in order to combine local edge information with the already known global shape prior, we connect not only the image feature maps $\mathbf{f}^{(2)}$ but also an upscaled version of $\mathbf{y}^{(1)}$ of the prior generated by our first layer to the patch predictors. That is, the upscaled $\mathbf{y}^{(1)}$ is added to $\mathbf{f}^{(2)}$ as an extra feature map. This allows the refinement process to ignore locally promising regions that do not fit the predicted shape while still predicting local edge shapes more accurately.

As it can be seen on Figure 3.3, the shape prior $\mathbf{y}^{(1)}$ localizes the object and also successfully predicts its overall shape, but produces blurry predictions regarding the exact boundaries. This is successfully corrected after the patch match layer, producing the output $\mathbf{y}^{(2)}$.

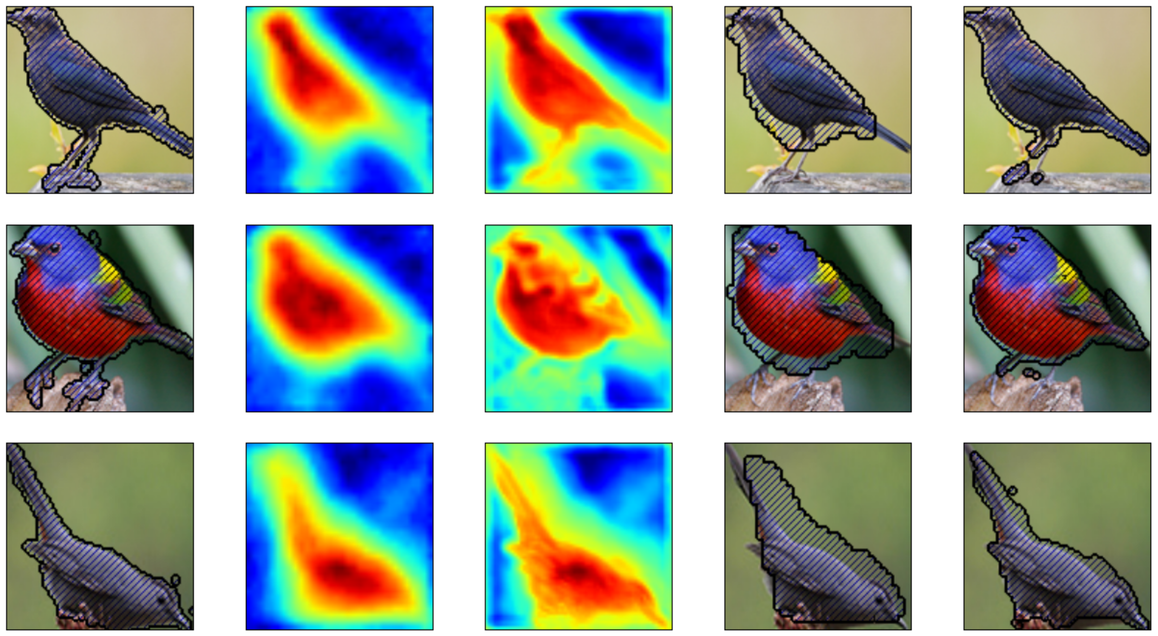


Figure 3.3. Example results from the convolutional refinement on the Caltech-UCSD Birds dataset.

The reason for the presence of the upscale operation "U" is that we are generating a shape prior whose resolution is half of that of the patch-matched $\mathbf{y}^{(2)}$ so we need to upscale it before feeding it to the patch match layer. Due to the low spatial accuracy of the shape prior, this lets us decrease the number of parameters in the fully connected layers without reducing accuracy while also speeding up computations.

3.4 Experiments

3.4.1 Implementation details

Code. We use the Caffe deep learning framework [36] to implement the core of our method; we also implement additional layers (in-place reshape and upscale) that were not included in the framework yet. Most of the processing is done on an nVidia GeForce GTX Titan GPU, with the most important constraint being the amount of available video memory (memory usage being on the order of 2GB while training).

Caffe takes networks in Google's Protocol Buffer format. Since for our experiments we need

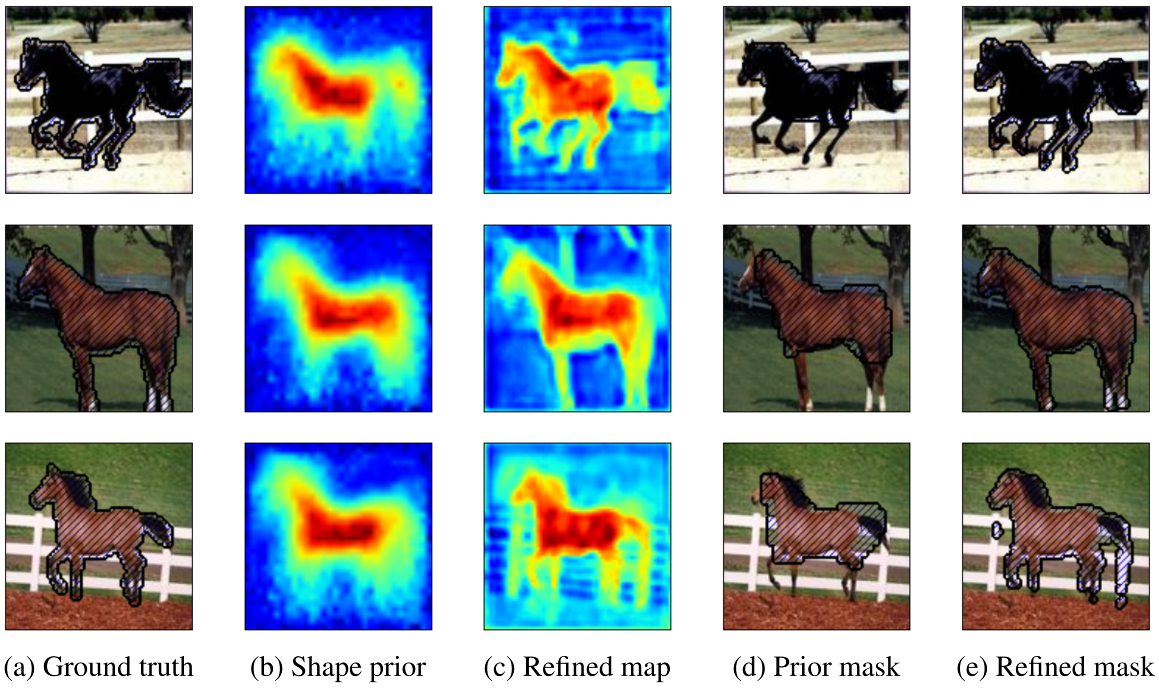


Figure 3.4. Example results from the convolutional refinement on Weizmann Horses.

multiple networks differing only in minor details or incrementally added layers, we use the templating framework Pyratemp [37] along with a different combination of parameter files for each step and dataset.

Features. For feature extraction, we used a network architecture based on [33]. Instead of retraining it, we used a pre-trained version available as a part of the Caffe framework [36]. We did not update these weights during the training process, except for the biases for the first convolutional layer, to account for the possibly different averages between ImageNet and our datasets.

The five convolutional layers of [33] give us one more choice: how many layers should we keep, what outputs should we use? We must make a tradeoff between localization accuracy and the semantic accuracy of features. For training our first, shape prior layer, we tried multiple different choices, but we found that given a sufficient amount of training iterations, the choice for the number of pre-trained image processing layers does not make a significant difference. In the end, we used the locally normalized output $\mathbf{f}^{(2)}$ of the second convolutional layer as the input for our shape prior. The choice is more clear for the second, convolutional stage: since we would like to have the highest resolution and spatial accuracy possible, we are using the normalized first convolutional output $\mathbf{f}^{(1HR)}$.

Training process. Although training the two stages of the synthesis network jointly also works reasonably well, we found that the training process is more stable if the loss of the first, global layer gets significantly more weight than that of the second layer. As an extreme, we settled on layerwise training, resulting in a performance gain on the order of a few percents.

Further gains could be achieved by considering the different purposes of the first and the second layer. For the global shape prior, in the case of uncertainty, we benefit from accurate probability estimates, thus we used the logistic loss

$$L_{\text{global}}(\mathbf{y}^{(1)}, \mathbf{y}^{(\text{gt})}) = \sum_i - \log(\mathbf{1}_{y_i^{(1)}=y_i^{(\text{gt})}} \sigma(y_i^{(1)}) + \mathbf{1}_{y_i^{(1)} \neq y_i^{(\text{gt})}} (1 - \sigma(y_i^{(1)}))) \tag{3.1}$$

where $y_i^{(\text{gt})} \in \{-1, 1\}$ are the ground truth mask pixels.

In contrast, the loss for the final output $\mathbf{y}^{(2)}$ approximates the pixelwise accuracy by

$$L_{\text{local}}(\mathbf{y}^{(2)}, \mathbf{y}^{(\text{gt})}) = ||\sigma(\mathbf{y}^{(2)}) - \mathbf{y}'^{(\text{gt})}||_2 \tag{3.2}$$

where $\mathbf{y}'^{\text{gt}} = \frac{\mathbf{y}^{\text{gt}}+1}{2}$ is the ground truth scaled to have a range from 0 to 1.

3.4.2 Datasets

To validate our model, we conducted experiments on the same datasets as in Chapter 2: we are using the Weizmann Horse [23], Penn-Fudan Pedestrian [30] and Caltech-UCSD Birds 200 [25] datasets, the latter with the refined foreground-background masks. We also did not change the training-test splits or the testing methodology, for a fair comparison.

3.4.3 Results

Based on the experiments, our architecture excels in situations in which there is a general shape prior but it needs to be heavily customized for the individual cases. For example, in the case of the

Table 3.1. Convolutional refinement: results on the Penn-Fudan Pedestrian dataset.

	AP	IoU
CRF	84.87	68.35
CHOPPs [1]	86.55	71.33
MMBM (1 layer)	89.91	76.92
MMBM (2 layers)	89.74	77.30
MMBM (1 layer, with GC	90.42	77.97
MMBM (2 layers), with GC	**90.77**	**79.42**
Ours, global only	84.08	63.79
Ours, refined	90.55	77.83

Table 3.2. Convolutional refinement: results on the Weizmann Horse dataset.

	AP	IoU
CRF	87.46	67.44
CHOPPs [1]	88.67	71.60
MMBM (1 layer)	89.43	69.59
MMBM (2 layers)	89.80	72.09
MMBM (1 layer), with GC	90.62	74.12
MMBM (2 layers), with GC	90.71	75.78
Ours, global only	83.88	49.96
Ours, refined	**95.16**	**83.55**

Weizmann Horse dataset, the general body plan of a horse is not highly variable but the different positions of legs are only weakly correlated at the pixel level, thus all of their combinations need to be learned separately by a global model. This might be a reason why the two-layer, ShapeBM version of MMBM has the biggest performance advantage over the single-layer version (74.12% for the ShapeBM, 69.59% for the global prior version, with the raw, pre-graph-cut results in Table 3.2). By using patchwise refinement, our model is able to take this even further, resulting in a significantly increased accuracy on this dataset.

For Caltech-UCSD Birds 200, the situation is less clear-cut: although it is this method that performs best on the overlap (IoU) metric, it only does so with a smaller margin (0.48%), and the 1-layer MMBM model from Chapter 2 still outperforms it in Average Precision. Nevertheless, these results also prove that the convolutional refinement is a viable substitute for CRF-based corrections when it comes to adapting global priors to local image details.

Finally, MMBM performs better on the Penn-Fudan Pedestrian dataset (with the difference being more significant on the IoU metric). The most likely explanation for this is the extensive clutter and occlusions present in these images, diminishing the advantage to be gained by local

Table 3.3. Convolutional refinement: results on the Caltech-UCSD Birds 200 dataset.

	AP	IoU
CRF	83.50	38.45
CHOPPs [1]	74.52	48.84
MMBM (1 layer)	88.07	72.96
MMBM (2 layers)	86.38	69.87
MMBM (1 layer), with GC	90.42	75.92
MMBM (2 layers), with GC	**90.77**	72.40
Ours, global only	84.02	69.40
Ours, refined	88.27	**76.30**

refinement. This renders the global shape model relatively more important, at which the more sophisticated training process of MMBM is able to perform well.

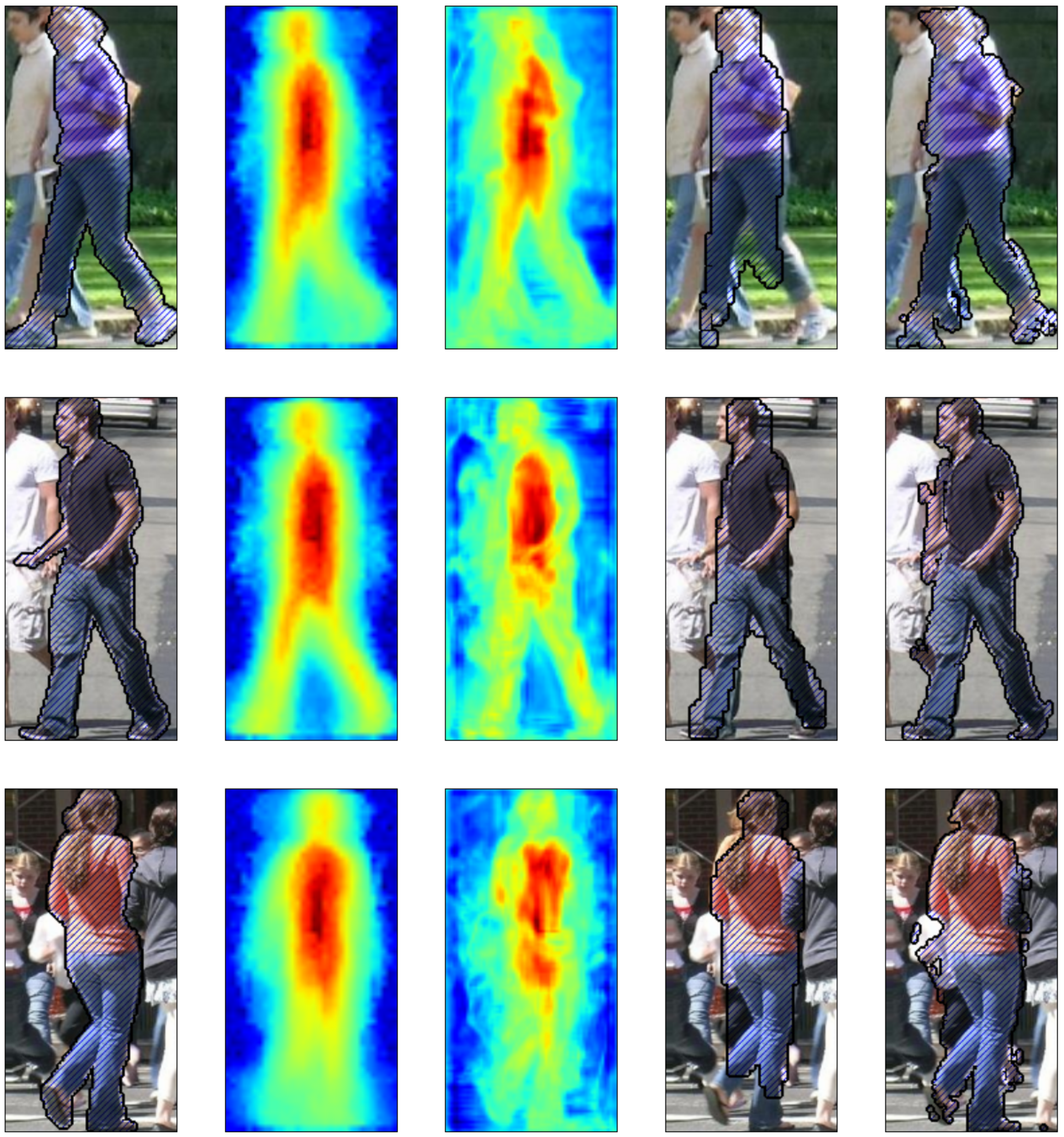

(a) Ground truth (b) Shape prior (c) Refined map (d) Prior mask (e) Refined mask

Figure 3.5. Example results from the convolutional refinement on the Penn-Fudan Pedestrian dataset.

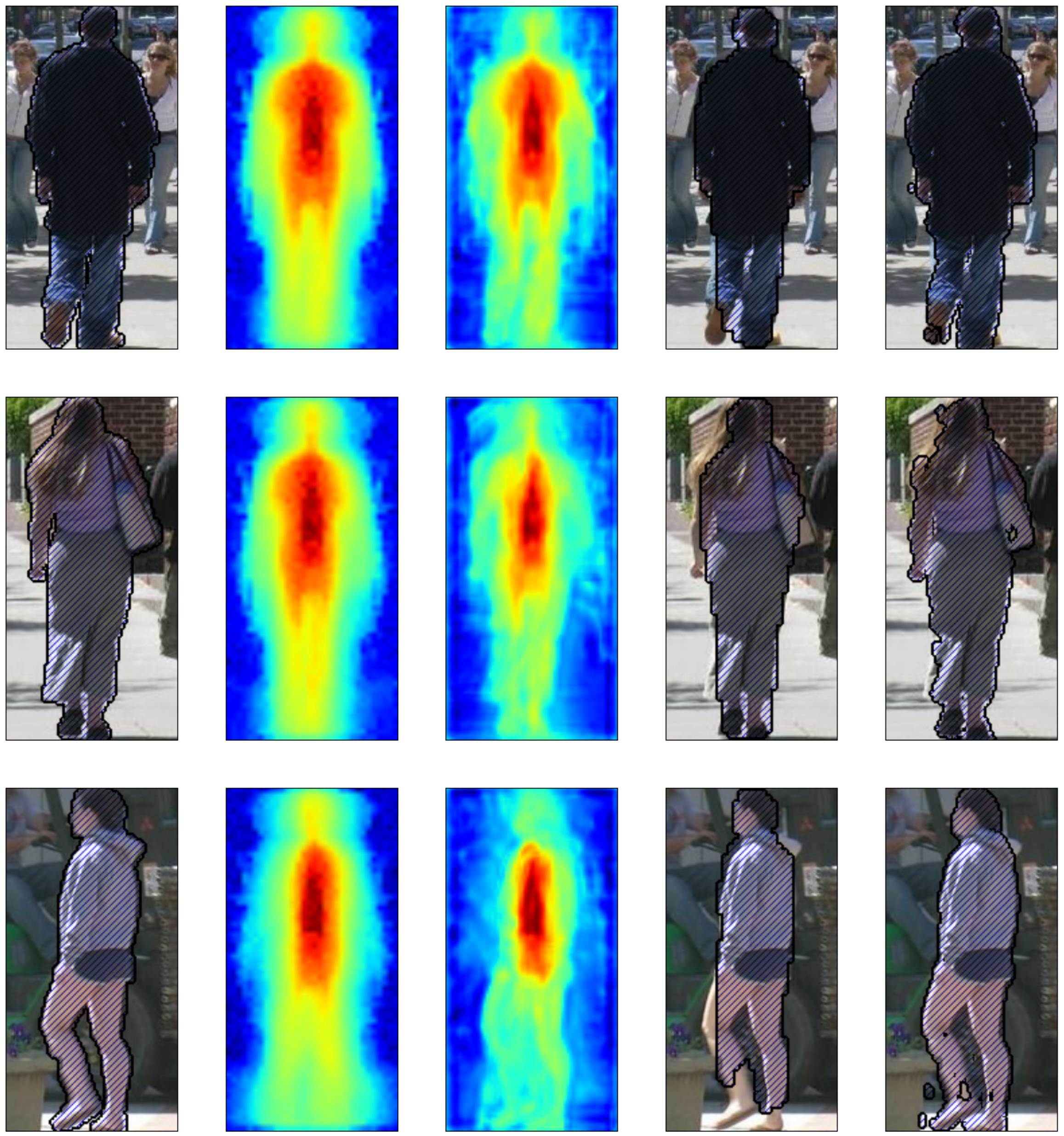


Figure 3.6. More results from the convolutional refinement on the Penn-Fudan Pedestrian dataset.

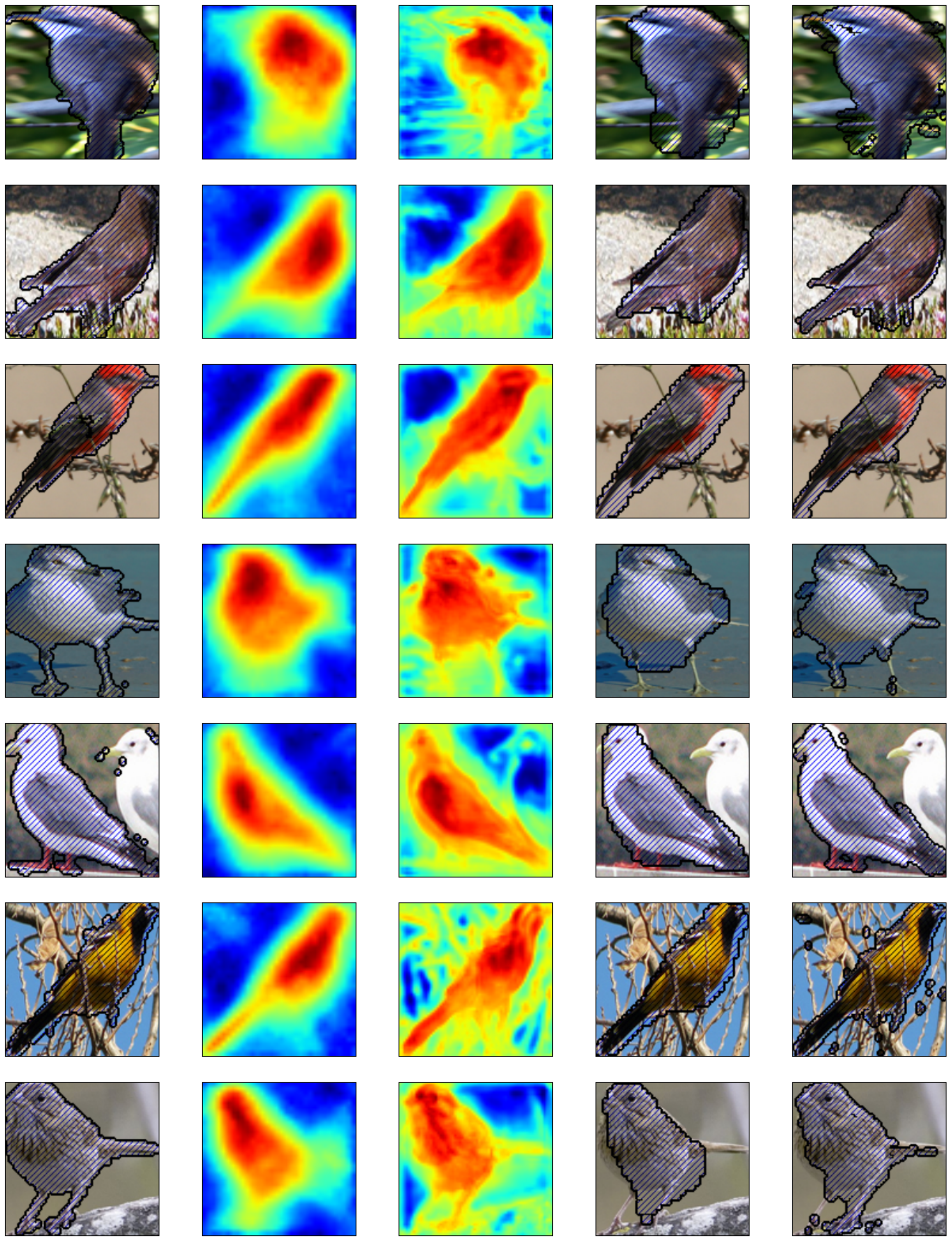


Figure 3.7. More results from the convolutional refinement on the Caltech-UCSD Birds 200 dataset.

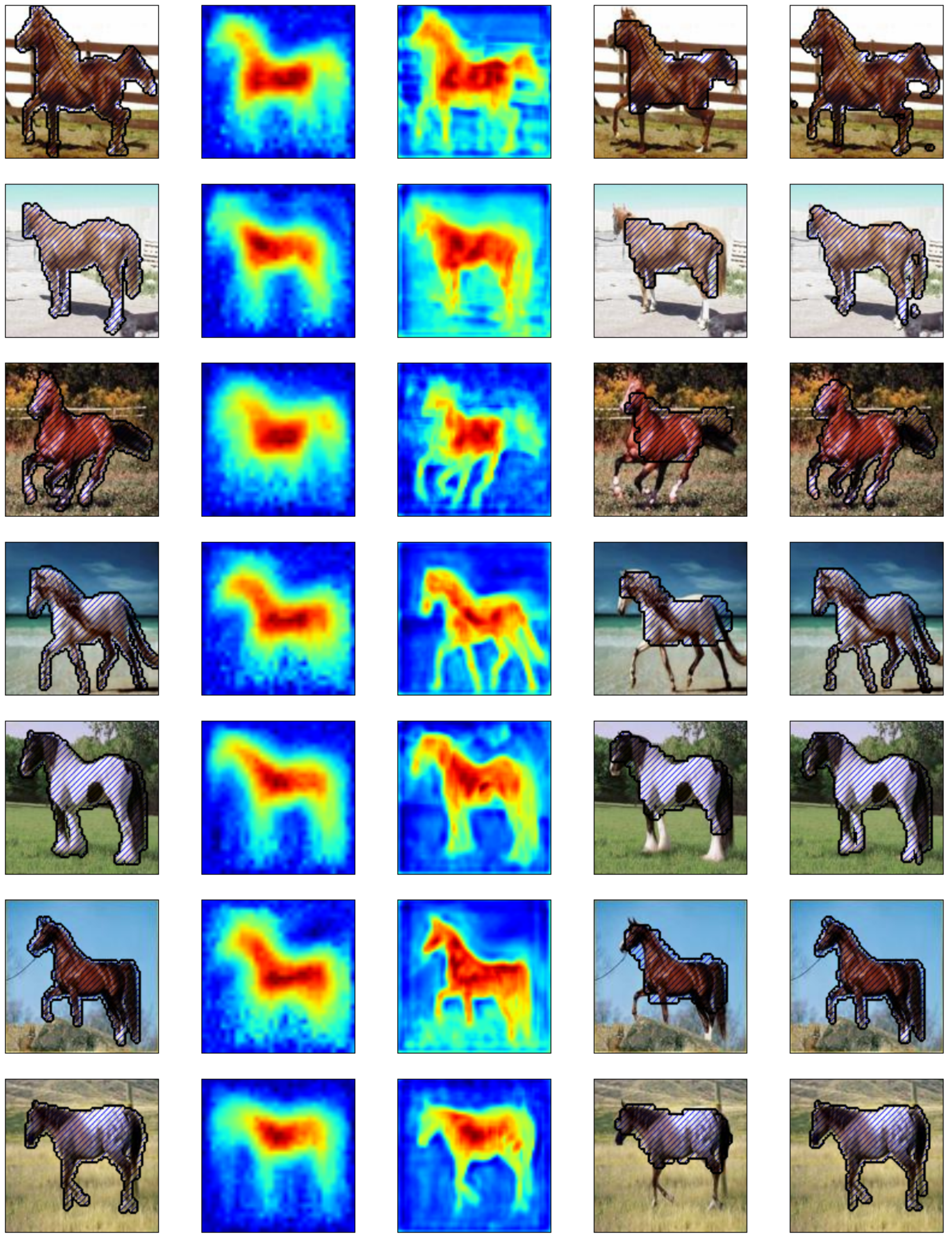


Figure 3.8. More results from the convolutional refinement on the Weizmann Horse dataset.

Chapter 4

Adaptive Structured Graph Cuts

4.1 Introduction

Formulating an approach purely as a deep learning architecture has numerous benefits, ranging from the availability of simple, gradient-based learning, easy inference at test time and the speed advantages provided by the several deep learning frameworks, some of which also exploit the advantages of modern, multicore CPUs and GPUs. Nevertheless, in order to obtain the best results, in certain cases the application of other, more complex inference methods could be also fruitful. It is therefore an interesting problem to investigate how to pose already known methods as deep learning, or, if this is not possible, use them as components of a network-based system.

Supervised, feedforward deep learning architectures do pay a price for having a simple sort of inference: nodes in a specific layer are conditionally independent given the previous layer and thus they cannot represent in-layer co-dependence relations easily. Local response normalization, like in [33], does alleviate this problem, but more advanced inference methods, such as Graph Cut, still present an enticing alternative when it comes to generating correct output segmentation masks.

Of course, it is relatively easy to use e.g. Conditional Random Fields (CRFs) as a refinement step only, as was indeed done in Chapter 2. In such a scenario, first a prior is obtained from a segmentation system, which is combined with pairwise weights obtained from an edge map. The balance between the two is usually determined by a single constant set by hand by a combination of cross-validation experiments, heuristics and best practice. However, this makes it hard to expand the model to, for example, use oriented edge maps, due to the larger number of parameters involved.

However, learning these parameters instead poses several challenges. Firstly, since the output of the inference process is not differentiable in terms of these, we cannot use simple backpropagation to obtain their values. Also, some possible parameter values might result in invalid inputs to the inference step, such as negative edge weights for Graph Cut.

This chapter presents an approach addressing these concerns. We still pose the problem of object segmentation in a deep learning framework, also proven by the fact that our implementation is done entirely under one of the many excellent deep learning libraries, Caffe [36]. We are focusing on CRFs and Graph Cut (as an inference method) as one of the modules in this system, and show how to learn the connection from preliminary segmentation masks and edge maps to unary and binary weights using a structured learning approach. We also introduce the problem of possibly negative edge weights and describe an approach to ameliorate this problem.

The architecture described in this chapter can be added to any Caffe-based segmentation algorithm easily. This includes the one described in Chapter 3, however, we found that the patch-based refinement process didn't leave much to correct for the Graph Cut inference. Therefore, we demonstrate our algorithm on an artificial dataset constructed by using coarse, low-resolution ground truth segmentation maps on the Caltech-UCSD Birds 200 dataset instead.

4.2 Related work

Learning CRF parameters by max-margin methods is not new, see for example [7] for a solution of a similar problem. However, their approach requires a separate learning framework (they are using svmStruct [38] for this purpose), and is thus it is not straightforward to add it to existing deep learning models. In contrast, our approach produces signals that can be backpropagated in an SGD process and thus can even be used to train multi-level networks supplying inputs to the CRF inference stage.

Another difference from these approaches is the use of anisotropic edge weights. An example

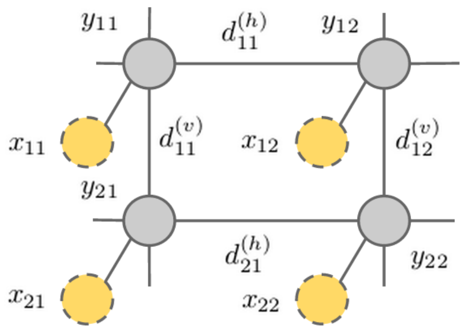


Figure 4.1. CRF nodes with our notation: x for the unary potentials, d for the pairwise ones, y are the foreground-background labels.

for such an approach is by Chen *et al.* in [39]; they are assigning different energies to connecting edges in a 8-neighborhood depending on the direction of the dominant local image edge. Nevertheless, instead of learning these weights, they are using fixed values (using the sine of the angle between the pixel distance vector and the image edge direction).

4.3 CRFs

The CRF formulation we are using is similar to that in most of the literature (e.g. [6]). We are using edges in a 4-neighborhood, with two labels for the foreground and background. Figure 4.1 introduces the notation that will be used in this chapter.

Usually, d_{ij} is chosen as a function of the local edge strength between pixels i and j, with the ratio between unary and binary edges determined by cross-validation. Instead, extending the approach presented by [39], we are using an anisotropic CRF. That is, instead of having horizontal and vertical edges with the same weights connecting them to the input edge map, our model has separate parameters to be learned for these. To give weights to these edges, we are using the edge strength map produced by the structured forests based edge detector by Dollár *et al.* [35] and processed by [2] to generate oriented edges in 8 different directions, presenting 8 feature maps, essentially. Thus, connecting the edge maps to the edge types, instead of a single "unary to binary edges ratio", we have 8×2 weights. This transformation is represented by a 1×1 convolution layer. In addition, we also performed experiments on the Ultrametric Contour Map (UCM) generated by [2] along with the raw map from [35].

Note that in theory, it would have been possible to connect the directional edge maps one by one to the different edge directions, resulting only in a smaller number of parameters to be learned. However, our architecture has the benefit of being able to learn negative correlations between edge directions (e.g. do not treat a horizontal edge as such if there is a much larger vertical edge around), with only a small increase in the amount of parameters (compared to the many weights e.g. fully connected layers have).

Conventionally, it is the CRF model itself that includes the unary and binary weights to be learned. In our case, we delegate the task of storing these weights to the preceding parts of the network and can thus assume that these are already of the right magnitude. This simplifies the implementation as weights don't need to be included, also enabling the use of more complex network structure to produce edge or unary maps and making it possible to apply backpropagation to train the entire structure.

As shown on Figure 4.2, our unary prior is scaled to obtain $\mathbf{x}$, the direct input to our graph cut layers (implemented, for example, by a 1×1 convolutional layer). In addition, we also load the edge map $\mathbf{e}$, and obtain the pre-scaled, per-edge-direction energies $\mathbf{d} = \{d_{i,j}\}$ for all the pairs $(i, j) \in \{C_{\text{vert}}, C_{\text{horiz}}\} = \mathcal{N}(\mathbf{I})$, with the C sets denoting vertical and horizontal neighboring pairs of pixels.

Thus, the energy function associated with the CRF becomes

$$E(\mathbf{y}, \mathbf{x}, \mathbf{d}) = E_{\text{unary}}(\mathbf{x}, \mathbf{y}) + E_{\text{binary}}(\mathbf{d}, \mathbf{y}) \tag{4.1}$$

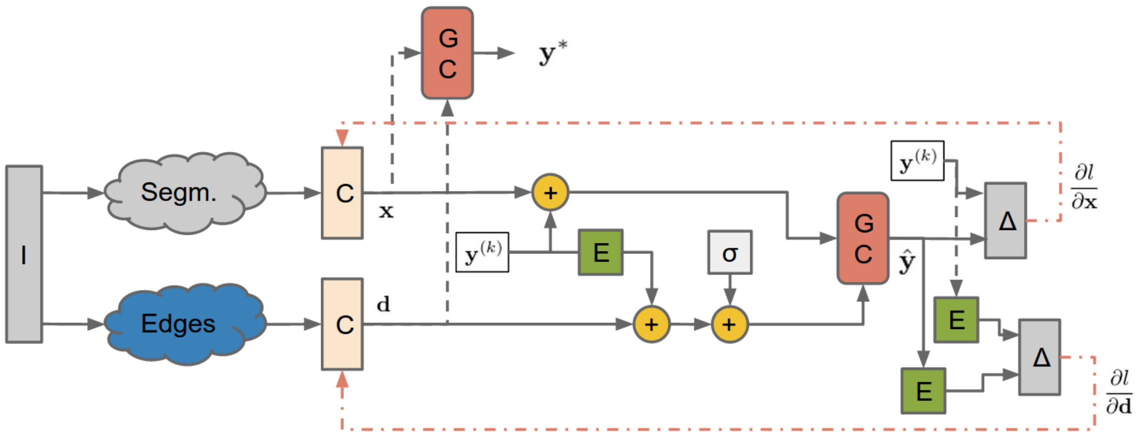
where

$$E_{\text{unary}}(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathbf{I}} x_i y_i \tag{4.2}$$

and

$$E_{\text{binary}}(\mathbf{d}, \mathbf{y}) = \sum_{(i,j) \in \mathcal{N}(\mathbf{I})} d_{ij} y_i y_j \tag{4.3}$$

with the assumption that $y_i \in \{-1, 1\}$ for background and foreground, respectively.



I
Segm.
Edges
C
C
x
d
GC
y^*
$y^{(k)}$
E
+
σ
GC
$\hat{y}$
Δ
$\frac{\partial l}{\partial x}$
$\frac{\partial l}{\partial d}$

Figure 4.2. An architecture overview. "C" denotes convolutional layers, "GC" is Graph Cut-based inference, "σ" is Gaussian noise and "E" represents a "mask to edges" transformation.

The optimization process is integrated into the deep learning framework as a "Graph Cut layer", having $\mathbf{d}$ and $\mathbf{x}$ as its inputs and supplying the optimization result

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} E(\mathbf{x}, \mathbf{d}, \mathbf{y}). \tag{4.4}$$

4.4 Learning

As shown above, the output of the network is obtained by an optimization process that is not only a black box from the point of view of gradient computation but it is also non-differentiable. In order to alleviate this difficulty, we resort to structured learning methods in obtaining gradients. These are then back-propagated throughout the rest of the network, just like with a conventional loss function.

For a k-th image, in the feedforward pass we obtain $\mathbf{x}^{(k)}$ and $\mathbf{d}^{(k)}$: the unary and binary pre-scaled coefficients of the CRF energy function, along with the ground truth $\mathbf{y}^{(k)}$. Ideally, we would prefer optimizing the Hamming distance

$$\Delta(\mathbf{y}^*, \mathbf{y}^{(k)}) = H(\mathbf{y}^*, \mathbf{y}^{(k)}) = -\sum_i y_i^* y_i^{(k)} + C \tag{4.5}$$

between the optimization result y^* and the ground truth (with C being a constant). Unfortunately, this is non-continuous in terms of the weights of the preceding network. To obtain a differentiable objective, we are replacing it with the structured objective

$$\begin{aligned} &l_{\text{max-margin}}(\mathbf{x}^{(k)}, \mathbf{d}^{(k)}, \mathbf{y}^{(k)}) = \\ &\max_{\hat{\mathbf{y}}} \Delta(\mathbf{y}^{(k)}, \hat{\mathbf{y}}) + E(\mathbf{x}^{(k)}, \mathbf{d}^{(k)}, \mathbf{y}^{(k)}) - E(\mathbf{x}^{(k)}, \mathbf{d}^{(k)}, \hat{\mathbf{y}}). \end{aligned} \tag{4.6}$$

Being the maximum of linear functions the energy consists of, the loss has a subgradient in both $\mathbf{x}$ and $\mathbf{d}$. Using the fact that the first energy term $E(\mathbf{x}^{(k)}, \mathbf{d}^{(k)}, \mathbf{y}^{(k)})$ does not depend on $\hat{\mathbf{y}}$, we can maximize the other two terms separately, which, after some swapping of signs, becomes

$$\begin{aligned} l_{\text{max-margin}}(\mathbf{x}^{(k)}, \mathbf{d}^{(k)}, \mathbf{y}^{(k)}) = E(\mathbf{x}^{(k)}, \mathbf{d}^{(k)}, \mathbf{y}^{(k)}) - \\ \min_{\hat{\mathbf{y}}} E(\mathbf{x}^{(k)}, \mathbf{d}^{(k)}, \hat{\mathbf{y}}) - \Delta(\mathbf{y}^{(k)}, \hat{\mathbf{y}}). \end{aligned} \tag{4.7}$$

The objective function for $\hat{\mathbf{y}}$ is very similar to the one we performed to obtain $\mathbf{y}^*$ itself. The only difference is a margin that, being a Hamming distance, rewards pixels of the solution $\hat{\mathbf{y}}$ being different from the ground truth, $\mathbf{y}^{(k)}$. Knowing the latter, this can be easily represented by an additional unary term for the Graph Cut. That is,

$$\begin{aligned} \hat{\mathbf{y}} = \arg\min_{\hat{\mathbf{y}}} \sum_{i \in \mathbf{I}} x_i \hat{y}_i + \sum_{(i,j) \in \mathcal{N}(\mathbf{I})} d_{ij} \hat{y}_i \hat{y}_j - (-\sum_{i \in \mathbf{I}} \hat{y}_i y_i^{(k)}) = \\ \arg\min_{\hat{\mathbf{y}}} \sum_{i \in \mathbf{I}} (x_i + y_i^{(k)}) \hat{y}_i + \sum_{(i,j) \in \mathcal{N}(\mathbf{I})} d_{ij} \hat{y}_i \hat{y}_j, \end{aligned} \tag{4.8}$$

that is, positively bias (and lower the energy of) all pixels in $\mathbf{y}$ that used to be -1 and give a negative bias for those that had a value of 1.

Note that in this formulation, edge weights d_{ij} and unary potentials x_i have the opposite signs than it might be expected. That is, a positive x_i encourages a pixel to be background (since, to minimize $x_i y_i$, $y_i = -1$ is the optimal choice. Similarly, it is a negative $d_i j$ is the one forcing two neighboring pixels to be of the same sign (label) by forcing $y_i y_j$ to be positive.

Computing the gradient of (4.6), we obtain

$$\frac{\partial l}{\partial \mathbf{x}} = \mathbf{y}^{(k)} - \hat{\mathbf{y}} \tag{4.9}$$

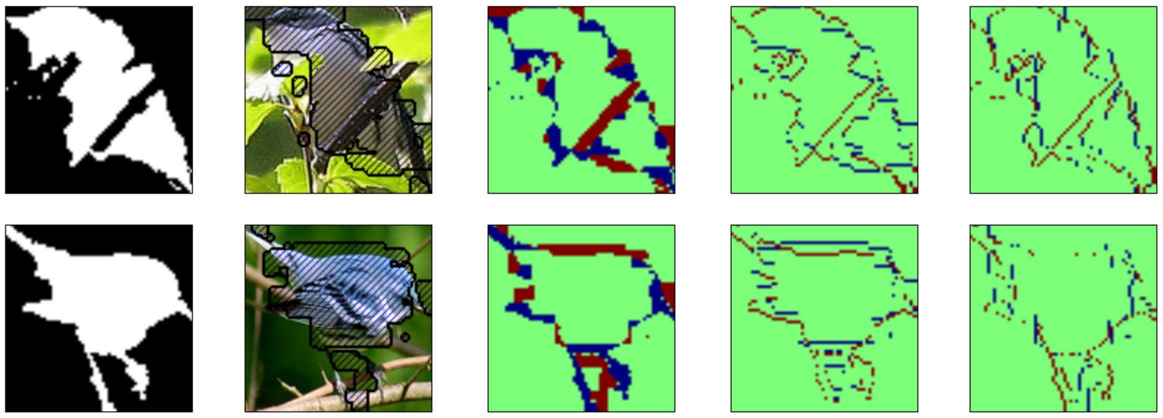

(a) Ground truth (b) Margin mask (c) Unary gradient (d) Horizontal pairwise gradient (e) Vertical pairwise gradient

Figure 4.3. The learning process. We using the margin-enhanced output $\hat{\mathbf{y}}$ on (b) against the ground truth mask and edges, resulting in the gradients on (c), (d) and (e). We are getting different training signals for horizontal and vertical edges.

and

$$\frac{\partial l}{\partial d_{ij}} = y_i^{(k)} y_j^{(k)} - \hat{y}_i \hat{y}_j. \tag{4.10}$$

Thus, essentially, we are backpropagating the difference of the ground truth and $\hat{y}$ towards the unary inputs. As for the gradient in (4.10), it is a "differential edge map" between the ground truth and $\hat{y}$. Latter is implemented by the "mask to edges" transformations on Figure 4.2 to fit it into the network.

4.4.1 About the edge margin

So far we have put a margin on the mask output but not on the resulting edges. However, enforcing a margin on the edge maps turns out to be even more crucial than of that on mask pixels.

To see the reason for this, consider an iteration early in the training process. Since we haven't learned the connection between the edge maps and the pairwise potentials yet, weights are close to zero, edge strengths are very low, thus the CRF just performs pixel-wise thresholding over the unary potentials. This is the solution to be improved upon by forcing the edges to follow those on the edge maps.

On average, ground truth edges are going to have higher edge strength values than the predicted edges. However, what about the general probability of edges?

CRFs are generally used for smoothing of results, using edges as an additional source of information. However, due to the nature of our algorithm, the prior probabilities (and therefore the mask obtained by thresholding) are already mostly smooth, the challenge lies in adjusting the edges to closely follow the edges instead. Thus, the ground truth mask $\mathbf{y}^{(k)}$ is likely to have more edge pixels than $\hat{\mathbf{y}}$, resulting in a negative average for (4.10), eventually driving earlier layers towards producing negative edge weights.

Given an inference algorithm without any constraints, this wouldn't pose any problem: with negative weights, the optimal $\hat{\mathbf{y}}$ would contain a large number of neighbor pairs with different labels (possibly laid out in a "chessboard" pattern near regions with indecisive unary priors), thus turning the above balance and eventually restoring positive edge weights. However, due to the nature of the Graph Cut algorithm only being capable of dealing with nonnegative edge weights, negative values get clamped to zero, again producing smooth edge maps, reinforcing the choice of the negative weights. This results in the network never learning the proper prior probability of the edges.

Looking closely, the nature of this problem is twofold. Firstly, the edge map from $\hat{\mathbf{y}}$ is not forced to have a large margin compared to $\mathbf{y}^{(k)}$ and thus is not guiding the learning process of the edge maps well. This is readily ameliorated by the addition of a second margin term

$$\Delta_{\text{edges}}(\hat{\mathbf{y}}, \mathbf{y}^{(k)}) = -\sum_{(i,j)\in\mathcal{N}(\mathbf{I})} \hat{y}_i y_j^{(k)}. \tag{4.11}$$

This will result in (4.8) becoming

$$\hat{\mathbf{y}} = \arg\min_{\hat{\mathbf{y}}} \sum_{i\in\mathbf{I}} (x_i + y_i^{(k)})\hat{y}_i + \sum_{(i,j)\in\mathcal{N}(\mathbf{I})} (d_{ij} + y_i^{(k)} y_j^{(k)})\hat{y}_i\hat{y}_j. \tag{4.12}$$

4.4.2 Solving the edge margin problem

However, using an edge margin still does not solve the second problem, namely, that the above expression involves optimization with negative edge weights.

We might simply replace them by $\max(d_{ij}, 0)$, which is indeed what our solver [40] is doing; since the optimal energies for our model are very likely to be positive, this will not be a problem once our gradient descent approaches this eventual solution. Nevertheless, we still need an approximate solution for $\hat{\mathbf{y}}$ in the initial stage of the descent.

Intuitively, the exact solution for $\hat{\mathbf{y}}$ would look like a chessboard pattern: in (4.12), the edge weights $d_{ij} + y_i^{(k)} y_j^{(k)}$ become dominated by the second, edge margin term since d_{ij} is still not the negative value of sufficient magnitude it will be eventually, and thus minimization is accomplished by choosing many negative $\hat{y_i}\hat{y_j}$ values, that is, as many edges as possible. However, note that in order to drive the learning process to the direction of assigning the proper weights to edges, we do not need the exact solution $\hat{\mathbf{y}}$, that is, the solution that balances between the right unaries and as many edges as possible; it is sufficient if we approximate this (intractable) solution.

To solve this problem, consider the purpose of having a margin: "find a solution that is far from the desired one but still has a good score". Let us suppose that we have an exact solution for (4.12), denoted by $\hat{\mathbf{y}_e}$. By adding an additional term $\Delta(\hat{\mathbf{y}}, \hat{\mathbf{y}_e}) = H(\hat{\mathbf{y}}, \hat{\mathbf{y}_e})$ to the minimization, we obtain

$$\hat{\mathbf{y}} = \arg\min_{\hat{\mathbf{y}}} \sum_{i \in \mathbf{I}} (x_i + y_i^{(k)} - \hat{y}_{e,i})\hat{y}_i + \sum_{(i,j) \in \mathcal{N}(\mathbf{I})} (d_{ij} + y_i^{(k)} y_j^{(k)})\hat{y}_i\hat{y}_j. \tag{4.13}$$

The solution does not change, since the additional term, being a difference between the solution and the optimal solution, is zero for the minimum.

In fact, even if our solution is only approximate (by replacing negative edge weights with zeroes), the knowledge of the optimal solution $\hat{y_e}$ might help obtaining a close to optimal $\hat{\mathbf{y}}$. In a way, this formulation is the opposite of having a margin: "find a solution that is close to the one that we would find if we solved the problem exactly".

Of course, if we know the optimal solution, there would be no point in solving it again. Thus we replace $\hat{\mathbf{y}}_e$ with an approximation $\hat{\mathbf{y}}_g$, one that is still closer to the chessboard-like pattern described above than the one one obtainable by Graph Cut and clamping edge weights at zero.

Since we are not aiming at the optimum, only a solution that is reasonably different from $\mathbf{y}^*$ in terms of edges, even a low-quality approximation will do well. For this purpose, we are simply using a Gaussian

$$\mathbf{y}_g \sim \mathcal{N}(\mathbf{y}_g, 0, \sigma) \tag{4.14}$$

with $\sigma = 1$. Although this "solution" is obviously not a close approximation of $\hat{\mathbf{y}}$, for negative edge weights it is still better than

$$\hat{\mathbf{y}}_{\text{clamp}} = \arg\min_{\hat{\mathbf{y}}} \sum_{i \in \mathbf{I}} (x_i + y_i^{(k)})\hat{y}_i + \sum_{(i,j) \in \mathcal{N}(\mathbf{I})} \min(0, d_{ij} + y_i^{(k)} y_j^{(k)})\hat{y}_i \hat{y}_j. \tag{4.15}$$

as even random noise is more likely to have more edges than the mostly flat output provided by (4.15). Therefore it is capable of guiding the gradient descent towards positive edge weights.

It is also easy to implement: we only need to add random Gaussian noise to the unary term of (4.13).

4.5 Experiments

4.5.1 Implementation details

Edge maps. We are using the Structured Forest based edge detector from [35], running it on the original, high-resolution image (e.g. 256×256 in case of CUB), and then applying [2] to obtain oriented edge maps with 8 directions. These are downscaled to fit the output mask size by the application of max-pooling.

Convolution for the edges. In practice, the edge energies $\mathbf{d}$ are stored in the form of feature maps $\mathbf{d}^{(h)}$ and $\mathbf{d}^{(v)}$, each having the same resolution as the image itself (see Figure 4.1). That is, edge magnitudes are also stored as raster images even though edges are not a property of the pixels themselves but of their connections. This raises the issue of alignment, namely, given two neighboring pixels i and j, when looking for the edge weight for them, should we use pixel i or pixel j on the edge map?

Instead of trying to find a hand-crafted solution, we opt for learning the optimal connection. Instead of using a 1×1 convolution, we also perform experiments using a 3×3 one, letting the learning process learn the preferred offsets and combinations of edge map pixels to be used. We measured a minor performance increase (on the order of 0.5% both in AP and IoU) when using the setup with the increased receptive field. We present both kinds of results later in the chapter.

Graph Cut layers. Graph Cut optimization is added to Caffe as a special layer, without any parameters stored by the layer itself, running the algorithm described in [40] and implemented by Vladimir Kolmogorov. Solving a single optimization problem takes 5ms on our machine (Intel Core i7 @ 3.4 GHz), which is similar to the per-training-image time taken by the algorithm presented in Chapter 3 (latter being run on a high-end GPU). This also demonstrates that it is feasible to use Graph Cuts not only for testing but also as part of the training process.

4.5.2 Results

We experimented with using the Graph Cut stage as an addition to the network described in Chapter 3 first, however, as we already mentioned, we found that the CRF and convolutional post-processing stages perform similar functions, thus adding both does not improve results. Also, a very good quality input prior encourages the system to use weak edges for the CRF model, since the possible corrections would be unlikely to help much.

Instead, we demonstrate the effectiveness of the learning process on an artificially generated dataset. Starting from the Caltech-UCSD Birds 200 ground truth, we applied an average pooling operation followed by thresholding, so that the effective resolution of the masks was reduced to 8×8 pixels (as opposed to the 64×64 output resolution we were using throughout the experiments). Given these, we tasked the inference process with the restoration of the original masks.

Figure 4.4 shows some results during the learning and evaluation process. For this figure, we used the UCM from [2]. The resulting output masks in column (d) do approximate object boundaries well. Artifacts from the original, low-resolution shape prior are still observable but mostly only at locations where the edge information was not conclusive enough.

The margin-augmented optimization result $\hat{\mathbf{y}}$ is also shown in column (e). Most of the differences from $\mathbf{y}^*$ are observable near the edges; for inner regions, the edge weights force the result to be consistent even in the presence of the margin augmentation. Also note that there is only little observable effect of the noise term described in Section 4.4.2 (but they are still there, at, for exam-

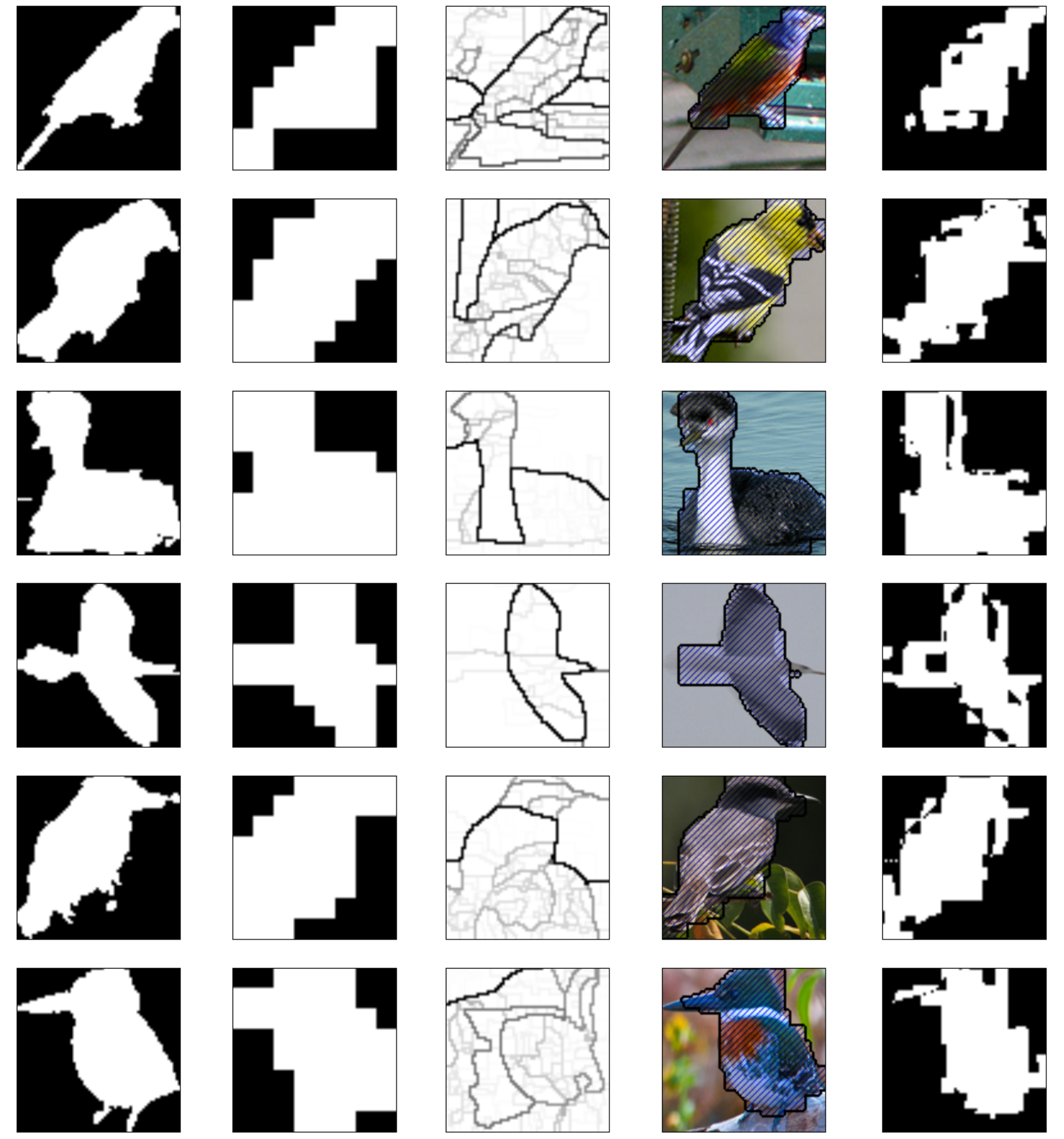


Figure 4.4. Example results from the output of the CRF inference, using the UCM from [2] as edge maps, with the application of a 3×3 convolution to obtain pairwise energies. We used a lower resolution version of the ground truth as the (unary) shape prior. (d) is the actual output of the system, while (e) shows the margin-enhanced mask, used for learning. (c) only shows the first, horizontal channel of the edge map, as it does not noticeably differ from the vertical one for UCM inputs.

Table 4.1. Comparison of design choices on an artificial dataset.

Edge map	Connecting layer	AP	IoU
Raw PD	single weight	91.08	81.35
Raw PD	3x3 convolution	91.34	81.77
Oriented	single weight	91.37	82.44
Oriented	3x3 convolution	91.79	82.65
UCM	single weight	91.61	82.27
UCM	3x3 convolution	**91.86**	**82.81**

ple, the single-pixel background hole at the legs of the bird in the second row). The reason for this is that the learned edge terms already compensate for its effects. It is easy to prove its importance though: for experiments with the noise term having been removed, no weight will be given to the edges eventually, resulting in a significantly worse result. However, the positive outcome is very robust regarding the exact magnitude of the noise term, described by the parameter σ: we got very similar eventual results when varying its value from $\sigma = 0.1$ to $\sigma = 5$.

For quantitative analysis, we compared the performance achievable by three different edge maps; we also varied the type of the convolutional layer connecting the edge map(s) to the pairwise edge strength inputs of the Graph Cut layers (the two options being a 1×1 and a 3×3 convolution). We got the best results for the single-channel UCM, followed by the 8-channel oriented edges and the raw results from [35].

It still remains an area for improvement to make the method more robust for real-world datasets, where the shape prior is already relatively accurate and thus provides a less reliable learning signal. However, the general principle of integrating deep learning and custom inference methods through structured learning is a promising area for future research.

Chapter 5

Conclusion

In this thesis, we present three different approaches to obtain and refine object segmentation masks using neural architectures. First, we apply CRFs in an unsupervised way, followed by joint training. Then, we describe an approach that makes use of supervised learning and patch-based mask refinement. Finally, a method to learn input weights for a graph cut based segmentation approach is discussed.

In light of our results on the three different datasets, we might draw several conclusions. First, we are reinforced in our suspicion that having a class-specific shape prior is a very important ingredient of a successful object segmentation algorithm. Using shape priors, we can differentiate between the object and cluttered background, multiple instances of the same object, and it might also fix errors due to discontinuities caused by occlusion (in fact, Figure 3.7 has examples for all of these). In this thesis, we discuss multiple methods to learn shape priors and show the tradeoff between the simplicity of the training process and the quality of the obtained global shape masks.

Second, we also demonstrate that global shape alone is not sufficient for good performance: it is best used as a prior to guide another step which takes into account local appearance. In addition to the application of Graph Cut improving results from Chapter 2, this is also shown by the large improvements in accuracy gained by the convolutional refinement method in Chapter 3. However, these methods help only if the global model provides a good starting point: a significant portion of the mistakes made by our models are due to the latter being incorrect, outlining a future research direction for their improvement.

Furthermore, we also visit the problem of learning parameters for Conditional Random Fields with 4-connected neighborhoods. Our approximate solution to the seemingly intractable problem of generating max-margin examples for training purposes serves as another example of how randomness can be successfully used where exact solutions do not prove to be appropriate.

We use three datasets to evaluate our segmentation algorithms, all with different kinds of challenges. The Weizmann Horse dataset generally presents more clear color cues for segmentation, with a smaller variety of overall shape, but with parts that are hard to model by global shape models, thus favoring our flexible, convolution-based approach. The Penn-Fudan Pedestrian dataset, on the other hand, is more demanding in terms of global shape modeling due to the extensive clutter being present and the unpredictable color of clothing in the images; for this dataset, it is the two-layer MMBM model that shows the best performance. Finally, Caltech-UCSD Birds 200 is challenging due to the large variability of both shape and color, with no clear winner out of the two models.

Of course, there are many questions still left open for further research. For example, how well can our segmentation methods be fit into a pipeline starting with detection and ending with recognition? How can these models be extended to not just foreground-background segmentation but also for parts labeling? Also, do we need all the training masks to learn shape models, as opposed to just bounding boxes? The methods described in this thesis can also serve as a basis of these directions, in addition to already giving multiple viable solutions to the object segmentation task itself.

References

[1] Y. Li, D. Tarlow, and R. Zemel, “Exploring compositional high order pattern potentials for structured output learning,” IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[2] Y.-T. Chen, J. Yang, and M.-H. Yang, “Extracting image regions by structured edge prediction,” IEEE Winter Conference on Applications of Computer Vision, 2015.

[3] http://xkcd.com/1425.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[5] J. Yang, S. Safar, and M.-H. Yang, “Max-margin boltzmann machines for object segmentation,” IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[6] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” International Conference on Machine Learning, 2001.

[7] M. Szummer, P. Kohli, and D. Hoiem, “Learning CRFs using graph cuts,” European Conference on Computer Vision, 2008.

[8] P. Kohli, L. Ladicky, and P. H. S. Torr, “Robust higher order potentials for enforcing label consistency,” IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[9] C. Rother, P. Kohli, W. Feng, and J. Jia, “Minimizing sparse higher order energy functions of discrete variables,” IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[10] A. Shekhovtsov, P. Kohli, and C. Rother, “Curvature prior for MRF-based segmentation and shape inpainting,” Annual Symposium of the German Association for Pattern Recognition (DAGM), 2012.

[11] M. P. Kumar, P. Torr, and A. Zisserman, “Obj cut,” IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[12] R. Salakhutdinov and G. Hinton, “Deep Boltzmann machines,” International Conference on Artificial Intelligence and Statistics, 2009.

[13] S. M. A. Eslami, N. Heess, and J. Winn, “The shape Boltzmann machine: a strong model of object shape,” IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[14] S. M. A. Eslami and C. K. I. Williams, “A generative model for parts-based object segmentation,” Neural Information Processing Systems, 2012.

[15] J. Besag, “On the statistical analysis of dirty pictures,” *Journal of the Royal Statistical Society: Series B*, vol. 48, no. 3, pp. 259–302, 1986.

[16] B. Taskar, C. Guestrin, and D. Koller, “Max-margin Markov networks,” Neural Information Processing Systems, 2003.

[17] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, “Large margin methods for structured and interdependent output variables,” *Journal of Machine Learning Research*, vol. 6, pp. 1453 – 1484, 2005.

[18] T. Joachims, T. Finley, and C.-N. Yu, “Cutting-plane training of structural svms,” *Machine Learning*, vol. 77, no. 1, pp. 27–59, 2009.

[19] L. Bertelli, T. Yu, D. Vu, and B. Gokturk, “Kernelized structural svm learning for supervised object segmentation,” IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[20] C.-N. J. Yu and T. Joachims, “Learning structural svms with latent variables,” International Conference on Machine Learning, 2009.

[21] X. Miao and R. P. N. Rao, “Large margin boltzmann machines,” IJCAI, 2009.

[22] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[23] E. Borenstein and S. Ullman, “Class-specific, top-down segmentation,” European Conference on Computer Vision, 2002.

[24] Y. Bo and C. Fowlkes, “Shape-based pedestrian parsing,” IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[25] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, “Caltech-UCSD Birds 200,” California Institute of Technology, Tech. Rep., 2010.

[26] A. Kae, K. Sohn, H. Lee, and E. Learned-Miller, “Augmenting CRFs with Boltzmann Machine shape priors for image labeling,” IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[27] F. Chen, H. Yu, R. Hu, and X. Zeng, “Deep learning shape priors for object segmentation,” IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[28] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural Computation*, vol. 14, no. 8, pp. 1771 – 1800, 2002.

[29] V. Mnih, H. Larochelle, and G. E. Hinton, “Conditional restricted Boltzmann machines for structured output prediction,” UAI, 2011.

[30] L. Wang, J. Shi, G. Song, and I.-F. Shen, “Object detection combining recognition and segmentation,” Asian Conference on Computer Vision, 2007.

[31] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[32] Y. Y. Boykov and M.-P. Jolly, “Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images,” International Conference on Computer Vision, 2001.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Neural Information Processing Systems, 2012.

[34] C. Dong, C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” European Conference on Computer Vision, 2014.

[35] P. Dollár and C. L. Zitnick, “Structured forests for fast edge detection,” International Conference on Computer Vision, 2013.

[36] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.

[37] R. Koebler, “pyratemp templating framework,” http://www.simple-is-better.org/template/pyratemp.html.

[38] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, “Support vector machine learning for interdependent and structured output spaces,” International Conference on Machine Learning, 2004.

[39] R. Chen, X. Li, and S. Li, “Image inpainting based on anisotropic MRF model,” Sixth International Symposium on Multispectral Image Processing and Pattern Recognition. International Society for Optics and Photonics, 2009.

[40] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.