UNIVERSITY OF CALIFORNIA
Los Angeles

Learning Representation for Scene Understanding: Epitomes, CRFs, and CNNs

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Liang-Chieh Chen

2015

ABSTRACT OF THE DISSERTATION

Learning Representation for Scene Understanding: Epitomes, CRFs, and CNNs

by

Liang-Chieh Chen

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2015

Professor Alan L. Yuille, Chair

Scene understanding, such as image classification and semantic image segmentation, has been a challenging problem in computer vision. The difficulties mainly come from the feature representation, *i.e.*, how to find a good representation for images. Instead of improving over hand-crafted features such as SIFT or HoG, we focus on learning image representations by generative and discriminative methods.

In this thesis, we explore three areas: (1) generative models, (2) graphical models, and (3) deep neural networks for learing image representations. In particular, we propose a dictionary of epitomes, a compact generative representation for explicitly modeling object co-relation within edge patches, and for explicitly modeling photometric and position variability of image patches. Subsequently, we exploit Conditional Random Fields (CRFs) to take into account the dependencies between outputs. Finally, we employ Deep Convolutional Neural Networks trained with large-scale datasets to learn feature representations. We further combine CRFs with deep networks to estimate complex representations. Specifically, We show that our proposed model can achieve state-of-art performance on challenging semantic image segmentation benchmarks.

The dissertation of Liang-Chieh Chen is approved.

Stefano Soatto

Demetri Terzopoulos

Ying Nian Wu

Alan L. Yuille, Committee Chair

University of California, Los Angeles

2015

To my family and my wife ...

for their constant support and unconditional love.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Thanks . . .

First and foremost, to my adviser, Alan L. Yuille, for encouraging my research and for his enlightening instructions. He has been a great mentor to me. I admire his passion about research: he is always keen to learn latest breakthroughs in computer vision. Most importantly, I have learned from him that only when you can express the idea clearly to others can you learn something by heart, and during the process you will always find a new way to further refine your ideas. I always remember the day that he, George and I were on the roof of Boelter Hall, reciprocating about the research ideas until night.

To my thesis committee, Stefano Soatto, Demetri Terzopoulos, and Ying Nian We, for their feedback and advice of this work.

To many great researchers whom I was lucky to collaborate with during the pursuit of Ph.D. degree. Thank George Papandreou, who has been helping me a lot (from the time when I started my first research project on computer vision and even now). I am very grateful for his generous support, and I always remember the night I slept over at his apartment in order to catch a deadline. Thank Raquel Urtasun, who has hosted me for two visits (one at Toyota Technological Institute at Chicago and the other at University of Toronto). I will never forget her energetic attitude toward research and she always works hard for the best. Thank Sanja Fidler, Alexander Schwing and Iasonas Kokkinos for helping me a lot on the projects. Without their help, I could not have finished them. Thank Kevin Murphy for having me as an intern at Google and for giving me helpful suggestions during the internship.

To many great friends I met during this journey. Thank all the group member (including visiting students) of Center for Cognition, Vision, and Learning (CCVL): Boyan Bonev, Yuanhao Chen, Xianjie Chen, Nam-gyu Cho, Xiaodi Hou, Xiaochen Lian, Xiaobai Liu, Junhua Mao, Roozbeh Mottaghi, Weichao Qiu, Zhou Ren, Chunyu Wang, Jianyu Wang, Peng Wang, Fangting Xia, Bo Xin, Xingyao Ye, Jun Zhu, and Yu Zhu, and thank all the member of Raquel's group (including visiting students): Chen Kong, Kaustav Kundu, Chenxi Liu, Wenjie Luo, Ye Meng, Edgar Simo-Serra, Shenlong Wang, Yali Wang, Jia Xu, Jian Yao, Ziyu Zhang, Yinan Zhao, and Yukun Zhu.

Thank them for discussing research with me, playing sports with me, and hanging out with me. Furthermore, thank Chien-Ju Ho, Ming-Chun Huang, Li-Yang Ku, Chi-Yu Li, Jui-Ting Weng, and Guan-Hua Tu for being wonderful friends at UCLA. They made this journey joyful.

To my family for their support. My hard-working parents have sacrificed their lives for providing me with unconditional love and care. I could not complete my degree without them. Thank my aunt for taking care of me, and thank my brother and sisters for looking after my dad and mom while I was not there. I am also grateful for my mother-in-law and father-in-law. Thank them for being considerate of that I could not visit them very often. At the end, I would like to express my appreciation to my beloved wife. Thank my wife for backing me up while I was struggling and depressed, and thank for being with me to share both the happy and sad moments. These past five years have not been an easy ride, both academically and personally.

VITA

2000–2004	B.S. (Electronics Engineering), National Chiao-Tung University, Taiwan.
2006–2008	M.S. (Electrical Engineering: Systems), University of Michigan- Ann Arbor, MI.
2013	Research Intern, Toyota Technological Institute at Chicago. Mentor: Dr. Raquel Urtasun.
2014	Visiting Student, University of Toronto. Mentor: Dr. Raquel Urtasun.
2015	Software Engineering Intern, Google Inc., Mountain View, CA. Mentor: Kevin Murphy.
2015	Software Engineering Intern, Baidu USA, Sunnyvale, CA. Mentor: Yang Yi.
2011–present	Research Assistant, University of California- Los Angeles. Advisor: Dr. Alan L. Yuille.

PUBLICATIONS

• George Papandreou*, **Liang-Chieh Chen***, Kevin Murphy, and Alan L. Yuille. *Weakly- and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation*, International Conference on Computer Vision (ICCV), 2015. (*equal contribution)

• **Liang-Chieh Chen***, Alexander G. Schwing*, Alan L. Yuille, and Raquel Urtasun. *Learning Deep Structured Models*, Interational Conference on Machine Learning (ICML), 2015. (*equal contribution)

• **Liang-Chieh Chen***, George Papandreou*, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. *Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs*, International Conference on Learning Representations (ICLR), 2015. (*equal contribution)

• **Liang-Chieh Chen**, Sanja Fidler, Alan L. Yuille, and Raquel Urtasun. *Beat the MTurkers: Automatic Image Labeling from Weak 3D Supervision*, International Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

• George Papandreou, **Liang-Chieh Chen**, and Alan L. Yuille. *Modeling Image Patches with a Generic Dictionary of Mini-Epitomes*, International Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

• **Liang-Chieh Chen**, George Papandreou, and Alan L. Yuille. *Learning a Dictionary of Shape Epitomes with Applications to Image Labeling*, International Conference on Computer Vision (ICCV), 2013

CHAPTER 1

Introduction

Teaching computers to understand an image, such as image classification and semantic segmentation, has been a challenging problem in computer vision. In the context of supervised learning, where annotations are provided, the learning process can be summarized as two main steps: Feature extraction and Classifier training. The most important part is the feature extraction step, and researchers have spent a lot time on developing good image representations. Some researchers have been mainly focusing on how to design the features manually. Among them, the most famous feature representations include SIFT [Low04] and HoG [DT05], which have been shown to be insufficient for high-level computer vision tasks recently. On the contrary, instead of trying to improve over the hand-crafted features, we focus on learning image representations for scene understanding in this thesis.

To learn image representations for scene understanding, we mainly explore along three directions: (1) generative models, specifically epitomes, to model edge patches and image patches, (2) graphical models, specifically Conditional Random Fields (CRFs), for structured prediction, and (3) deep neural networks, specifically Deep Convolutional Neural Networks (DCNNs), for hierarchical image representations. This dissertation introduces a series of new methods, which combine the advantages from those three areas.

1.1 Contributions

The contributions of this thesis can be analyzed with respect to four axes: (1) local representation *vs*. hierarchical representation, (2) employing conditional random field for structured prediction or not, (3) joint training *vs*. piecewise training, and (4) learning with strong *vs*. weak supervision.

In this section, we first analyze the contributions along those four axes, and then we discuss the contribution of each chapter.

1.1.1 Overview of the contribution

In this subsection, we analyze the contribution of this thesis along the four axes:

Local *vs*. hierarchical representation Our dictionary of mini-epitomes represents local edge structures or local raw image patches. The dictionary is learned unsupervisedly and is compact in the sense that a mini-epitome is able to generate several similar edge templates or image patches with similar appearance. On the other hand, deep neural networks, trained supervisedly with large-scale datasets, learn feature representation hierarchically. Lower neural layers usually learn edge structures, and higher layers start to learn semantic structures, such as car wheels.

CRF *vs*. non-CRF Conditional Random Fields are a powerful mathematical tool to capture the output variable dependency. For example, neighboring pixels should have high probability to be assigned by the same semantic label, and thus it refines the final prediction result. We have shown that for strong classifiers, such as DCNNs, the prediction is usually very coarse due to the employed down-sampling, and applying a fully-connected CRF can significantly improve the segmentation result especially along the object boundaries.

Joint *vs*. piecewise training We also employed two schemes to train our models. Piecewise training is efficient as it optimizes each module separately. However, this piecewise training process is suboptimal since the training error gradients should focus on errors that can not be corrected by all the modules jointly. We also show that joint training results in significant performance gains over piecewise training.

Strong *vs*. weak supervision Training a model supervisedly requires a large amount of annotations. Besides, labeling large-scale datasets with very high quality is both time-consuming and expensive in terms of budget. Therefore, it is interesting to investigate training models with weak

annotations, which requires significantly less annotation effort. In this thesis, we have explored two settings. In the first setting, we explore training a car segmentation model with weak 3D bounding box supervision, while in the second setting, we explore training a deep convolutional neural network for semantic segmentation with simply weak 2D bounding box or image-level annotations. In both cases, we show that it is possible to train models with weak supervision and attains performance as good as the annotations by Mechanical Turkers or as when exploiting strong annotations for training.

We identify the contributions of each chapter with respect to the four axes in Tab. 1.1.

	Local representation	Graphical model (CRF)	Joint training	Weak supervision
Chapter 2	✓	✓		
Chapter 3	✓			
Chapter 4	✓	✓		✓
Chapter 5		✓	✓	
Chapter 6		✓		
Chapter 7		✓		✓

Table 1.1: The contributions of each chapter analyzed along four axes: models employing (1) local *vs*. hierarchical feature representation, (2) conditional random fields for structured prediction or not, (3) joint or piecewise training, and (4) weak or strong supervision during training.

1.1.2 Overview of the thesis

Herein, we briefly overview the contribution of each chapter.

Epitomes for patch shape We learn an edge representation unsupervisedly, called a dictionary of shape epitomes, in Chapter 2. These shape epitomes represent the local edge structure of the image and include hidden variables to encode shift and rotations. They are learnt in an unsupervised manner from groundtruth edges. This dictionary is compact but is also able to capture the typical shapes of edges in natural images. We illustrate the shape epitomes by applying them to the image

labeling task, and apply shape epitomes to image labeling by using Conditional Random Field (CRF) Models. They are alternatives to the superpixel or pixel representations used in most CRFs. In our approach, the shape of an image patch is encoded by a shape epitome from the dictionary. Unlike the superpixel representation, our method avoids making early decisions which cannot be reversed. Our resulting hierarchical CRFs efficiently capture both local and global class co-occurrence properties. We demonstrate its quantitative and qualitative properties of our approach with image labeling experiments on two standard datasets: MSRC-21 and Stanford Background.

Epitomes for patch intensity As an alternative to hand-crafted features (such as SIFT), we develop a generative model, in Chapter 3, for the raw intensity of image patches and show that it can support image classification performance on par with optimized SIFT-based techniques in a bag-of-visual-words setting. Key ingredient of the model is a compact dictionary of mini-epitomes, learned in an unsupervised fashion on a large collection of images. The use of epitomes allows us to explicitly account for photometric and position variability in image appearance. We show that this flexibility considerably increases the capacity of the dictionary to accurately approximate the appearance of image patches and support recognition tasks. For image classification, we develop histogram-based image encoding methods tailored to the epitomic representation, as well as an "epitomic footprint" encoding which is easy to visualize and highlights the generative nature of our model. We discuss in detail computational aspects and develop efficient algorithms to make the model scalable to large tasks. The proposed techniques are evaluated with experiments on the challenging PASCAL VOC-07 image classification benchmark.

3D information and weak supervision We explore a simple feature representation: employing Gaussian Mixture Models to capture the pixel RGB color distribution and 3D information for cars in Chapter 4. Specifically, we show how to exploit 3D information to automatically generate very accurate object segmentations given annotated 3D bounding boxes. We formulate the problem as the one of inference in a binary Conditional Random Field which exploits appearance models, stereo and/or noisy point clouds, a repository of 3D CAD models as well as topological constraints. We show that our method can segment cars with human-level accuracy (*i.e.*, performing as well as

highly recommended MTurkers).

Deep structured models We bring together the advantages from graphical models (specifically, CRFs) and DCNNs in Chapter 5 and Chapter 6. Many problems in real-world applications involve predicting several random variables that are statistically related. CRFs are a powerful mathematical tools to capture the correlation among random variables, while DCNNs have demonstrated state-of-art performance on recognition tasks. Combining CRFs with deep learning is able to estimate complex representations while taking into account the dependencies between the output random variables. In Chapter 5, we employ a training algorithm to jointly learn structured models with deep features that form the CRF potentials. Our method is efficient, as it blends learning and inference, and makes use of GPU accelerations. We show that joint learning of the deep features and the CRF parameters lead to significant performance gains in the tasks of word prediction and image tagging.

In Chapter 6, we employ the fully connected CRF [KK11] as a post-processing step for the outputs from DCNNs. Specifically, we show that responses at the final layer of DCNNs are not sufficiently localized for accurate object segmentation. This is due to the very invariance properties that make DCNNs good for high level tasks. We overcome this poor localization property of deep networks by combining the responses at the final DCNN layer with a fully connected CRF. Qualitatively, our "DeepLab" system is able to localize segment boundaries at a level of accuracy which is beyond previous methods. Quantitatively, our method sets the new state-of-art at the PASCAL VOC-2012 semantic image segmentation task. We show how these results can be obtained efficiently: Careful network re-purposing and a novel application of the 'hole' algorithm from the wavelet community allow dense computation of neural net responses at 8 frames per second on a modern GPU.

Deep structured models with weak supervision We study the challenging problem in which only weak annotations (such as image-level labels or bounding box annotations) or a small number of strong annotations (*i.e.*, pixel-level annotations) are available for training DCNNs in Chapter 7. Deep convolutional neural networks (DCNNs) trained on a large number of images with strong

pixel-level annotations have recently significantly pushed the state-of-art in semantic image segmentation. However, it is time-consuming and labor-intensive to collect the pixel-level annotations. We use Expectation-Maximization (EM) methods for training DCNNs under these weakly supervised and semi-supervised settings. Extensive experimental evaluation shows that our techniques can learn models delivering competitive results on the challenging PASCAL VOC 2012 image segmentation benchmark, while requiring significantly less annotation effort.

CHAPTER 2

Learning a Dictionary of Shape Epitomes

In this chapter, we employ a novel representation for local edge structure based on a dictionary of shape epitomes, which were inspired by [JFK03]. This dictionary is learnt from annotated edges and captures the mid-level shape structures. By explicitly encoding shift and rotation invariance into the epitomes, we are able to accurately capture object shapes using a compact dictionary of only five shape epitomes. We explore the potential of shape epitomes by applying them to the task of image labeling. Most modern image labeling systems are based on Conditional Random Fields (CRFs) [KH06, LMP01] for integrating local cues with neighborhood constraints. Image segments are typically represented in the pixel domain [Gou12, KK11, SWR09], or in the domain of superpixels (a region of pixels with uniform statistics) [FVS09, GRB08, GFK09, HZR06, LVZ11, MBH10].

One motivation for shape epitomes was the success of segmentation templates for image labeling [ZCL12]. These templates also represent the local edge structure but differ from pixels and superpixels because they represent typical edges structures, such as L-junctions, and hence provide a prior model for edge structures. Each patch in the image was encoded by a particular segmentation template with semantic labels assigned to the regions specified by the template, as illustrated in Fig. 2.1. Segmentation templates, like superpixels, have computational advantages over pixel-based approaches by constraining the search process and also allow enforcing label consistency over large regions. Compared to superpixels, segmentation templates do not make early decisions based on unsupervised over-segmentation and, more importantly, explicitly enumerate the possible spatial configurations of labels making it easier to capture local relations between object classes. See Table 2.1 for a comparison summary.

But those segmentation-templates [ZCL12] have limitations. Firstly, they were hand-specified.

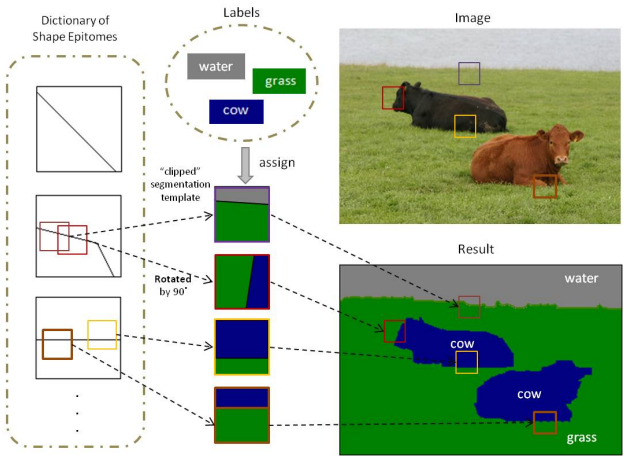


Figure 2.1: Proposed dictionary of *Shape Epitomes* in the context of image labeling. Segmentation templates are generated from the shape epitomes, by specifying the values of the hidden variables. Image labels are assigned to the regions within the templates, and thus the local relationship between object classes is explicitly modeled. Note the rotation and shift-invariance illustrated in the second and third shape epitome, respectively.

Secondly, there were not invariant to shift and rotation which implies that a very large number of them would be needed to give an accurate representation of edge structures in the images (Zhu *et al*. [ZCL12] used only thirty segmentation-templates which meant that they could only represent the edges very roughly).

Each shape epitome can be thought of a set of segmentation-templates which are indexed by hidden variable corresponding to shift and rotation. More precisely, a shape epitome consists of two square regions one inside the other. The hidden variable allows the inner square region to shift and rotate within the the bigger square, as shown in Fig. 2.1. The hidden variable specifies the shift and rotation. In the current work, each shape epitome corresponds to $81 \times 4 = 324$ segmentation-templates. Hence, as we will show, a small dictionary of shape epitomes is able to accurately represent the edge structures (see Sec. 2.3.3.1). Intuitively the learned dictionary captures generic mid-level shape-structures, hence making it transferable across datasets. By explicitly encoding shift and rotation invariance, our learned epitomic dictionary is compact and only uses five shape epitomes. We also show that shape epitomes can be generalized to allow the inner square to expand which allow the representation to deal with scale (see Sec. 2.3.3.4).

	Pixel	Superpixel	Template
Computation	−	+	+
Flexibility	+	+	+
Long-Range	−	+	+
Explicit Configuration	−	−	+

Table 2.1: General comparison between representations from the aspects of *Computation*, *Flexibility* (better align with object shapes), *Long-Range* consistency, and ability to *Explicitly* model the local *configuration* of objects. We improve the flexibility of template-based representation by learning a dictionary of shape epitomes.

We propose shape epitomes as a general purpose representation for edge structures (i.e. a mid-level image description). In this chapter we illustrate them by applying them to the image labeling task. For image labeling, we consider three increasingly more complex models, which adapt current CRF techniques for shape epitomes. We use patches at a single fine resolution whose shape is encoded by a segmentation template (i.e. a shape epitome with hidden variable specified). The patches are overlapping, thus allowing neighbors to directly communicate with each other and find configurations which are consistent in their area of overlap (Model-1). We explore two enhancements of this basic model: Adding global nodes to enforce image-level consistency (Model-2) and also further adding an auxiliary node to encourage sparsity among active global nodes, i.e., encourage that only few object classes occur within an image (Model-3). We conduct experiments on two standard datasets, MSRC-21 and Stanford Background, obtaining promising results.

Our model is based on the success of several works. First, the ability to generate an image from a condensed epitomic representation [JFK03]. We leverage on this idea to learn a dictionary of shape epitomes. Each segmentation template is generated within a *shape epitome*. This encodes the shift-invariance into the dictionary, since a segmentation template is able to move within a shape epitome. Besides, we encode rotation invariance by allowing the shape epitome to rotate by 0, 90, 180, and 270 degrees.

Second, the potential of using template-based representation and overlapped patches. It has

been shown that learning the generic patterns capturing statistics over large neighborhoods can be beneficial for image denoising [RB09] and image labeling [KP09]. Besides, finding the mutual consensus between neighboring nodes by using overlapped patches [ZW11] has shown to be effective. Similar ideas have been applied to image labeling [KBB11]. However, they did not learn a dictionary for object shapes.

Third, the power of introducing simple global nodes for image labeling. Ladicky *et al*. [LRK09] introduced global nodes that can take values from the predefined label set L and a "free" label. There is no energy cost, when the global node takes the free label. Gonfaus *et al*. [GBW10] proposed a *harmony* model to generalize the idea by allowing the global node to take labels from the power set of L. However, it is computationally challenging to find the most probable state for the global node from the power set. Then, Lucchi *et al*. [LLB11] proposed the Class Independent Model (CIM) to decompose the global node into $|L|$ global nodes. Our model moves further based on the CIM by encoding the image-level co-occurrence, and adding an auxiliary node to encourage the sparsity of active global nodes, similar to [DOI12].

2.1 Learning a dictionary of shape epitomes

In this section, we present our algorithm for learning the dictionary of shape epitomes from annotated images.

To learn the generic dictionary, we use the BSDS500 dataset [AMF11], which provides ground truth of object boundaries. Given that, we extract $M \times M$ patches around the shape boundaries (called shape patches). We cluster these shape patches using affinity propagation [FD07] to build our shape epitomes (note that the size of shape patches is the same as that of shape epitomes). The segmentation templates are of smaller size $m \times m$ ($m < M$) than the shape epitomes, and are generated as sub-windows of them. By generating the segmentation template from a larger shape epitome, we are able to explicitly encode shift-invariance into the dictionary, as illustrated in Fig. 2.1. Therefore, one shape epitome compactly groups many segmentation templates which are shifted versions of each other.

Clustering by affinity propagation requires a similarity measure $F(P_1, P_2)$ between two $M \times$

M shape patches P_1 and P_2. We induce $F(P_1, P_2)$ from another similarity measure $F_T(T_1, T_2)$ between two $m \times m$ segmentation templates T_1 and T_2 extracted from P_1 and P_2, respectively. Specifically, let $T(i, j)$ denote the segmentation template extracted from P and centered at (i, j), with $(0, 0)$ being the center of P. We define the similarity between the two shape patches P_1 and P_2 to be

$$F(P_1, P_2) \quad = \quad \max_{\frac{m-M}{2} \le i,j \le \frac{M-m}{2}} \frac{1}{2}[F_T(T_1(i, j), T_2(0, 0)) \; + \; F_T(T_1(0, 0), T_2(-i, -j))] \,, \quad (2.1)$$

as illustrated in Fig. 2.2. We employ the *covering* of the template T_1 by the template T_2 [AMF11] as the similarity measure $F_T(T_1, T_2)$ between them:

$$F_T(T_1, T_2) = \frac{1}{|T_2|} \sum_{r_2 \in T_2} |r_2| \max_{r_1 \in T_1} \frac{|r_1 \cap r_2|}{|r_1 \cup r_2|} \,,$$

where r_1 and r_2 are the regions in templates T_1 and T_2, respectively, and $|r|$ is the area of region r. Note that $F_T(T_1, T_2)$ and consequently $F(P_1, P_2)$ range from 0 (no similarity) to 1 (full similarity).

Directly applying affinity propagation results in many similar shape epitomes because simple horizontal or vertical boundaries are over-represented in the training set. We follow [JT05] and grow the dictionary incrementally, ensuring that each newly added shape epitome is separated from previous ones by at least distance t, as follows:

1. *Clustering.* Apply affinity propagation to find one shape epitome that contains the most members (i.e., the largest cluster) in current training set.
2. *Assigning.* For each shape patch in training set, assign it to the shape epitome found in step 1, if their distance, defined as $1 - F(P_1, P_2)$, is smaller than t.
3. *Update.* Remove the shape patches that are assigned to the shape epitome from the current training set.
4. Repeat until no shape patch is left in the training set.

2.2 Adapting CRFs for segmentation templates

Having learned the dictionary of shape epitomes, we now proceed to show how we can build models for image labeling on top of it. We propose three models by adapting current CRF models

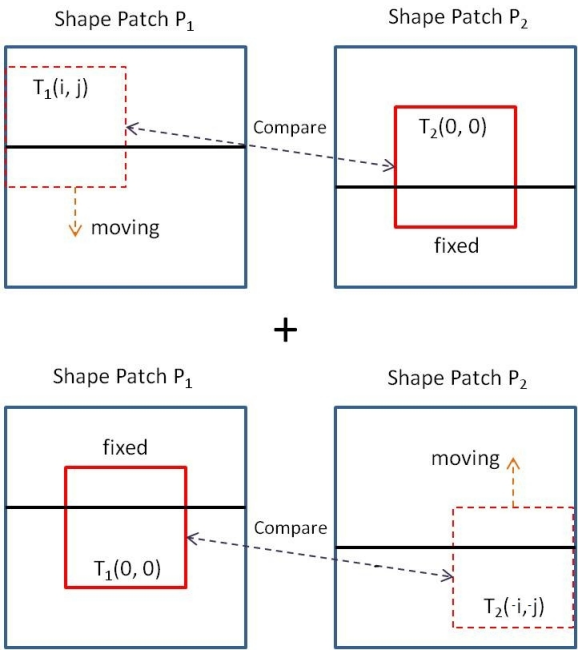


Figure 2.2: The similarity measure between two shape patches. The optimal value of shift variables (i, j) is shown for this example.

to the template-based representation.

The problem of image labeling in this context can be formulated as follows. Given an image I, we represent it by a set of overlapped $m \times m$ patches. The goal is to encode each patch by a segmentation template, and by assigning labels (from a categorical set L) to each region in the segmentation template. Specifically, the labeling assignment $\boldsymbol{x}$ is represented by both segmentation template and labels. That is, $\boldsymbol{x} = \{x_i\}_{i \in \mathcal{V}}$ with $x_i = \{s_i, \boldsymbol{l}_i\}$, where $\mathcal{V}$ is the set of patches, s_i and $\boldsymbol{l}_i$ denote the type of segmentation template and object labeling, respectively. Note that $\boldsymbol{l}_i$ is a vector, whose length is the number of regions within the segmentation template. For example, $\boldsymbol{l}_i = (cow, grass)$ means that label cow and label grass are assigned to the first region and second region within segmentation template s_i. We call our models SeCRF, short for Shape epitome CRF.

2.2.1 Model 1: One-level SeCRF

We first introduce a flat model, which is represented by a graph with a single layer $\mathcal{G} = \{\mathcal{V}_l, \mathcal{E}_l\}$, as shown in Fig. 2.3(a). Each node corresponds to a patch region, and it is encoded by both the

type of segmentation template and the labels assigned to it. The image region represented by node i (i.e., i-th patch) is denoted by $R(i)$.

The energy of $\boldsymbol{x}$ given image I is given by:

$$E(\boldsymbol{x}|I) \quad = \quad E_1(\boldsymbol{x};\alpha_1) \;+\; E_2(\boldsymbol{x};\alpha_2) \;+\; E_3(\boldsymbol{s};\alpha_3) \;+\; E_4(\boldsymbol{l};\boldsymbol{\alpha_4}) \;+\; E_5(\boldsymbol{x};\boldsymbol{\alpha_5}) \quad (2.2)$$

where $\boldsymbol{\alpha}$ is the model parameters. Note we suppress the dependency on the image I in subsequent equations. Each energy term is defined below.

The first term $E_1(\boldsymbol{x};\alpha_1)$ is the data term which accumulates the pixel features with respect to certain type of segmentation template and labels assigned to the corresponding regions. We set $E_1(\boldsymbol{x};\alpha_1) = -\alpha_1 \sum_{i\in\mathcal{V}_l} \psi_1(x_i)$, and

$$\psi_1(x_i) = \frac{1}{|R(i)|} \sum_{p\in R(i)} \log Pr(x_i^p|I)$$

where we define x_i^p as the labeling of pixel p in the region of segmentation template s_i. The value $Pr(x_i^p|I)$ is computed by a strong classifier with features (e.g., filter bank responses) extracted within a region centered at position p.

The second term is used to encourage the consistency between neighboring nodes in their area of overlap. For a pixel that is covered by both node i and j, we encourage node i to assign the same label to it as node j. The consistency is defined by using the Hamming distance:

$$E_2(\boldsymbol{x};\alpha_2) = -\alpha_2 \sum_{(i,j)\in\mathcal{E}_l} \psi_2(x_i, x_j)$$

where

$$\psi_2(x_i, x_j) = \frac{1}{|O(i,j)|} \sum_{p\in O(i,j)} \delta(x_i^p = x_j^p)$$

where $O(i,j)$ is the overlapped region between nodes i and j, and $\delta(x_i^p = x_j^p) = 1$ if $x_i^p = x_j^p$, and zero, otherwise. In our experiments, we use 4-neighborhood.

The third term encodes the generic prior of segmentation templates. Specifically, we binarize the type of s_i to be either 1, meaning that it contains some type of shapes, or 0, meaning that it contains no shape.

$$E_3(\boldsymbol{s};\alpha_3) = -\alpha_3 \sum_{i\in\mathcal{V}_l} \log Pr(s_i)$$

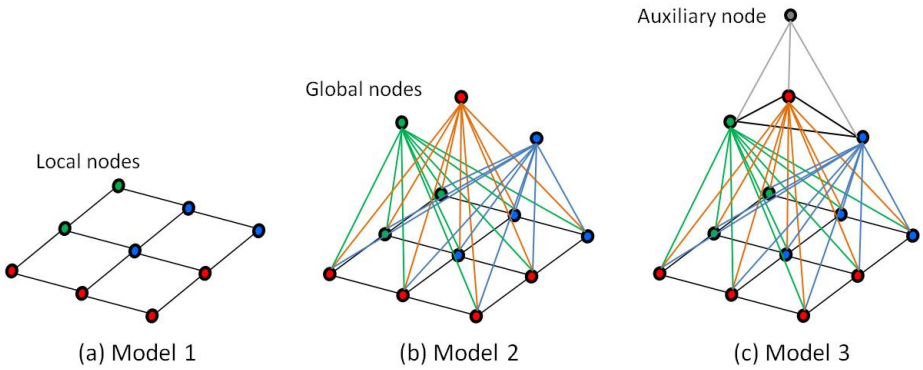


Figure 2.3: Adapting CRFs for segmentation templates. (a) Model 1 uses only a single layer of local nodes. (b) Model 2 adds global nodes to encode global consistency, similar to [LLB11] (but the energy value is soft in our model). (c) Model 3 encodes the pairwise co-occurrence between global nodes, and adds an auxiliary node to encourage the sparsity of active global nodes.

The fourth term $E_4(\boldsymbol{x};\boldsymbol{\alpha}_4)$ is used to model the co-occurrence of two object classes within a segmentation template. Note that parameter $\boldsymbol{\alpha}_4$ is a 2-D matrix, indexed by u and v, ranging over the label set L.

$$E_4(\boldsymbol{l};\boldsymbol{\alpha}_4) = -\sum_{i\in\mathcal{V}_l}\sum_{u,v=1,\ldots,|L|}\alpha_4(u,v)\psi_4(u,v,\boldsymbol{l}_i)$$

where $|L|$ is the total number of object classes, and $\psi_4(u,v,\boldsymbol{l}_i)$ is an indicator function which equals one when both object classes u and v belong to $\boldsymbol{l}_i$.

The fifth term $E_5(\boldsymbol{x};\boldsymbol{\alpha}_5)$ models the spatial relationship between two classes within a segmentation template. We model only the "above" relationship. For example, we encourage sky to appear above road, but not vice versa.

$$E_5(\boldsymbol{x};\boldsymbol{\alpha}_5) = -\sum_{i\in\mathcal{V}_l}\sum_{u,v=1,\ldots,|L|}\alpha_5(u,v)\psi_5(u,v,x_i)$$

where $\psi_5(u,v,x_i)$ is an indicator function which equals one when object class m is above class n within a certain segmentation template. Note that for some segmentation template that does not have the "above" relationship (e.g., a template with vertical boundary), this term is not used.

2.2.2 Model 2: Two-level SeCRF

Motivated by the Class Independent Model (CIM) in [LLB11], we add $|L|$ independent global nodes $\{\mathcal{V}_g\}$ to enforce image-level consistency, as shown in Fig. 2.3(b). A global node encodes the absence or presence of a object class in the image (i.e., $y_i \in \{0,1\}, \forall i \in \mathcal{V}_g$), and it is densely connected to every local node. We denote the set of edges connecting global nodes and local nodes as $\{\mathcal{E}_{lg}\}$, and then labeling assignment $\boldsymbol{x} = \{\{x_i\}_{i\in\mathcal{V}_l} \cup \{y_i\}_{i\in\mathcal{V}_g}\}$. An extra global-local energy term is added to Equation 2.2 with each global node y_j having a 2-D matrix parameter $\boldsymbol{\alpha}_6^j$:

$$E_6(\boldsymbol{x};\boldsymbol{\alpha}_6) = -\sum_{(i,j)\in\mathcal{E}_{lg}}\sum_{u=1}^{|L|}\sum_{v=0}^{1}\alpha_6^j(u,v)\psi_6(u,v,x_i,y_j)$$

where

$$\psi_6(u,v,x_i,y_j) = \begin{cases} \dfrac{1}{|R(i)|}\displaystyle\sum_{p\in R(i)}\delta(x_i^p = u), & \text{if } y_j = v \\ 0, & \text{otherwise} \end{cases}$$

Note that our Model 2 differs from CIM in two parts. First, the value of function ψ_6 is proportional to the number of pixels whose labels are u in the node x_i. This formulation is different from the energy cost used in the original CIM, which is either zero or one (i.e., a hard value). On the contrary, we formulate this energy cost as a soft value between zero and one. Second, our local nodes are based on overlapped segmentation templates (not superpixels) so that neighbors can directly communicate with each other. Furthermore, unlike the robust P^n model [KLT09], our penalty depends on the region area within a segmentation template, and thus it is a function of the segmentation template type.

2.2.3 Model 3: Three-level SeCRF

We further refine Model 2 by adding image-level classification scores to the unary term of global nodes [SJC08]. Specifically, we train $|L|$ SVM classifiers to predict the presence or absence of object classes, following the pipeline of [CLV11]. The unary energy for global nodes is then defined as follows.

$$E_7(\boldsymbol{y};\alpha_7) = -\alpha_7\sum_{i\in\mathcal{V}_g}C(y_i|I)$$

where $C(y_i|I)$ is the output of i-th classifier.

The independency among global nodes in Model 2 ignores the co-occurrence between object classes in the image level. Hence, we add edges $\{\mathcal{E}_g\}$ to connect every pair of global nodes, and define an energy term on them:

$$E_8(\boldsymbol{y};\boldsymbol{\alpha}_8) = -\sum_{(i,j)=e\in\mathcal{E}_g}\sum_{u,v=0}^{1}\alpha_8^e(u,v)\delta(y_i=u,y_j=v)$$

where $\boldsymbol{\alpha}_8^e$ depends on the specific edge $e=\{i,j\}$ that connects two different global nodes, y_i and y_j.

As shown in Fig. 2.3(c), we also add an auxiliary node $\mathcal{V}_a$ (then, $\boldsymbol{x}=\{\{x_i\}_{i\in\mathcal{V}_l}\cup\{y_i\}_{i\in\mathcal{V}_g}\cup\{z_i\}_{i=\mathcal{V}_a}\}$). This node favors sparsity among global nodes (similar to [DOI12]) by introducing a set of edges $\{\mathcal{E}_{ga}\}$ from $\{\mathcal{V}_g\}$ to $\mathcal{V}_a$. Specifically, $\mathcal{V}_a$ is a dummy node, which can take only one meaningless state. We define an energy term on $\{\mathcal{E}_{ga}\}$ to encourage only few global nodes to be active as follows.

$$E_9(\boldsymbol{y},z_j;\alpha_9) = -\alpha_9\sum_{(i,j)\in\mathcal{E}_{ga}}\delta(y_i=0)$$

where $\delta(y_i=0)$ equals one when the global node y_i is off. This energy term has the effect of biasing the global nodes.

2.3 Experimental Evaluation

In this section, we first show the results of learning a dictionary of shape epitomes following the methods described in Sec. 2. We then use this dictionary for image labeling using the SeCRF models of Sec. 3.

2.3.1 Learned dictionary of shape epitomes

We learn the dictionary of shape epitomes from shape patches extracted from the BSDS500 dataset [AMF11]. In the experiment, we fix the size of a shape patch to be 25×25, and the size of segmentation template 17×17, namely $M=25$ and $m=17$. After applying affinity propagation incrementally with distance $t=0.05$, the first 10 shape epitomes are shown at the top of Fig. 2.4.

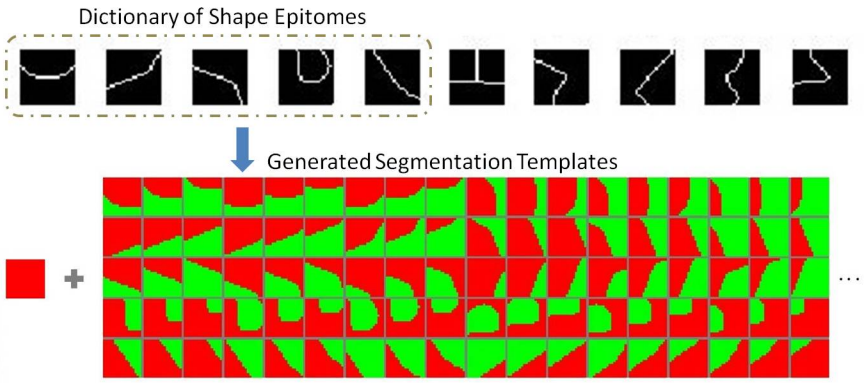


Figure 2.4: Top row: first 10 shape epitomes learned by our method. Bottom: a flat segmentation template (i.e., no shape) and some others generated from the first 5 shape epitomes. Note that some of them are generated from the rotated shape epitomes.

For computational and modeling purposes it is desirable to have a compact dictionary consisting of only few shape epitomes. We have found that the first 5 shape epitomes contain most of the shape patches in the training set of BSDS500, and the cluster size decreases very quickly.

In our setting, a segmentation template is allowed to move within a shape epitome for each horizontal and vertical displacement up to ± 4 pixels. We define *stride* as the step-size for horizontal/vertical displacement. For example, if stride $= 4$, we can generate 9 templates from each shape epitome, only considering the nine templates $T(i, j)\ \forall i, j \in \{-4, 0, 4\}$ at all four possible orientations (0, 90, 180 and 270 degrees), ending up with $45 = 9 \times 5$ templates per epitome. In total, there are $181(5 \times 45 + 1)$ segmentation templates, including the flat one that contains no shape. On the other hand, if stride $= 1$, we use every template within a shape epitome, resulting in 1621 $(81 \times 5 \times 4 + 1)$ segmentation templates.

Using this compact dictionary of 5 shape epitomes suffices to accurately encode the ground truth segmentations in our datasets, as demonstrated in Sec. 2.3.3.1. As one can observe in Fig. 2.4, our generated segmentation templates cover the common boundary shapes, such as vertical/horizontal edges, L-junctions, and U-shapes. The learned dictionary thus captures generic mid-level shape-structures and can be used across datasets. We emphasize that we learn it on the BSDS500 dataset and use it unadapted for image labeling on MSRC-21 and Stanford Background datasets.

2.3.2 Implementation details for image labeling

MAP Inference. We use loopy belief propagation (LBP) to minimize the energy function in Equation 2.2. We prune the unpromising states by rejecting the unlikely proposals whose E_1 data terms are too high, similar to [ZCL12]. We fix the number of states per node to be 100, since in our experiments adding more states only improve the performance marginally at the sacrifice of computation time.

Learning the parameters. We use the same structure-perceptron algorithm [Col02] as HIM [ZCL12], because we would like to have a direct comparison with it by emphasizing on the representation part of our model, not learning.

Fusion of predicted labels. The traditional Conditional Random Field models directly assign an object class label to each pixel in the image. On the contrary, our model uses overlapped patches, and each patch is encoded by a segmentation template and by labels assigned to the regions in the template. The number of patches that will cover the same pixel depends on the size of overlap between patches. We set the overlap size to be $(m-1)/2$ pixels in all experiments. To find the labels for every pixel, we fuse the predicted labels for each pixel by letting the patch having the minimal unary energy ($E_1 + E_3 + E_4 + E_5$) determine the final result of the covered pixel, since the pairwise term E_2 already encourages consistency.

2.3.3 Results

For image labeling, we experiment on two datasets: (1) The MSRC-21 with 591 images and $|L| = 21$ classes, using the original splitting (45% for training, 10% for validation, and 45% for testing) from [SWR09]. (2) The Stanford Background dataset [GFK09] consisting of 715 images and $|L| = 8$ classes, which we randomly partition into training set (572 images) and test set (143 images). Note in all the experiments, we fix $M = 25$, and $m = 17$ except in Sec. 2.3.3.4.

2.3.3.1 Encoding the ground truth

The ground truth provided by the datasets contains the true labeling for each pixel, not the true states of segmentation template type with regions labeled. This experiment is designed to see if our learned dictionary of shape epitomes can accurately encode the ground truth. We estimate the true states of the local nodes by selecting the pairs of segmentation template type and labeling (i.e., find true $x_i = (s_i, \boldsymbol{l}_i)$) that have maximum overlap with the true pixel labels. For MSRC-21 dataset, our result shows that this encoding of ground truth results in 0.27% error in labeling image pixels, while HIM [ZCL12] reported 2% error. This shows that our learned dictionary of shape epitomes is flexible enough to more accurately encode the MSRC ground truth than the hand-crafted dictionary of [ZCL12].

Here, we show the advantage of using our learned dictionary of shape epitomes over *directly* learning a dictionary of segmentation templates (in the latter case, the training shape patches have size $m \times m$ instead of $M \times M$) by conducting experiments on the Stanford Background dataset, which provides more detailed object boundaries. We propose to compare those two dictionaries in terms of the error of encoding the ground truth, when given the same covered areas, which is equivalent to learning the same number of parameters. Suppose the size of the dictionary of shape epitomes is K_E, and the size of the dictionary of segmentation templates is K_T. Given K_E, to cover the same areas, we select $K_T = 25^2/17^2 K_E$. As shown in Fig. 2.5, our learned dictionary of shape epitomes attains better performance than the dictionary of segmentation templates when given the same number of parameters.

2.3.3.2 Image labeling: MSRC-21 dataset

We generate 9 segmentation templates from each of the 5 shape epitomes in the labeling experiments (i.e., 181 templates totally). In a first set of experiments we directly compare our models with HIM [ZCL12]. We use the same boosting-based data term as HIM, provided by the authors, the main difference between HIM and our model lying in the representation part. As shown in Fig.2.7, our learned dictionary encodes the object shapes better than the hand-crafted dictionary used by HIM. Furthermore, both our Model 2 and Model 3 attain better performance than HIM

(see Table 2.3).

We also compare our model with the recent method of [Gou12] which incorporates powerful non-local patch similarity. We have used the same boosting-based data term as [Gou12], as implemented in the Darwin software library[1]. As shown in Table 2.3, our Model 3 attains similar performance to [Gou12], although we do not use non-local cues at the patch level.

2.3.3.3 Image labeling: Stanford background dataset

In this experiment, we use the data term provided by the Darwin software library. The results for the Stanford Background dataset are shown in Fig. 2.8. We achieve comparable results with other state-of-the-art models. Specifically, our segmentation template-based Model 3 performs better than the more complicated model of [GFK09], which builds on a dynamic superpixel representation and incorporates both semantic and geometric constraints in a slow iterative inference procedure. We also perform better than the hierarchical semantic region labeling method of [MBH10]. Our models perform somewhat worse than the long-range model of [Gou12] (unlike the MSRC case), and the segmentation tree model of [LVZ11], which however employs different image features.

2.3.3.4 Scaling the segmentation templates

Here, we show that our learned dictionary can generate different sizes of segmentation templates, while attaining good performance on the Stanford Background dataset. Specifically, we explore the effect of varying the size of generated segmentation templates as the dictionary of shape epitomes is fixed. First, we explore the effect by encoding the ground truth. The size varies from $m = \{13, 17, 21, 25\}$. The stride variable is also changed to generate different number of segmentation templates from the dictionary. As shown in Fig. 2.6, the error is consistently decreased when m or stride is smaller. Second, we extract spatially equally 9 segmentation templates from the dictionary for different m (all resulting in 181 templates), and apply our Model 1 based on these templates to label the test images, as shown in Table 2.2. These results show that our proposed representation:

[1] http://drwn.anu.edu.au, version 1.2

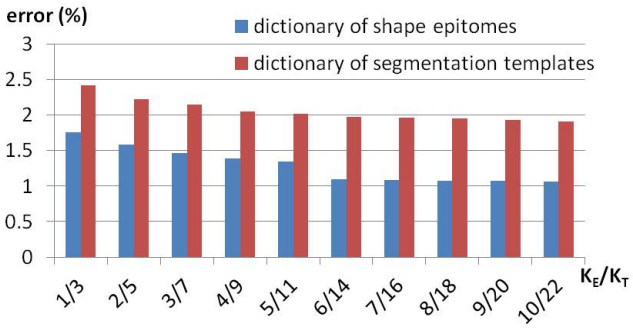


Figure 2.5: Error (%) of encoding ground truth of Stanford Background dataset, when using a dictionary of K_E shape epitomes or a dictionary of K_T segmentation templates.

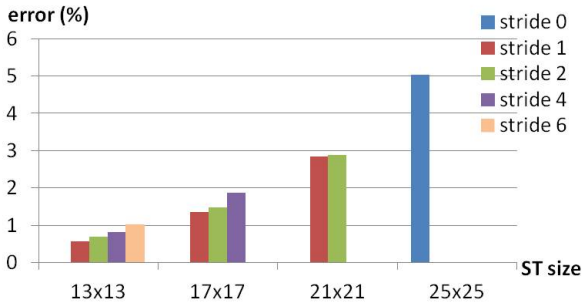


Figure 2.6: Error (%) of encoding ground truth of Stanford Background dataset. The dictionary of shape epitomes is fixed. The size of generated templates is different, and so is the stride.

shape epitomes is also able to handle scale effects without relearning the dictionary.

2.4 Conclusion

In this chapter, we introduced shape epitomes and showed that they could efficiently encode the edge structures in the MSRC and Stanford Background datasets. This efficient encoding is due to their ability to represent local shifts and rotations explicitly. The dictionary of shape epitomes were learnt from BSDS500 dataset. Next we explored the use of shape epitomes for CRF models of image labeling. The proposed SeCRF model can attain comparable results with other state-of-the-art models.

In Chapter 6, we revisit the semantic segmentation task by combining a more powerful unary term (*i.e.*, deep convolutional neural network) and a fully connected Conditional Random Field

Template size	13 × 13	17 × 17	21 × 21
Global	76.9	76.7	76.3

Table 2.2: Reuse the dictionary of shape epitomes with different size of generated templates on Stanford Background dataset.

		building	grass	tree	cow	sheep	sky	airplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Average**	**Global**
Exp1	Pxl-Cls	59.4	95.8	85.2	73.8	74.4	91.6	80.3	66.3	82.9	63.3	84.5	59.3	49.7	40.4	82.4	65.3	73.8	61.3	37.8	65.5	17.6	67.2	75.9
use same data	Model 1	62.7	96.2	87.9	78.7	78.6	92.6	83.7	66.8	85.5	69.2	87.1	63.9	53.6	45.3	85.7	71.8	75.4	66.5	41.9	68.4	17.5	70.4	78.1
term as HIM	Model 2	66.2	97.7	88.9	88.0	85.7	91.9	82.8	72.6	85.7	80.1	89.8	66.6	64.0	54.1	90.4	74.1	78.9	60.3	53.8	71.3	15.3	74.2	81.4
	Model 3	69.1	97.7	88.5	86.5	84.0	91.6	82.7	70.7	85.6	80.4	90.3	68.5	62.5	67.6	90.7	73.0	79.0	73.3	50.9	69.8	13.1	75.0	81.7
	HIM [ZCL12]	66.5	96.2	87.9	82.3	83.3	91.4	80.7	65.7	89.0	79.0	91.9	78.5	69.9	44.5	92.6	80.3	78.2	77.6	41.2	71.9	13.1	74.1	81.2
Exp2	Pxl-Cls	55.9	95.9	82.0	77.1	71.1	90.3	72.4	69.1	79.7	54.3	78.7	62.2	42.5	38.8	64.3	58.0	84.4	59.4	39.8	64.4	27.8	65.2	74.5
use data term	Model 1	63.9	96.9	86.6	81.9	75.2	91.9	76.1	72.8	81.3	59.9	84.4	65.8	45.5	41.7	66.1	61.9	87.7	64.0	43.4	67.5	29.4	68.7	77.6
from Darwin	Model 2	71.6	98.3	91.9	90.1	76.1	94.5	68.3	78.3	82.2	57.3	84.7	74.0	44.7	35.8	73.6	55.4	88.7	67.0	44.9	61.8	23.6	69.7	80.4
software library	Model 3	73.0	98.1	92.3	91.4	78.4	94.4	70.6	77.2	82.5	60.2	86.2	73.4	48.4	35.3	76.8	60.3	89.0	68.0	44.6	63.1	22.2	70.7	81.1
	[Gou12]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	71.1	81.0
some	[LRK09]	80	96	86	74	87	99	74	87	86	87	82	97	95	30	86	31	95	51	69	66	9	75	86
state-of-the-art	[MBH10]	63	93	88	84	65	89	69	78	74	81	84	80	51	55	84	80	69	47	59	71	24	71	78
models	[GBW10]	60	78	77	91	68	88	87	76	73	77	93	97	73	57	95	81	76	81	46	56	46	75	77
	DPG [LLB11]	65	87	87	84	75	93	94	78	83	72	93	86	70	50	93	80	86	78	28	58	27	76	80
	[KK11]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	78.3	86.0

Table 2.3: MSRC labeling results. Pixel-wise classification rates are provided for each category. **Global** accuracy refers to the pixel-wise classification rate averaged over the whole dataset, and **Average** accuracy refers to the mean of all object class classification rates. The Pxl-Cls model is the pixel-wise classifier, whose output is integrated in our models.

for refining segmentation results along object boundaries with long range connection. We show state-of-art results by exploiting those two modules.

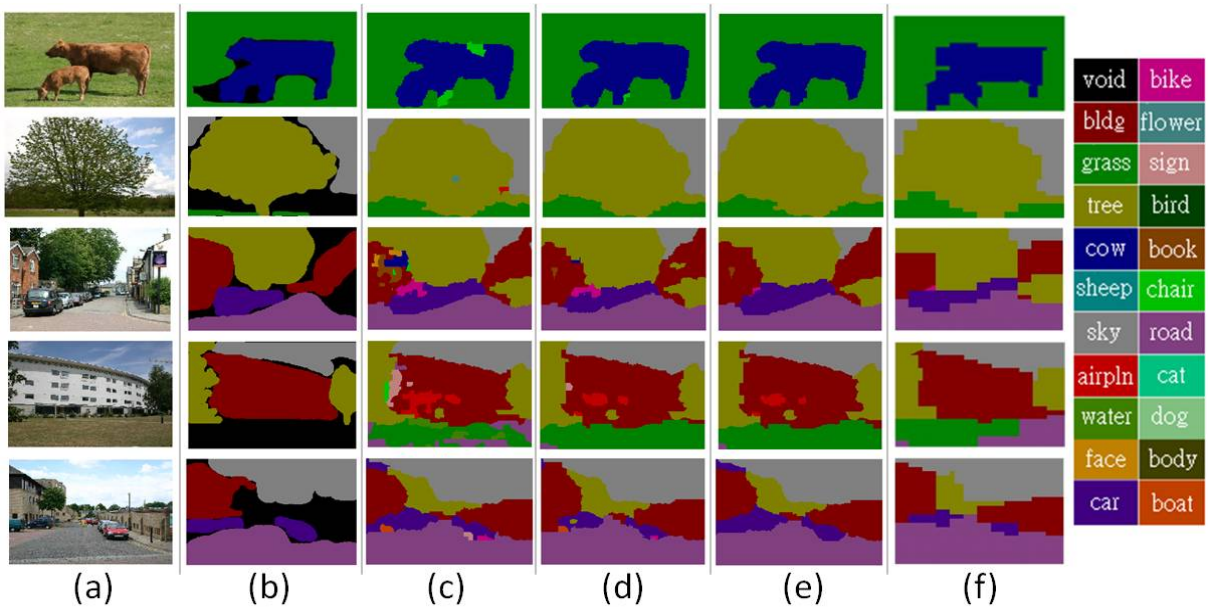
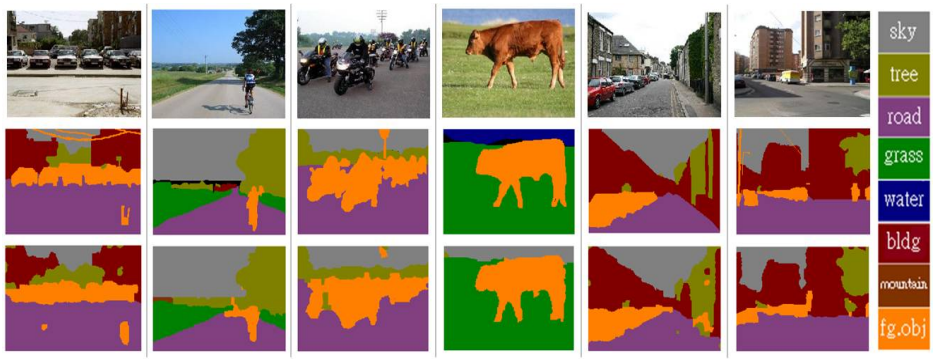


Figure 2.7: Qualitative results for the MSRC dataset. (a) Original image. (b) Ground truth. (c) Model 1. (d) Model 2. (e) Model 3. (f) HIM (excerpted from [ZCL12]). Note that our models capture object shapes more accurately than the HIM.



(a) (top) Original image. (middle) Ground truth. (bottom) Model 3. Note that our model is able to capture object shapes, especially the cow shape in the fourth column.

Method	**Global**
Pxl-Cls	73.9
Model 1	76.7
Model 2	77.2
Model 3	77.4
[GFK09]	76.4
[MBH10]	76.9
[Gou12]	79.6
[LVZ11]	81.9

(b) **Global** accuracy is the pixel-wise classification rate averaged over dataset.

Figure 2.8: Qualitative and quantitative results on the Stanford Background dataset.

CHAPTER 3

Modeling Image Patches with a Generic Dictionary of Mini-Epitomes

Our goal in this work is to investigate to which extent generative image models can also be competitive for visual recognition tasks. We use the raw image patch intensity as the fundamental representation in our model. Appearance patches have been successfully applied so far mostly in image generation tasks such as texture synthesis, image denoising, and image super-resolution [FPC00, EF01, AEB06, YSM12, ZW11].

Using raw appearance patches maximally preserves information in the original image. The main challenge with this modeling approach in image classification tasks is that the associated image description can be too sensitive to nuisance parameters such as illumination conditions or object position. Therefore, most computer vision systems for image categorization and recognition rely on features built on top of discriminative patch descriptors like SIFT [Low04]. SIFT has been explicitly designed for invariance to these nuisance parameters, which allows it to work reliably in conjunction with simple classification rules in a bag of visual words framework [ZML07, LSP06]. However, the SIFT and other similar descriptors are not suitable for image generation tasks and are very difficult to visualize [WJP11].

Instead of designing an image descriptor to be maximally invariant from the ground up, we attempt to explicitly model photometric and position nuisance parameters as attributes of a generative patch-based representation. Specifically, we develop a probabilistic epitomic model which can faithfully reconstruct the raw appearance of image patches using a compact dictionary of mini-epitomes learned from a large set of images. Our first main contribution is to show that explicitly matching the image patches to their best position in the mini-epitomes greatly improves reconstruc-

tion accuracy compared to a non-epitomic baseline which does not cater for position alignment. This allows us to accurately capture the appearance of image patches by a generic, rather than image specific, dictionary.

We design image descriptors for image classification tasks based on the proposed mini-epitomic dictionary. Our second main contribution is to show that bag-of-words type classifiers built on top of our epitomic representation not only improve over ones built on non-epitomic patch dictionaries, but also yield classification results competitive to those based on the SIFT representation. Beyond histogram-type encodings, we also investigate an "epitomic footprint" encoding which captures how the appearance of a specific image deviates from the appearance of the generic dictionary. This epitomic footprint descriptor can be visualized or stored as a small image and at the same time be used directly as feature vector in a linear SVM image classifier.

Employing the proposed model requires finding the best match in the epitomic dictionary for each patch in an image. We have experimented both with an exact search algorithm implemented in CUDA as well as with approximate nearest neighbor techniques. Both allow efficient epitomic patch matching and image encoding in about 1 sec for 400×500 images and typical settings for the model parameter values, making the model scalable to large datasets. In this chapter, we report image classification results on the challenging PASCAL VOC-07 image classification benchmark [EVW10] and compare the performance of our model both with the non-epitomic baseline and the modern SIFT-based classification techniques reviewed by [CLV11].

Key element of the proposed method is the explicit modeling of patch position using mini-epitomes. The epitomic image representation and the related idea of transformation invariant clustering were developed in [JFK03, FJ03] and also used in [ZGW05] for texture modeling, but have never been applied before for learning generic visual dictionaries on large datasets and in the context of visual recognition tasks.

The idea to use a patch-based representation for image classification first appeared in [PP97] and was further developed by [VZ05], who applied it to homogeneous texture classification and compared it to the filterbank-based representation of [LM01]. Recently, [CLN11] demonstrated competitive image classification results on the CIFAR-10 dataset of small images with patch dic-

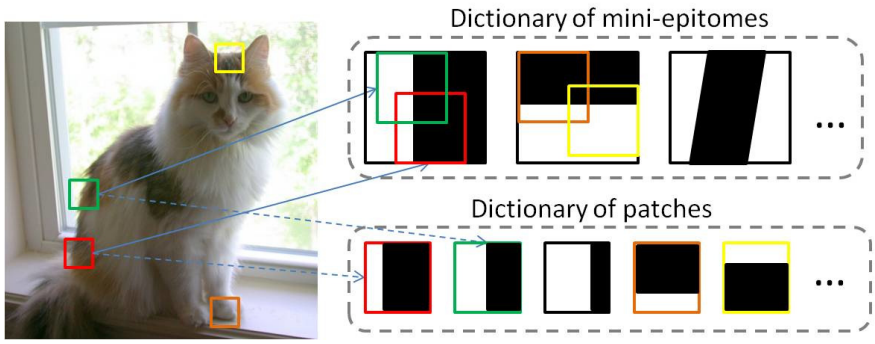


Figure 3.1: Unlike a conventional patch dictionary, the epitomic representation allows each image patch to move so as to find its best match within a mini-epitome.

tionaries trained by K-means. Neither of these works explicitly handles patch position or demonstrates performance comparable to modern SIFT encodings [CLV11] on challenging large-scale classification tasks.

More generally, unsupervised learning of image features has received considerable attention recently. Most related to our work is [ZKT10], which also attempts to explicitly model the position of visual pattern in a deconvolutional model. However, their model requires iteratively solving a large-scale sparse coding problem both during train and test time. The image classification performance they report significantly lags modern SIFT-based models such as those described in [CLV11], despite the fact that they learn a multi-layered feature representation. The power of learned patch-level features has also been demonstrated recently in [BRF11, DHH12, RR13]. Using mini-epitomes instead of image patches could also prove beneficial in their setting.

Sparsity and ICA provide a compelling framework for learning image patch dictionaries [OF96, BS97]. Sparsity coupled with epitomes has been explored in [AE08, BMB11] but these works focus on learning dictionaries on a single or a few images. While each image patch is represented as a linear combination of a few dictionary elements in sparse models, it is approximated by just one dictionary element in our model. One can thus think of the proposed model as an extremely sparse representation, or alternatively as an epitomic form of vector quantization [GG92].

3.1 Image Modeling with Mini-Epitomes

3.1.1 Model description

With reference to Fig. 3.1, let $\{\mathbf{x}_i\}_{i=1}^N$ be a set of possibly overlapping patches of size $h \times w$ pixels cropped from a large collection of images. Our dictionary comprises K mini-epitomes $\{\boldsymbol{\mu}_k\}_{k=1}^K$ of size $H \times W$, with $H \geq h$ and $W \geq w$. The length of the vectorized patches and epitomes is then $d = h \cdot w$ and $D = H \cdot W$, respectively. We approximate each image patch $\mathbf{x}_i$ with its best match in the dictionary by searching over the $N_p = h_p \times w_p$ (with $h_p = H - h + 1$, $w_p = W - w + 1$) distinct sub-patches of size $h \times w$ fully contained in each mini-epitome. Typical sizes we employ are 8×8 for patches and 16×16 for mini-epitomes, implying that each mini-epitome can generate $N_p = 9 \cdot 9 = 81$ patches of size 8×8. Our focus is on representing every image with a common vocabulary of visual words, so we use a single universal epitomic dictionary for analyzing image patches from any image. We have been working with datasets consisting of overlapping patches extracted from thousands of images and with dictionaries containing from $K = 32$ up to 2048 mini-epitomes.

We model the appearance of image patches using a Gaussian mixture model (GMM). We employ a generative model in which we activate one of the image epitomes $\boldsymbol{\mu}_k$ with probability $P(l_i = k) = \pi_k$, then crop an $h \times w$ sub-patch from it by selecting the position $p_i = (x_i, y_i)$ of its top-left corner uniformly at random from any of the N_p valid positions. We assume that an image patch $\mathbf{x}_i$ is then conditionally generated from a multivariate Gaussian distribution

$$P(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_i; \alpha_i \mathbf{T}_{p_i} \boldsymbol{\mu}_{l_i} + \beta_i \mathbf{1}, c_i^2 \boldsymbol{\Sigma}_0)\,. \tag{3.1}$$

The label/position latent variable vector $\mathbf{z}_i = (l_i, x_i, y_i)$ controls the Gaussian mean via $\boldsymbol{\nu}_{\mathbf{z}_i} = \mathbf{T}_{p_i} \boldsymbol{\mu}_{l_i}$. Here $\mathbf{T}_{p_i}$ is a $d \times D$ projection matrix of zeros and ones which crops the sub-patch at position $p_i = (x_i, y_i)$ of a mini-epitome. The scalars α_i and β_i determine an affine mapping on the appearance vector and account for some photometric variability and $\mathbf{1}$ is the all-ones $d \times 1$ vector. Here $\bar{x}$ denotes the patch mean value and λ is a small regularization constant (we use $\lambda = d$ for image values between 0 and 255). In the experiments reported in this chapter we choose $\pi_k = 1/K$ and fix the $d \times d$ covariance matrix $\boldsymbol{\Sigma}_0^{-1} = \mathbf{D}^T\mathbf{D} + \epsilon\mathbf{I}$, where $\mathbf{D}$ is the gradient operator computing

the $x-$ and $y-$ derivatives of the $h \times w$ patch and ϵ is a small constant. Importantly, we assume that $\mathbf{\Sigma}_0$ is modulated by the patch gradient contrast $c_i^2 \triangleq \|\mathbf{D}(\mathbf{x}_i - \bar{x}_i \mathbf{1})\|_2^2 + \lambda$ but is shared across all dictionary elements and thus does not depend on the latent variable vector. This choice implies that we compute distances between patches by a Mahalanobis metric and corresponds to whitening the vectorized image patches by left-multiplying them with $\mathbf{D}$. We present algorithms for learning the epitomic means $\{\boldsymbol{\mu}_k\}_{k=1}^K$ in Sec. 3.1.4.

3.1.2 Epitomic patch matching

To match a patch $\mathbf{x}_i$ to the dictionary, we seek the mini-epitome label and position $\mathbf{z}_i = (l_i, x_i, y_i)$, as well as the photometric correction parameters (α_i, β_i) that maximize the probability in Eq. (3.1), or equivalently minimize the squared reconstruction error (note that $\mathbf{D1} = \mathbf{0}$)

$$R^2(\mathbf{x}_i; k, p) = \frac{1}{c_i^2}\Big(\|\mathbf{D}\left(\mathbf{x}_i - \alpha_i \mathbf{T}_p \boldsymbol{\mu}_k\right)\|^2 + \lambda(|\alpha_i| - 1)^2\Big)\,, \tag{3.2}$$

where the last regularization term discourages matches between patches and mini-epitomes whose contrast widely differs. We can compute in closed form for each candidate match $\boldsymbol{\nu}_{\mathbf{z}_i} = \mathbf{T}_{p_i}\boldsymbol{\mu}_{l_i}$ in the dictionary the optimal $\hat{\beta}_i = \bar{x}_i - \hat{\alpha}_i \bar{\nu}_{\mathbf{z}_i}$ and $\hat{\alpha}_i = \frac{\tilde{\mathbf{x}}_i^T \tilde{\boldsymbol{\nu}}_{\mathbf{z}_i} \pm \lambda}{\tilde{\boldsymbol{\nu}}_{\mathbf{z}_i}^T \tilde{\boldsymbol{\nu}}_{\mathbf{z}_i} + \lambda}$, where $\tilde{\mathbf{x}}_i = \mathbf{D}\mathbf{x}_i$ and $\tilde{\boldsymbol{\nu}}_{\mathbf{z}_i} = \mathbf{D}\boldsymbol{\nu}_{\mathbf{z}_i}$ are the whitened patches. The sign in the nominator is positive if $\tilde{\mathbf{x}}_i^T \tilde{\boldsymbol{\nu}}_{\mathbf{z}_i} \geq 0$ and negative otherwise. Having computed the best photometric correction parameters, we can substitute back in Eq. (3.2) and evaluate the reconstruction error $R^2(\mathbf{x}_i; k, p)$.

Epitomic matching versus max-pooling Searching for the best match in the epitome resembles the max-pooling process in convolutional neural networks [JKR09]. However in these two models the roles of dictionary elements and image patches are reversed: In epitomic matching, each image patch is assigned to one dictionary element. On the other hand, in max-pooling each dictionary element (filter in the terminology of [JKR09]) looks for its best matching patch within a search window. Max-pooling thus typically assigns some image patches to multiple filters while other patches remain orphan. This subtle but crucial difference makes it difficult for max-pooling to be used as a basis for building whole image probabilistic models, as the probability of orphan image areas is not well defined. Contrary to that, mini-epitomes naturally lend themselves as building

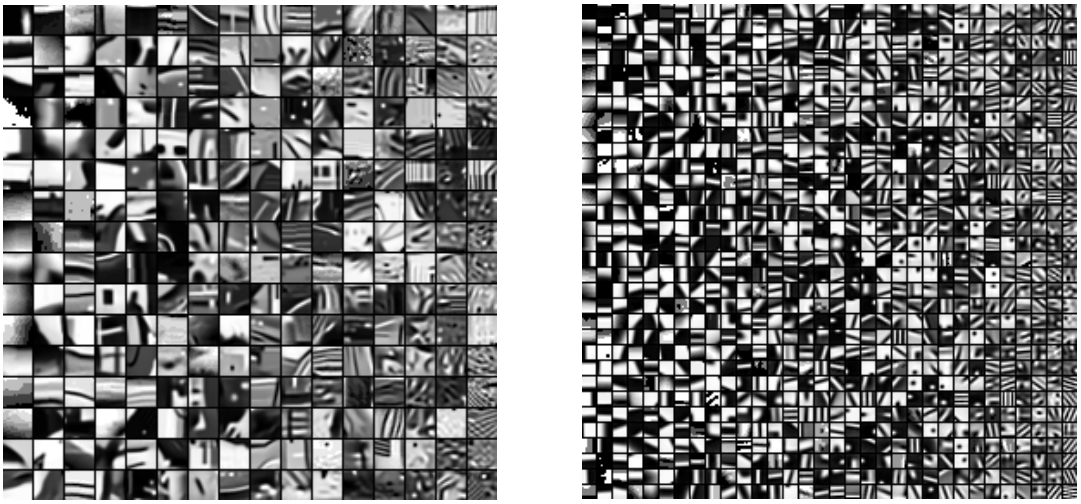
blocks for probabilistic image models able to explain and generate the whole image area.

3.1.3 Efficient epitomic search algorithms

We search over all mini-epitomes and positions in them to select the mini-epitome label and position pair (k, p) which achieves the least reconstruction error. The most expensive part of this matching process is computing the inner product of every patch in an image with all $h \times w$ sub-patches in every mini-epitome in the dictionary.

Exact search The complexity of the straightforward algorithm for matching N image patches to a dictionary with K mini-epitomes is $\mathcal{O}(N \cdot K \cdot h_p \cdot w_p \cdot h \cdot w)$. For the patch and epitome sizes we explore in our experiments, it takes more than 10 sec to exactly match each PASCAL VOC image with an optimized Matlab CPU implementation. Our optimized GPU software has drastically reduced computation time to 0.7 sec on a laptop's Nvidia GTX 650M graphics unit for $16\times16/8\times8$ epitomic dictionaries with K=256 elements. The starting point of our implementation was Krizhevsky's fast CUDA convolution library but we were able to further optimize the code by exploiting the fact that patches within a mini-epitome share filter values, which allowed us to make better use of the GPU's fast shared memory. Finally, we have also implemented the fast exact algorithm of [I 12], but it has proven less efficient than the GPU code for the range of epitome and patch sizes we have experimented with.

Approximate search We have also investigated the use of approximate nearest neighbor (ANN) methods for approximate epitomic patch matching. Contrary to exact search methods, ANN search time typically grows sub-linearly with the dictionary size, and is thus better scalable to extremely large dictionary sizes. The approach we have followed is to extract all patches from each mini-epitome along with their negated pairs, whiten, and then normalize them to be unit-norm vectors, resulting in an inflated epitomic dictionary with $N_p \cdot K \cdot 2$ elements. After similarly whitening and normalizing the input image patches, we search for their best match with standard off-the-shelf kd-tree and hierarchical kmeans algorithms as implemented in the FLANN library [ML09]. When using kd-trees, we have found it crucial to apply a rotation transformation based on the fast



(a) Our epitomic patch dictionary ($K = 256$) (b) Non-epitomic dictionary ($K = 1024$)

Figure 3.2: Patch dictionaries learned on the full VOC-07 training set, ordered column-wise from top-left by their relative frequency.

2-D discrete cosine transform (DCT), instead of searching directly for the best match in the image gradient domain. We also found that the performance loss due to ANN is negligible, for moderate search times comparable to those of SIFT-based VQ encoding algorithms.

3.1.4 Epitomic dictionary learning

Parameter refinement by Expectation-Maximization Given a large training set of unlabeled image patches $\{\mathbf{x}_i\}_{i=1}^N$, our goal is to learn the maximum likelihood model parameters $\boldsymbol{\theta} = (\{\pi_k, \boldsymbol{\mu}_k\}_{k=1}^K)$ for the epitomic GMM model in Eq. (3.1). As is standard with Gaussian mixture model learning, we employ the EM algorithm [DLR77] and maximize the expected complete log-likelihood

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{p\in\mathcal{P}}\gamma_i(k,p) \cdot \log\left(\pi_k\mathcal{N}(\mathbf{x}_i;\alpha_i\mathbf{T}_p\boldsymbol{\mu}_k + \beta_i 1, c_i^2\boldsymbol{\Sigma}_0)\right), \quad (3.3)$$

where $\mathcal{P}$ is the set of valid positions in the epitome. In the E-step, we compute the assignment of each patch to the dictionary, given the current model parameter values. We use the hard assignment version of EM and set $\gamma_i(k,p) = 1$ if the i-th patch best matches in the p-th position in the k-th

mini-epitome and 0 otherwise. In the M-step, we update each of the K mini-epitomes $\boldsymbol{\mu}_k$ by

$$\left(\sum_{i,p} \gamma_i(k,p) \frac{\alpha_i^2}{c_i^2} \mathbf{T}_p^T \boldsymbol{\Sigma}_0^{-1} \mathbf{T}_p \right) \boldsymbol{\mu}_k = \sum_{i,p} \gamma_i(k,p) \frac{\alpha_i}{c_i^2} \mathbf{T}_p^T \boldsymbol{\Sigma}_0^{-1} (\mathbf{x}_i - \bar{x}_i \mathbf{1}) . \quad (3.4)$$

In practice, it has been sufficient for all our experiments to run EM for 10 iterations.

Diverse dictionary initialization with epitomic K-means++ Careful parameter initialization can help EM converge faster and reach a good local optimum solution. For this purpose, we have adapted the K-means++ [AV07] algorithm to initialize our epitomic dictionary.

K-means++ selects the initial dictionary elements among the training set data instances. It randomly picks the first element and then incrementally grows the dictionary by selecting subsequent elements with probability proportional to their squared distance to the elements already in the dictionary. We adapt the standard K-means++ algorithm to our epitomic setup and select a $H \times W$ training image patch as a new mini-epitome with probability proportional to the sum of $R^2(\mathbf{x}_i; k, p)$ in a neighborhood of size $h_p \times w_p$ around the i-th patch. This corresponds to spatially smoothing the squared reconstruction error $R^2(\mathbf{x}_i; k, p)$ by a $h_p \times w_p$ box filter.

Learned epitomic dictionary We show in Fig. 3.2 the epitomic dictionary with $K = 256$ mini-epitomes we learned with the proposed algorithm on the full VOC-07 training set. We juxtapose it with the corresponding non-epitomic dictionary with $K = 1024$ members we learned with the same algorithm, simply setting $H = W = h = w = 8$. We have chosen the non-epitomic dictionary to have 4 times as many members so as both dictionaries occupy the same area (note that $16^2/8^2 = 4$) and thus be commensurate in the sense that they have equal number of parameters.

As expected, the non-epitomic dictionary looks very similar to the K-means patch dictionaries reported in [CLN11]. Our epitomic dictionary looks qualitatively different: It is more diverse and contains a rich set of visual patterns, including sharp edges, lines, corners, junctions, and sinewaves. It has less spatial redundancy than its non-epitomic counterpart, which needs to encode shifted versions of the same pattern as distinct codewords.

3.1.5 Reconstructing patches and images

Beyond qualitative comparisons, we have tried to systematically evaluate the generative expressive power of our epitomic dictionary compared to the non-epitomic baseline.

For this purpose, having trained the two dictionaries on the PASCAL VOC-07 train set, we have quantified how accurately they perform in reconstructing the images in the full VOC-07 test set. From each test image, we extract its 8×8 overlapping patches (with stride 2 pixels in each direction) that form the set of ground truth patches $\{\mathbf{x}_i\}_{i=1}^N$. For each patch $\mathbf{x}_i$ we compute its closest match $\hat{\mathbf{x}}_i = (\alpha_i \mathbf{T}_p \boldsymbol{\mu}_k + \beta_i \mathbf{1})$ in each of the two dictionaries by finding the parameters (α_i, β_i) and (k, p) that minimize the squared reconstruction $R^2(\mathbf{x}_i; k, p)$ in Eq. (3.2) – note that $p = (0, 0)$ in the non-epitomic case.

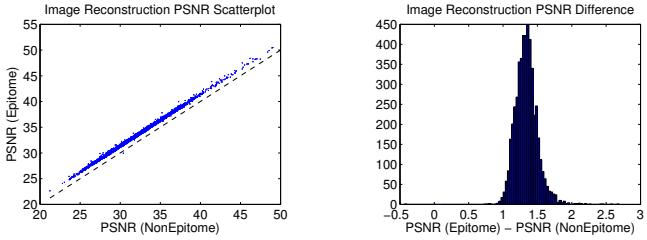
We quantify how close $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$ are in terms of normalized cross-correlation in both the raw intensity and gradient domains, $\mathrm{NCC}(i) = \frac{(\mathbf{x}_i - \bar{x}_i)^T (\hat{\mathbf{x}}_i - \bar{x}_i) + \lambda}{\|\mathbf{x}_i - \bar{x}_i\|_\lambda \|\hat{\mathbf{x}}_i - \bar{x}_i\|_\lambda}$ and $\mathrm{NCC}_D(i) = \frac{(\mathbf{x}_i - \bar{x}_i)^T \mathbf{D}^T \mathbf{D} (\hat{\mathbf{x}}_i - \bar{x}_i) + \lambda}{\|\mathbf{D}(\mathbf{x}_i - \bar{x}_i)\|_\lambda \|\mathbf{D}(\hat{\mathbf{x}}_i - \bar{x}_i)\|_\lambda}$ respectively, where $\|\mathbf{x}\|_\lambda \triangleq (\mathbf{x}^T \mathbf{x} + \lambda)^{1/2}$. Note that NCC takes values between 0 (poor match) and 1 (perfect match).

We can also reconstruct the original full-sized images by simply placing the reconstructed patches $\hat{\mathbf{x}}_i$ in their corresponding image positions and averaging at each pixel the values of all overlapping patches that contain it. We quantify the full image reconstruction quality in terms of PSNR. We show an example of an image reconstructed by this process in Fig. 3.3(a,b). Note that reconstructing an image from its SIFT descriptor [WJP11] is far less accurate and less straightforward than using a generative image model such as the proposed one.

To evaluate the reconstruction ability of each dictionary, we plot in Fig. 3.4 the empirical complementary cumulative distribution function (CCDF=1-CDF, where CDF is the cumulative distribution function) for the selected metrics. The plots summarize VOC-07 test set statistics of: (a/b) the NCC/ NCC_D for all $N \approx 5 \times 10^7$ patches and (c) the PSNR for all 4952 images. If $p = \mathrm{CCDF}(v)$, then $p \times 100\%$ of the samples in the dataset have values at least equal to v (higher CCDF curves are better).

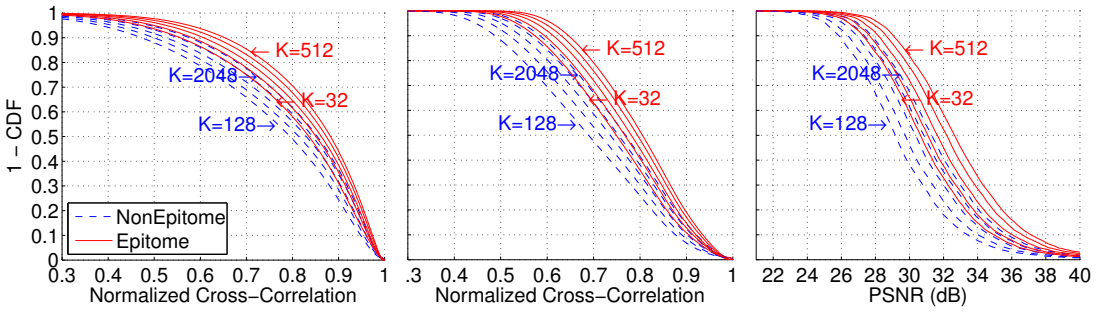
There are several observations we can make by inspecting Fig. 3.4. First, for either dictionary type, whenever we double the dictionary size K, the CCDF curves shift to the right/up by a rouphly

(a) Original image (b) Reconstructed (PSNR=29.2dB)



(c+d) Epitome *vs*. Non-epitome PSNR

Figure 3.3: (a,b) Image reconstruction example with the $K = 512$ epitomic dictionary. (c) Image reconstruction on VOC-07 test set: $K = 512$ epitomic *vs*. $K = 2048$ non-epitomic dictionaries.



(a) Raw image patches (b) Whitened image patches (c) Whole images

Figure 3.4: Image reconstruction evaluation on the full VOC-07 test set with our epitomic patch dictionary *vs*. a non-epitomic dictionary for various dictionary sizes K (powers of 2). (a,b): Normalized cross-correlation of raw (NCC) and whitened (NCC_D) image patches. (c): PSNR of reconstructed whole images. Plots depict 1-CDF (higher is better).

constant step. For example, we can read from Fig. 3.4(a) that the $K = 32$ epitomic dictionary already suffices to explain 58% of the image patches with $\mathrm{NCC} \geq 0.8$. Each time we double K we explain 3% more image patches at this level, with the $K = 512$ epitomic dictionary being able to reconstruct 70% of the image patches at $\mathrm{NCC} \geq 0.8$. In comparison, the $K = 2048$ non-epitomic baseline can only reconstruct 62% of the image patches at the same accuracy level. This demonstrates how difficult it is to capture with very high precision the space of all image patches [LN11].

Second, comparing the performance of the two dictionary types, we observe that our epitomic model significantly improves over the non-epitomic baseline in terms of reconstruction accuracy. For example, we can see that the $K = 64$ epitomic dictionary is roughly as accurate as the $K = 2048$ non-epitomic dictionary which has 32 times more elements (the same holds for the $K = 32/1024$ dictionaries). Accounting for the fact that each 16×16 mini-epitome occupies 4 times larger area than each cluster center of the 8×8 non-epitomic dictionary, implies that the epitomic dictionary is $32/4 = 8$ times more compact (in terms of number of model parameters) than the non-epitomic baseline. We further show in Fig. 3.3(c) that the epitomic dictionary consistently performs better (except for 1 out of the 4952 test images) in terms of image reconstruction PSNR (1.34 dB on average).

3.2 Image Description and Classification with Mini-Epitome Dictionaries

3.2.1 Image classification tasks

Here we explore how the proposed dictionary of mini-epitomes can be used for describing the appearance of image patches in image classification tasks. We focus our evaluation on the challenging PASCAL VOC datasets [EVW10]. We consider the standard PASCAL VOC-07 image classification benchmark.

We extract histogram-type features from both epitomic and non-epitomic patch representations which we feed to 1-vs-all SVM classifiers. We use χ^2 kernels approximated by explicit feature maps [VZ12] and also employ spatial pyramid matching [LSP06]. Our implementation closely

follows the publicly available setup of [CLV11], which presents a systematic evaluation and tuned implementation of SIFT features coupled with state-of-the-art encoding techniques.

3.2.2 Image description with mini-epitomes

Here we focus on extracting histogram type descriptors treating our epitomic dictionary as a bag of visual words. From each image, we densely extract $h \times w$ overlapping patches $\{\mathbf{x}_i\}_{i=1}^{N}$ (with stride 2 pixels in each direction). Matching each patch $\mathbf{x}_i$ to the epitomic dictionary yields its closest $h \times w$ patch in the epitomic dictionary, encoded by the epitomic label $l_i \in 1 : K$ and the position $p_i = (x_i, y_i)$, with $x_i = 0 : w_p - 1$ and $y_i = 0 : h_p - 1$. We use hard assignments (VQ) in all reported results.

In this setting, the most straightforward way to summarize the content of an image is to build a histogram with K bins, each counting how many times the specific epitome has been activated. This "*Epitome-Pos-1x1*" descriptor is very compact but completely discards the exact position of the match within the epitome.

Our epitomic dictionary allows us to also encode the position information p_i into the descriptor. While some of the $H \times W$ mini-epitomes in our learned dictionary (see Fig. 3.2) are homogeneous, others contain $h \times w$ patches with visually diverse appearance. These patches cannot be discriminated based on their epitomic label alone. We can encode the exact position p_i of the match in the epitome by a product histogram with $N_p \cdot K$ bins, where $N_p = h_p \times w_p$. However this yields a rather large descriptor (note that $N_p = 81$ in our setting) which is very sensitive to the exact position. We opt instead to summarize the position p_i using a coarse spatial grid. Specifically, for the experiments reported here we summarize the match positions in a $t \times t$ spatial grid of bins yielding a "*Epitome-Pos-txt*" descriptor with total length $t \cdot t \cdot K$. For example, in the *Epitome-16/8-Pos-4x4* descriptor the epitomic position bins have size 3×3 pixels and stride 2 pixels in each direction. The (b_x, b_y) bin $(b_x, b_y = 0 : 3)$ gets a vote for each matched patch whose position $p_i = (x_i, y_i)$ satisfies $2b_x \leq x_i < 2b_x + 3$ and $2b_y \leq y_i < 2b_y + 3$.

In all reported experiments we also encode the sign of the match, putting matches with positive and negative α_i's in different bins, which we have found to considerably improve performance at

the cost of doubling the descriptor size.

3.2.3 Classification results

For all the results involving the epitomic as well as the non-epitomic patch models, we have learned dictionaries of various sizes on the full VOC-07 train set. We summarize our results in Table 3.1 and illustrate them with plots in Fig. 3.5.

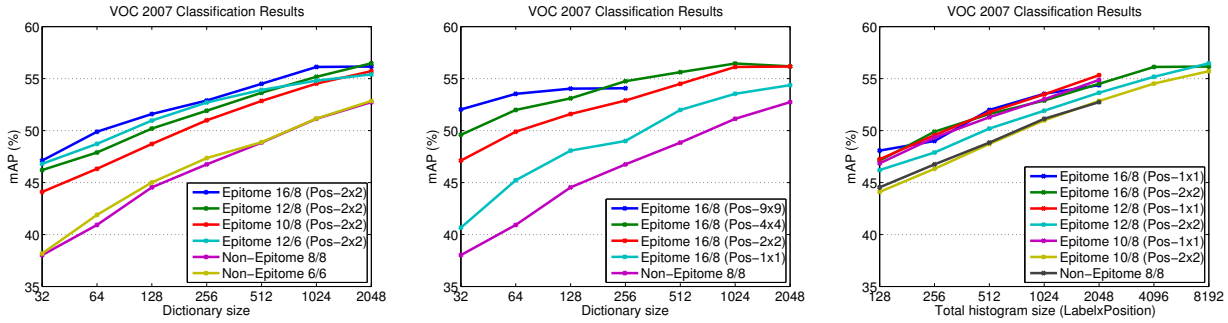
We first explore in Fig. 3.5(a) how epitome and patch sizes as well as dictionary sizes affect performance. We find that the exact setting for the epitome/patch size does not matter much. Similarly to the findings of [CLN11], we observe that the performance of all descriptors increases when we use dictionaries with more elements.

In Fig. 3.5(b) we show that position encoding considerably improves recognition performance with the coarse 2×2 scheme exhibiting an excellent trade-off between performance and descriptor size.

In all our experiments the proposed epitomic dictionaries significantly outperform the non-epitomic baseline, both for fixed dictionary size K, Fig. 3.5(a,b) and for fixed descriptor length Fig. 3.5(c).

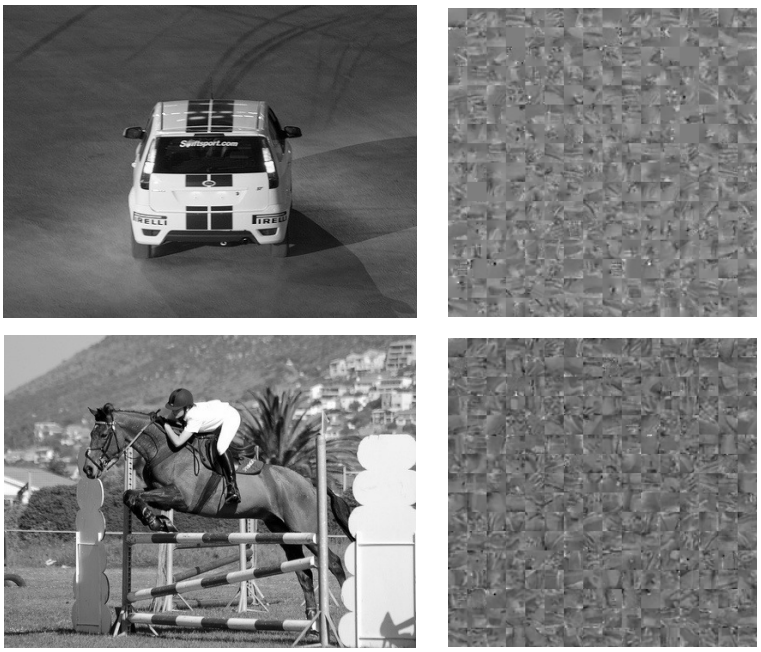
Comparing with the performance of VQ descriptors based on SIFT, see Table 3.2, the most impressive finding is that epitomic descriptors built on as few as $K = 256$ or 512 dictionaries yield performance around 55% mAP, which takes SIFT dictionaries of size 10K to achieve. Our best result at 56.5% mAP with 1024 mini-epitomes even slightly outperforms the best SIFT VQ result reported in [CLV11], attained with a dictionary of 25K visual words. This result is also comparable to KCB and LLC-based methods for encoding SIFT but still lags behind the Fisher Vector descriptor which holds the state-of-the-art in this task at about 61% mAP [SPM13, CLV11]. It will be interesting to adapt such powerful encoding methods in the epitomic setting.

Epitomic footprint encoding We have also explored an epitomic footprint encoding, which is related to the mean-vector Fisher Vector encoding in [SPM13]. The main idea is to encode the difference between the appearance content of a specific image compared to the generic epitome,



(a) (b) (c)

Figure 3.5: (a) Performance of the epitomic dictionary model *vs*. the non-epitomic baseline for different epitome/patch sizes. (b) Effect of encoding the epitome position at different detail levels. (c) Comparison of the epitomic model (with or without position encoding) and the non-epitomic model at the same total histogram length.



(a) Image (b) Epitomic footprint

Figure 3.6: Epitomic footprint descriptors of images.

Epitome	Position	Dictionary Size K						
/Patch	Encod.	32	64	128	256	512	1024	2048
16/8	1x1	40.66	45.22	48.07	49.00	51.98	53.54	54.37
	2x2	47.11	49.89	51.59	52.89	54.50	56.12	56.16
	4x4	49.59	51.98	53.10	54.75	55.62	**56.45**	56.18
	9x9	52.03	53.53	54.03	54.07	-	-	-
12/8	1x1	41.01	44.94	47.24	49.56	51.76	53.48	55.33
	2x2	46.20	47.89	50.19	51.91	53.64	55.17	**56.47**
10/8	1x1	41.12	44.07	46.85	49.33	51.28	53.01	54.87
	2x2	44.10	46.32	48.71	50.98	52.85	54.52	55.71
12/6	1x1	40.69	43.83	46.55	49.73	51.05	52.37	54.24
	2x2	46.80	48.72	50.96	52.70	53.91	54.80	55.40
	3x3	48.43	50.40	52.17	53.45	55.16	55.11	55.47
8/8	1x1	38.02	40.92	44.54	46.75	48.84	51.13	52.73
6/6	1x1	38.17	41.89	45.01	47.35	48.88	51.15	52.85

Table 3.1: Image classification results (mAP) of our epitomic dictionary on the Pascal VOC 2007 dataset.

which captures how much the epitome needs to adapt to best approximate a novel image. An appealling property of the epitomic footprint descriptor is that it can be visualized or stored as a small image and at the same time be used directly as feature vector in a linear SVM image classifier, yielding performance around 52% mAP in our experiments. Please see Fig.3.6 for a visualization.

3.3 Conclusion

We have shown that explicitly accounting for illumination and position variability can significantly improve both reconstruction and classification performance of a patch-based image dictionary. Moreover, we have demonstrated that the proposed epitomic dictionary model can perform similarly to SIFT in image classification tasks, implying that generative patch image models can be competitive with discriminative descriptors when properly accounting for nuisance factors.

Method	mAP	Method	mAP	Method	mAP
VQ-4K	53.42	KCB-4K	54.60	LLC-4K	53.79
VQ-10K	54.98	KCB-25K	56.26	LLC-10K	56.01
VQ-25K	56.07	FV-256	61.69	LLC-25K	57.60

Table 3.2: Image classification results (mAP) of top-performing SIFT-based methods on the Pascal VOC 2007 dataset [CLV11].

We can think of the proposed model as a "shallow network" in the terminology of deep learning. Recently, Papandreou *et al.* [PKS14] have extended this model to be deep and have demonstrated excellent performance on the challenging ImageNet [DDS09] image classification benchmark.

CHAPTER 4

Automatic Image Labeling from Weak 3D Supervision

Over the past few years, we have witnessed immense progress in solving visual recognition tasks. This is due to the availability of new sources of labeled data as well as the development of richer, holistic representations that combine multiple tasks [FMY13, KXS13, YFU12]. Most recent datasets provide annotations for multiple recognition tasks. For example, PASCAL VOC [EVW10] provides classification, detection and semantic segmentation labels for a handful of objects. ImageNet [DDS09] provides large-scale image classification and object localization annotations. Acquiring ground-truth data for each of these recognition tasks is very expensive even when employing crowd-sourcing systems such as MTurk. Thus, recently a number of approaches have tackled semantic segmentation in scenarios where weak annotations such as image tags, 2D bounding boxes, or strokes, are available.

In this chapter, our aim is to leverage the already labeled tasks in order to automate labeling of the more time consuming ones. In particular, we show how to exploit annotated 3D bounding boxes (available *e.g.*, in KITTI [GLU12]) to perform accurate object segmentation *without* any user in the loop. This is in contrast to interactive segmentation techniques, where the user can iteratively correct mistakes by giving additional strokes/seeds. Towards this goal, we develop an approach that exploits 3D information in the form of stereo, noisy point clouds, 3D CAD models, as well as appearance models and topological constraints (Fig. 4.1).

We demonstrate the effectiveness of our approach in the context of the challenging KITTI autonomous driving dataset [GLU12], which has been annotated with 3D bounding boxes, but not segmentation. This dataset is particularly rich as it contains image pairs collected with a stereo rig as well as point clouds captured with a LIDAR. It is also very challenging, as both the objects present in the scene as well as the ego-car (where the sensors are mounted) are moving. This poses

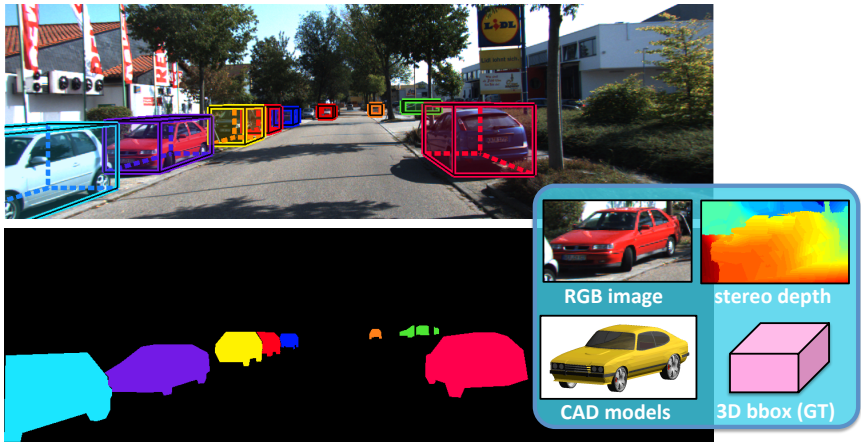
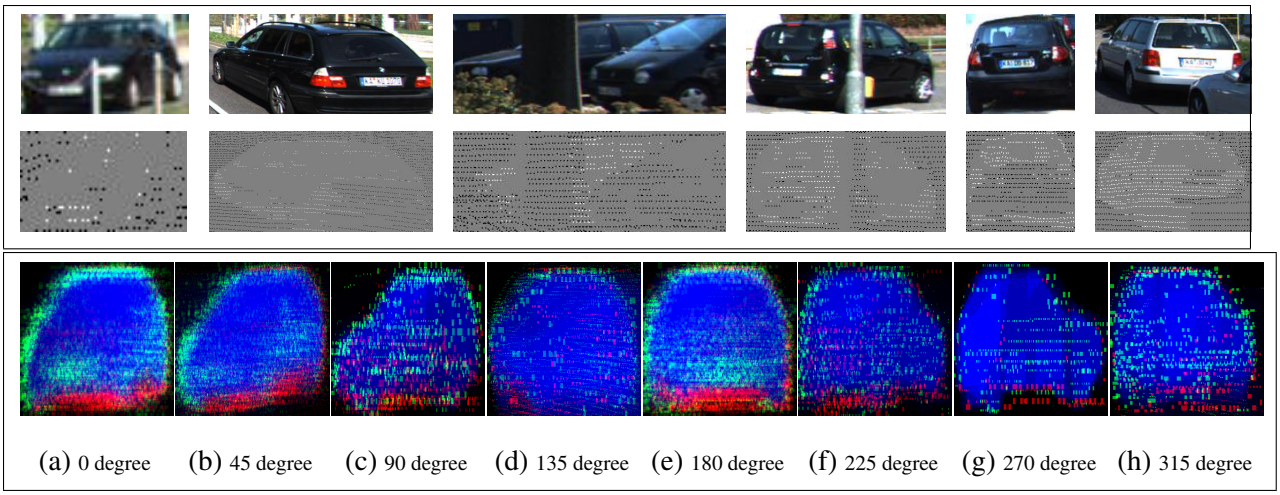


Figure 4.1: Our goal is to **automatically** generate segmentation ground-truth (bottom) using weak labels (top image).

many difficulties, some of which are illustrated in Fig. 4.2. First, the ground-truth 3D bounding boxes can be fairly noisy, as annotation in the presence of occlusion is very difficult. Non-reflective materials such as car windows and certain types of paint make the LIDAR point clouds very sparse. The point clouds also contain outliers, particularly near occlusion boundaries due to errors in registration. Furthermore, the objects in the scene move (fast) and the LIDAR does not perform instantaneous capture. The 3D information is also particularly ambiguous for the far-away objects as the density of the point clouds decreases with the distance to the sensor. Similarly, errors in the stereo estimation process also increase dramatically with the distance to the camera. Occlusion, low-resolution and saturated areas are other sources of ambiguities.

Our approach successfully deals with these difficulties and is able to perform as well as human annotators (i.e., MTurk) while being fully automatic. We conducted our experiments on a subset of KITTI [GLU12], which contains 950 cars with different scales, lighting/shading conditions, and occlusion levels. We reach a remarkable accuracy of 86% IOU in this challenging setting, outperforming GrabCut [RKB04] by 23%. Importantly, we require as little as 10 to 20 labeled objects to train the system. Furthermore, our approach can also be used to denoise MTurk annotations, improving by additional 3%. Our code and annotations are available at: https://bitbucket.org/liang_chieh_chen/segkitti.

Our work is similar to other techniques that tackle a variety of topics that we now briefly review.



(a) 0 degree (b) 45 degree (c) 90 degree (d) 135 degree (e) 180 degree (f) 225 degree (g) 270 degree (h) 315 degree

Figure 4.2: **Challenges of LIDAR: (Top box)** Points falling inside the ground-truth 3D boxes are white, and black outside. Gray pixels indicate there is no LIDAR observation. Points are sparse for small cars as well as large ones due to non-reflective materials (see first two examples). The projected points do not align well with image boundaries (see trunk and pole in the third and fourth examples). The GT 3D boxes are noisy, clipping car points or including background ones, *e.g.*, floor (see last two images). **(Bottom box)** Point clouds are averaged over the dataset for 8 different viewpoints. Red: car points that fall outside the foreground GT mask (false positives). Green: background points that fall within the GT mask (false negatives).

Interactive segmentation: In [BJ01], scribbles were used as seeds to model the appearance of foreground and background, and segmentation was performed via graph-cuts by combining appearance cues with a smoothness term [BK04]. GrabCut [RKB04] utilizes annotations in the form of 2D bounding boxes, and computes the foreground/background models using EM. [GF09] extended this idea to 3D by employing point clouds. 3D information as well as video has been used to ease the labeling task. Optical flow and structure from motion constraints are used in [XOT13] to propagate image labels in RGB-D videos. Our work bears similarity with [KGF12, XSU14] as we also exploit existing annotations for segmentation. However, while they utilize 2D boxes provided in ImageNet or tags in SIFT flow, we leverage 3D boxes and define a rich set of 2D and 3D potentials.

Point clouds: Point clouds have been exploited for semantic segmentation [BSF08]. [XQ09] enforced multiview consistency by feature tracking. In [XH10], neighboring points in a point cloud were grouped to form planar patches to which labels (walls, floors, ceilings, clutter) were assigned. [KAJ11] over-segmented the point clouds and modeled object co-occurrences and geometric arrangement. In [ZZY13], the physical stability of objects is employed when segmenting point clouds. [KXS13, LFU13, ZZY13] proposed to explore the compatibility between multiple segmentation hypotheses and 3D maps in order to perform 3D object detection. In contrast, we seek to exploit labeling compatibility between 2D image pixels and 3D point clouds. Besides, we focus on object segmentation, rather than object localization.

Shape priors and CAD models: Many approaches have incorporated shape priors into MRFs. [HPM04] combined MRFs and deformable models, [FZ05] utilize a template with a level-set formulation, and [YFU12] incorporate shape priors computed from deformable part-based models in holistic scene understanding. [VKR08] use DijkstraGC to impose object connectivity priors into segmentation. Objcut [KTZ10] employed learned object poses, while PoseCut [KRB08] exploits a human pose-specific shape prior. [LKR09] encodes tightness of the segmentation to the bounding boxes provided by users. The star shape prior of [Vek08] enforces connectivity along rays launched from a user-specified center. This was improved in [GRC10] by using multiple stars and Geodesic paths. The prevalent use of CAD models has been to augment the training data with synthetically generated views or to learn a geometric model from them [LS10, LPT13, ZSS13], for the task of object detection. In [SLH12], whole synthetic scenes are matched to real ones in order to transfer segmentation labels. Similarly, we fit CAD models to our examples and use the mask of the best fit as a potential.

4.1 Segmentation from Weakly Labeled Data

In this chapter, we are interested in performing automatic segmentation given annotated 3D bounding boxes, a collection of CAD models, LIDAR point clouds and/or stereo image pairs. We frame the problem as the one of figure/ground segmentation in a Markov Random Field (MRF), which

exploits appearance, smoothness, shape priors from CAD models and 3D information. To exploit 3D, we use two alternative depth sources: LIDAR, which provides us with sparse point clouds, and depth from stereo. We first describe how we compute depth in both settings. We then detail the energy of our MRF and discuss learning and inference.

4.1.1 Obtaining Dense Depth Maps

We employ PCBP [YMU13] to compute **stereo depth maps**, as it is the current state-of-the-art in KITTI. PCBP is a slanted plane MRF model with explicit reasoning about the type of boundary (coplanar, hinge, occlusion) between neighboring superpixels. This allows us to encode physical validity at junctions, resulting in better plane estimates.

As shown in Fig. 4.2, the **LIDAR point clouds** can be very sparse, even for nearby objects. To overcome this issue, we reconstruct a dense depth map from the sparse set of points. Let d_i be a continuous random variable encoding the depth of the i-th pixel, and let $\hat{d}_i$, with $i \in \mathcal{V}_D$, be the observed depth at pixel i, with $\mathcal{V}_D$ the sparse subset. We formulate the dense reconstruction as the one of inference in a continuous MRF. Specifically, the MRF encourages the reconstructed depth to be close to the observed depth measurements as well as the depth of neighboring pixels to be smooth. We define the energy as follows:

$$E(\mathbf{d}|I, \hat{\mathbf{d}}) = \sum_{i \in \mathcal{V}_D} \|d_i - \hat{d}_i\|_p^p + \sum_{(i,j) \in \mathcal{E}} w_{ij} \|d_i - d_j\|_p^p$$

with $\mathbf{d} = (d_1, \cdots, d_N)$ the set of variables encoding depth for all pixels, $\hat{\mathbf{d}}$ the set of observed depth values, I the color image and $\mathcal{E}$ the set of 4-connected neighbors. We use $w_{ij} = \lambda \cdot \exp(-\frac{\|I_i - I_j\|_2^2}{\beta})$, with λ the relative weight between the first and the second term, and β a scalar. Note that this energy generalizes [DT06] to p-norms: When $p = 2$ we obtain a Gaussian MRF, and when $p = 1$ a Laplacian MRF. Exact inference can be performed in both cases [SCN07].

4.1.2 Semantic Segmentation using 3D Data

Our setting is the following: we assume we are given an example annotated with a 3D bounding box. Our goal is to produce a figure-ground segmentation of the depicted object. Note that the

image within the projected box can in fact contain two objects due to occlusion. Our evaluation measures instance-level overlap, and thus our goal is to only label the correct object corresponding to the given 3D box.

Let $y_i \in \{0, 1\}$ be a random variable encoding whether the i-th pixel is background or foreground, respectively. We now describe the potentials in our segmentation MRF.

Appearance Model: Following [RKB04], we utilize two Gaussian Mixture Models (GMMs) to model the appearance of foreground/background. However, instead of using a human in the loop to define the seeds, we employ the available 3D information. We define the foreground seeds $\mathcal{F}$ to be the set of pixels inside the 2D bounding box which, when projected to 3D (using depth from either LIDAR or stereo), lie inside the 3D box (white pixels in Fig. 4.2). To compute the background seeds $\mathcal{B}$, we take the remaining pixels in the 2D box (black pixels in Fig. 4.2) as well as a 2-pixel band around the box. To make this process more robust, we compute a 2D convex hull around $\mathcal{F}$. We then remove all background seeds that fall inside this hull and have larger depth values than the mean depth value of the foreground seeds. These outliers are usually caused by the laser rays passing through the car windows. This simple procedure led to a slight improvement in performance. Then

$$\phi_i^{app}(y_i) = \begin{cases} -\log Pr(I_i|\theta_F^I) & \text{if } y_i = 1 \\ -\log Pr(I_i|\theta_B^I) & \text{if } y_i = 0 \end{cases}$$

where I_i is the color of pixel i, and θ_F^I and θ_B^I are the color appearance models for the foreground and background.

Smoothness: We employed an Ising prior

$$\phi_{ij}^{ising}(y_i, y_j) = \mathbb{1}_{(y_i \neq y_j)}$$

as well as a contrast sensitive Potts model

$$\phi_{ij}^{cs}(y_i, y_j) = \exp\{-\frac{\|I_i - I_j\|_2^2}{\beta}\} \cdot \mathbb{1}_{(y_i \neq y_j)}$$

where $\beta = \mathbb{E}(2 \cdot \|I_i - I_j\|_2^2)$ and $\mathbb{E}(\cdot)$ denotes the expectation over an image sample [BJ01].

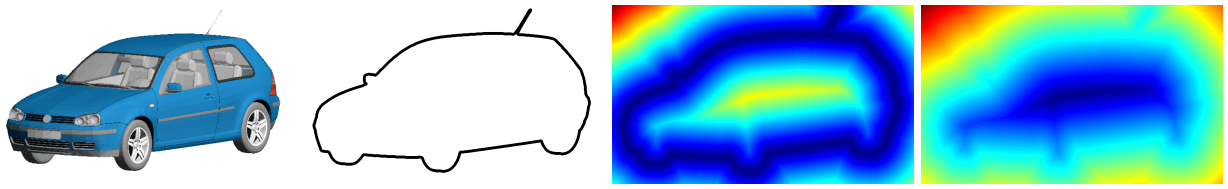
Topological Information: We modify the generic shape prior of [GRC10, Vek08], which enforces the connectivity of segmentation pixels along the rays originating from a point (*i.e.*, the star), to be asymmetric as follows

$$\phi_{ij}^{topo}(y_i, y_j) = \mathbb{1}(y_i = 1, y_j = 0)$$

for pixels on the ray going from j to i. Unlike [GRC10, Vek08], we automatically obtain the stars via K-means on the set of foreground seeds $\mathcal{F}$. Furthermore, we learn the importance of this potential instead of assigning it an infinite weight.

Leveraging CAD models: We use a collection of 180 car CAD models collected in [FDU12]. The idea is to find, for each 3D KITTI object, a CAD model that best fits this example and use the resulting CAD's shape as a prior for segmentation. Towards this goal, we first solve for the best 3D transformation that aligns each CAD model with each input 3D bounding box in terms of vertex error in 3D. This can be done in closed form via Procrustes analysis. The next step is to select the model which best represents each example. We chose the model that minimizes an error function consisting of 3 terms: 3D and 2D vertex error and real-world dimensions error. To compute the 2D error we project the CAD's 3D box to the image, fit a 2D box around it and compute the L2 distance between its vertices and the vertices of the input 2D box. To compute the last error term, we use the real-world dimensions of each CAD model. This captures, for example, that a VW Golf is smaller than a Toyota Tundra. This cannot be obtained from the CAD models, as they have arbitrary scale. Instead, we obtain this information from the car manufacturers and other online resources, as each CAD model's meta-data contains information about the car brand and model [FDU12]. Real-world dimension error is then the sum of L1-distances between the real-world dims (length, height, width) of a CAD and the dims of the input 3D box. We weigh each error term differently and learn the weights by cross-validation on the training set. The best CAD model is then projected to the image plane and its contour is extracted. We perform the signed distance transform [FH12] on the contour (the sign is negative within the car contour), and normalize it to be between -1 and 1 for every example in order to not bias large cars. Thus

$$\phi_i^{CAD}(y_i) = sdt(i|CAD) \cdot \mathbb{1}(y_i = 1)$$



(a) CAD shape (b) contour (c) dist. trans. (d) signed d.t.

Figure 4.3: (a) CAD model projected to image plane, (b) contour, (c,d) distance and signed distance transform.

where $sdt(i|CAD)$ is the value of pixel i after signed distance transform. This is illustrated in Fig. 4.3.

Depth seeds: We penalize labelings that misclassify foreground and background seed points as follows

$$\phi_i^{depth,f}(y_i) = \mathbb{1}(y_i = 0, i \in \mathcal{F})$$
$$\phi_i^{depth,b}(y_i) = \mathbb{1}(y_i = 1, i \in \mathcal{B})$$

with $\mathcal{F}, \mathcal{B}$ the set of foreground and background seeds.

Dense depth: We learn two GMMs to represent the histogram of depth for the background and foreground. Thus

$$\phi_i^{depth}(y_i) = \begin{cases} -\log Pr(d_i|\theta_F^S) & \text{if } y_i = 1 \\ -\log Pr(d_i|\theta_B^S) & \text{if } y_i = 0 \end{cases}$$

with d_i is the dense reconstructed depth (either from stereo or the MRF defined in section 4.1.1), and θ_F^S and θ_B^S are the depth appearance models for foreground and background.

4.1.3 Learning and Inference

We use a log-linear model

$$E(\boldsymbol{y}|I, \mathbf{d}) = \boldsymbol{w}^T \cdot \Psi(\boldsymbol{y}|I, \mathbf{d})$$

where $\Psi(\boldsymbol{y}|I, \mathbf{d})$ is the concatenation of all potentials summed across cliques (as we share the weights between them). This results in $\boldsymbol{w} \in \mathbb{R}^8$, with one weight for appearance, two for smoothness, two for discrete depth, one for continuous depth, one for the topology and one for CAD.

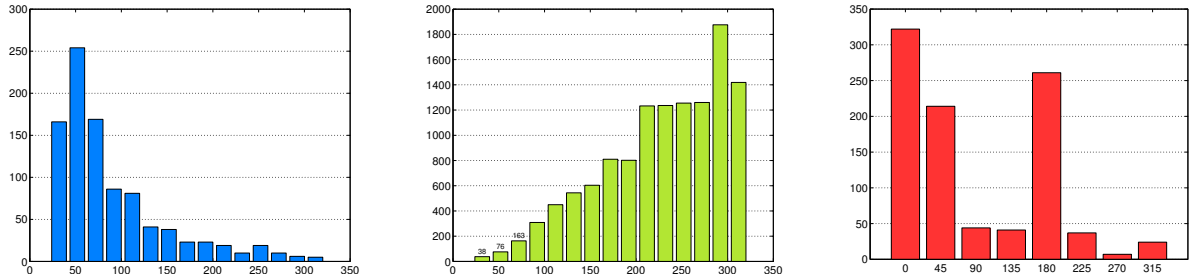
Inference in our model can be done exactly via graph-cuts, as we have a binary MRF with sub-modular potentials. We use the graph-cut implementation of [BK04]. We employ structural SVMs [TGK04, TJH05] to learn the parameters of the model, and use the parallel cutting plane algorithm of [SFP13]. As loss we use Hamming distance, which counts the percentage of pixels that are wrongly labeled. This loss decomposes into unary potentials, and thus the loss-augmented inference can be solved exactly via graph-cuts.

4.2 Experimental Evaluation

We selected a random subset of images from the KITTI `raw data` [GLU12], having a total of 950 cars. We asked 9 in-house annotators to provide very high-quality segmentations. It took on average 60 seconds to label each car, and 16h to label the full dataset, where each image is labeled by a single annotator. As shown in Fig. 4.4 (left), our dataset contains cars at very different resolutions. The projection of the 3D bounding boxes into the image ranges from 20×24 to 279×372 pixels. In order to maintain the same distribution of scales, we build a 30 bin histogram of object sizes and select randomly within each bin $40\%, 10\%$, and 50% of cars as training, validation, and test set, resulting in 382 examples for training, 100 for validation, and 468 for testing. Following PASCAL VOC we utilize intersection-over-union (IOU) as our evaluation metric.

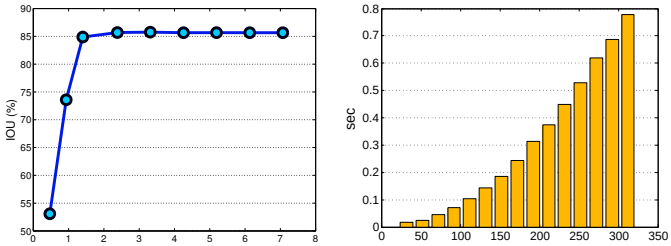
The 3D LIDAR point clouds are very sparse. As shown in Fig. 4.4 (center), small cars contain only 38.2 points on average, which is 3.7% of the 2D bounding box. This makes segmenting small cars very difficult. As expected, the number of points increases with scale. However, for the largest scale (the last bin), we find that there is one car with very few points due to its metallic paint. This biases the statistics as there are not that many examples of this size.

KITTI was collected with sensors (*i.e.*, cameras and LIDAR) mounted on top of a driving car. As a consequence, most of the cars in the images are oriented with 0, 45 and 180 degrees (driving towards or away from the ego-car). Fig. 4.4 (right) shows the distribution of car orientations (at



(BBox size) (LIDAR point clouds) (Orientation histogram)

Figure 4.4: **Dataset statistics:** **(Left)** Histogram of car scales in terms of bounding box size, defined as the square root of the bounding box area. **(Center)** Average number of LIDAR points for different scales. **(Right)** Histogram of car orientations.

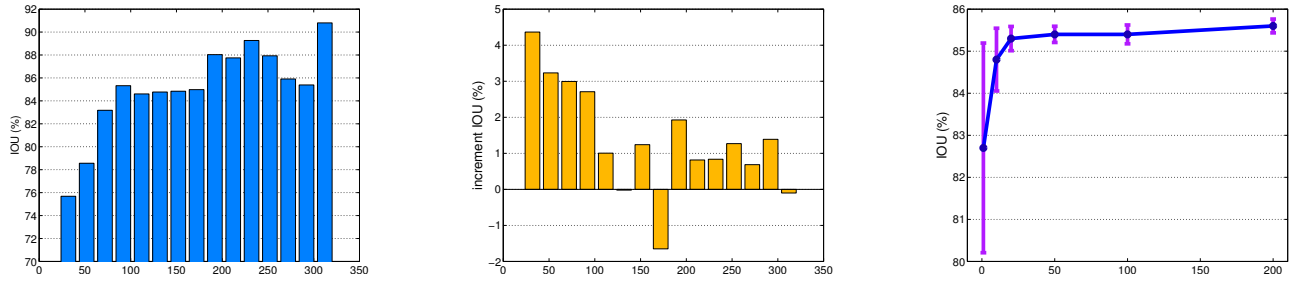


(Training time in minutes) (Avg. inference time)

Figure 4.5: **Time: (Left)** IOU as a function of training time, **(Right)** average inference time as a function of scale.

0° the car is facing south, and west at 90°). To further analyze the point clouds, we overlay their projection with our ground-truth segmentations and average them for 8 different azimuth angles, after resizing to a common scale to compensate for the size difference among examples. As shown in the bottom row of Fig. 4.2, most of the mistakes (green and red) are along the car boundaries due to registration errors and the fact that objects and the ego-car are moving. Besides, the noise introduced when labeling 3D bounding boxes causes mistakes on the car and near the ground plane.

To benchmark and compare crowd-sourcing annotations, we collected segmentations via MTurk in three batches. For the first two batches our quality requirements were lower, requiring MTurkers to have at least 75% approval rate. We paid 1 cent per car. For the third batch we required 95% approval rate and 500 approved tasks, and paid them 5 cents per car. We asked the MTurkers to draw the outer boundary of the car within the marked box as accurately as possible. For images



(LIDAR) (Improvement LIDAR wrt. stereo) (Training set size)

Figure 4.6: **(Left)** IOU as a function of bounding box size when using LIDAR. **(Center)** IOU difference when employing our full model with LIDAR vs Stereo as a function of size. **(Right)** IOU as a function of number of training examples. With only 10-20 train-val images, our model can reach performance similar to MTurk. Results are averaged across 5 partitions.

with low contrast we asked them to make an educated guess. If a car was occluded (by *e.g.*, a tree), they were asked to draw a boundary *also* around each occluding region. If the car was occluded by another car, they were asked to mark the other car as an occluder. We showed three examples of good annotations, a fully visible car, a low contrast example, and a partially occluded car.

Comparison to state-of-the-art: Table 4.1 compares our approach to a set of GrabCuts based baselines, which, like our approach, do not use a user in the loop. The first baseline (A) projects the 3D bounding box onto the image, and uses the pixels inside as the foreground seeds and a band outside as background seeds. The second baseline (B) utilizes only pixels inside 1/4 of the box around the center as seeds. The last baseline (C), which is an instance of our model, employs the projected LIDAR points as seeds for foreground or background, depending on whether they are inside or outside the 3D bounding box. Our approach outperforms the baselines by more than 20% in terms of IOU when employing LIDAR and/or stereo. Moreover, utilizing the point clouds as seeds outperforms the other heuristics.

Size matters: As shown in Fig. 4.6 (left) segmenting big cars is easier than smaller ones.

Method	Baselines			Our Model		
	[RKB04] A	[RKB04] B	[RKB04] C	Stereo	LIDAR	Hybrid
IOU (%)	52.9	59.1	62.1	**84.4**	**85.6**	**85.9**

Table 4.1: **Comparison to GrabCuts:** Baseline A uses all pixels inside the 2D bounding box as foreground seeds, Baseline B uses pixels inside 1/4 of the 2D box as foreground seeds and baseline C uses projected LIDAR points as seeds. By hybrid we mean an approach that uses dense depth from stereo and depth seeds from LIDAR.

Importance of the features: As shown in Table 4.2, each potential increases performance. The CAD model is the potential that contributes the most.

Stereo vs LIDAR: As shown in Tables 4.1 and 4.2, we can reach almost the same performance with stereo or LIDAR. This is great news for autonomous driving as cameras are much cheaper sensors. The highest accuracy can be obtained when we use a hybrid approach, which takes advantage of both stereo and LIDAR. Fig. 4.6 (center) shows that LIDAR is particularly beneficial wrt. stereo for small cars.

Time: Fig. 4.5 shows the training and average inference time as a function of size. Learning converges within 2 minutes. The parallel cutting plane algorithm of [SFP13] makes the training time 3.3 times faster than the original algorithm (when using four threads). The average inference time is proportional to the car scale. Inference for the full test set takes 44 seconds using a single core.

Automatic vs MTurk: As shown in Table 4.3, our approach performs as well as MTurker, while being fully automatic. We can also **de-noise** MTurk annotations by using their segmentations as an additional potential in the model. The potential that we use is similar to the one for the CAD models, where instead of the CAD mask we use the MTurk segmentation. This pushes the performance even higher. Such a setting can for example be used to reduce the number of annotators per image. "Oracle" here means a scenario in which we can choose the best annotation

Model	Appearance	Smooth	Depth seeds	Dense depth	Topo	CAD	Stereo IOU (%)	LIDAR IOU (%)
Basic (no ising)	✓	✓					60.1	61.1
Basic	✓	✓					61.6	62.1
Topology	✓	✓			✓		64.4	64.2
Dense depth	✓	✓		✓			68.4	71.3
Discrete depth	✓	✓	✓				–	74.2
All depth	✓	✓	✓	✓			–	76.2
CAD	✓	✓				✓	82.9	83.5
Full Model w/o CAD	✓	✓	✓	✓	✓		70.2	77.6
Full Model w/o Dense depth	✓	✓	✓		✓	✓	83.4	84.1
Full Model w/o Depth Seeds	✓	✓		✓	✓	✓	–	85.4
Full Model w/o Topology	✓	✓	✓	✓		✓	84.2	85.5
Full Model	✓	✓	✓	✓	✓	✓	**84.4**	**85.6**

Table 4.2: **Importance of Features:** Every feature helps, and the CAD model potential helps the most. Notably, performance with stereo is similar to LIDAR. Note that when using stereo, we do not use Depth seeds.

within the three batches. Note that, even in this setting, we can de-noise the results using our model.

Training Set Size: We next investigate performance of our full model as a function of the number of training images. As shown in Fig. 4.6 (right), results similar to MTurk can be obtained when training with as little as 10-20 images.

Gaussian vs Laplacian MRF: Fig. 4.7 (left) shows performance of the basic model (appearance + smoothness) when enriching it with continuous depth as a function of the smoothness term in the dense reconstruction. Note that $p = 1$ (i.e., Laplacian MRF) is more robust to outliers, and a wide range of weights result in good performance.

Number of Stars: As shown in Fig. 4.7 (right) using a single star yields the best performance for a model containing appearance, smoothness and topology. This is expected as cars are a single connected component, with the exception of occlusion (*e.g.*, car behind a pole).

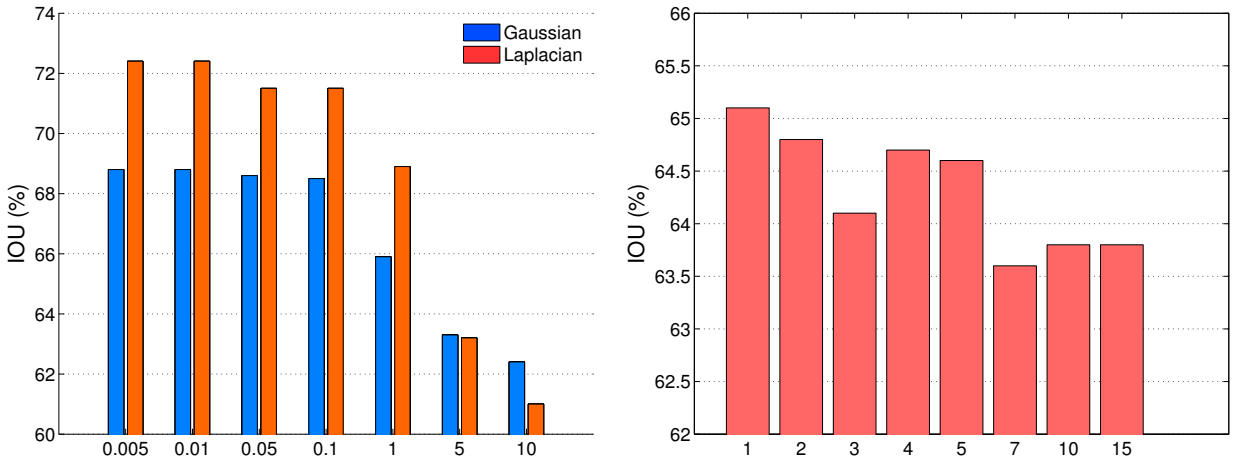
	MTurk accuracy	Ours with MTurk pot.
Batch 1	85.3	**87.1**
Batch 2	85.9	**88.3**
Batch 3	86.7	**87.6**
Batch 1,2,3	–	**88.9**
Oracle	90.2	**90.6**

Table 4.3: **Denoising MTurk:** At 85.9% our model's accuracy is comparable to MTurkers'. Our model is also able to de-noise MTurk annotations, further boosting performance.

Average shape: An alternative shape prior to CAD models is to cluster the data by orientation, and average the training segmentations for each bin (see Fig. 4.2). Adding this to the base model with appearance and smoothness results in 67.6% IOU, while using it in the full model results in 80.3%, *i.e.*, 5% worse than when using CAD.

Depth as a segmentation algorithm: We also look into whether the dense reconstructions alone can be used to perform segmentation. Towards this goal, we classify a point as car if it is inside the 3D bounding box, and background otherwise. Laplacian MRF performs best, with 76.2% IOU. The performance of the Gaussian MRF is 4% lower since it is less robust. The performance when using stereo is 69.8%, which is 15% worse than our full model. This shows that depth alone is not sufficient and that we greatly benefit from using multiple diverse cues.

Qualitative results: Fig. 4.8 depicts segmentation results when using our full model with stereo and LIDAR (4th and 6th columns respectively). Our model is robust to low-resolution imagery, saturation, noise, sparse point clouds, depth estimation errors and can successfully segment out occluders. The last three rows show failure modes. Our approach has difficulty with very small cars that are heavily occluded, and point clouds with a large number of outliers.



(Smoothness weight λ) (Number of stars)

Figure 4.7: **Importance of parameters: (Left)** smoothness weight for the continuous MRF, and **(Right)** the number of stars for the topology potential. Result are on validation set.

4.3 Conclusions

We have presented an approach that can generate segmentations of the same quality as human labelers (i.e., MTurkers) using only weak supervision in the form of 3D bounding boxes. Towards this goal, we have exploited appearance models, stereo and/or noisy point clouds, a repository of 3D CAD models as well as topological constraints.

Recently, Zhang *et al*. [ZSF15] have extended this work. They applied our model to generate thousands of car segmentations, and train a deep convolutional neural network with the generated annotations. They have shown outstanding performance on the task of car instance segmentation and depth ordering.

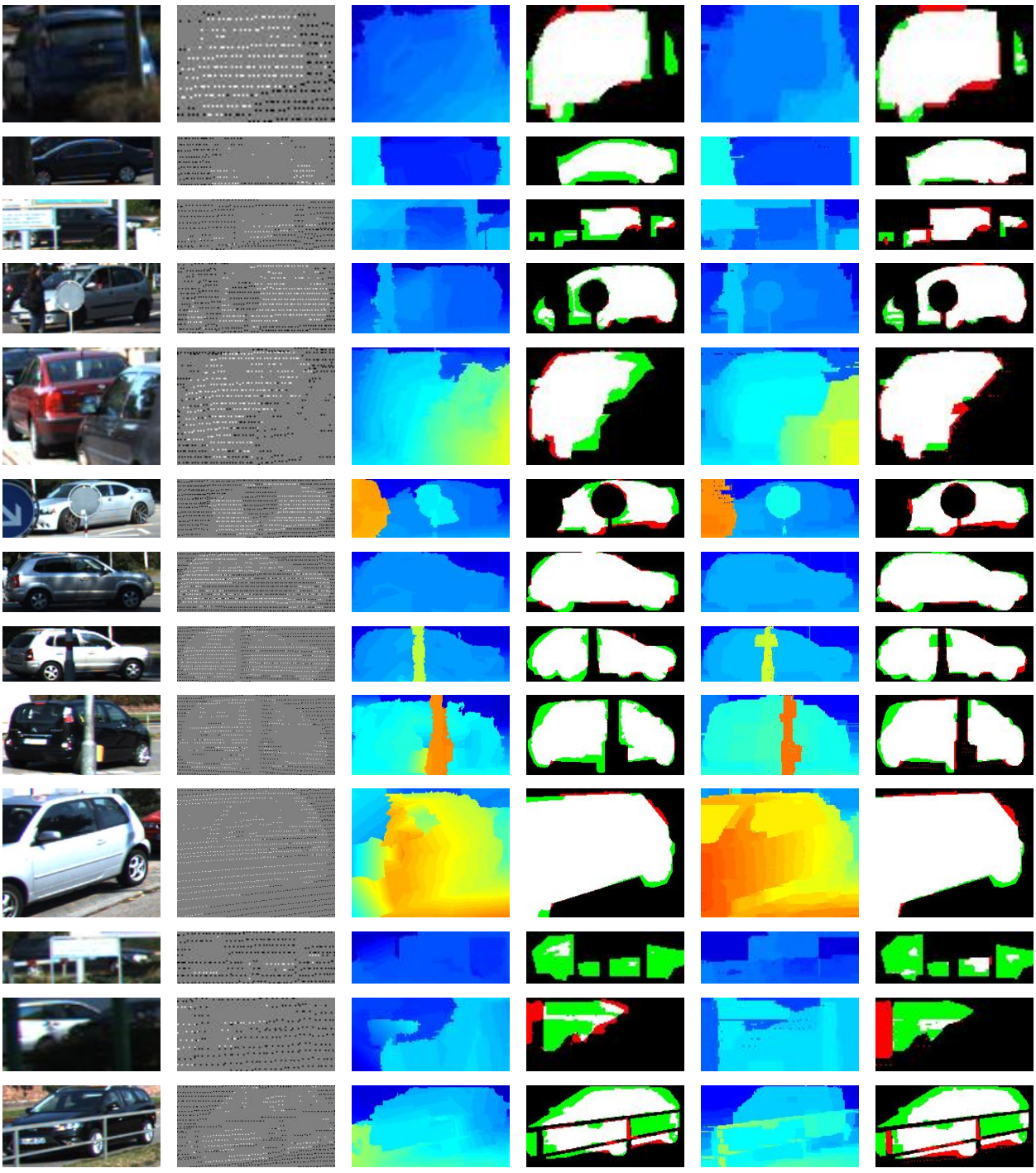


Figure 4.8: **Segmentation results:** Each row shows the image, LIDAR points (White: car, Black: bckgr.), stereo depth, results of our full model with stereo (White: True Positive, Black: True Negative, Red: False Positive, Green: False Negative), depth images reconstructed by Laplacian MRF, and results of our full model with LIDAR. Last three rows show our failure modes.

CHAPTER 5

Learning Deep Structured Models

Deep learning algorithms attempt to model high-level abstractions of the data using architectures composed of multiple non-linear transformations. A multiplicity of variants have been proposed [HSA84, LBB98, HS06, BLP07, SH12, ZF14] and shown to be extremely successful in a wide variety of applications including computer vision, speech recognition as well as natural language processing [LGR09, SHB12, JSD14, KSH13, ERF14]. Recently, state-of-the-art results have been achieved in many computer vision tasks, outperforming competitive methods by a large margin [KSH13, GDD14].

Deep neural networks can, however, be even more powerful when combined with graphical models in order to capture the statistical dependencies between the variables of interest. For example, [DDJ14] exploit mutual exclusion, overlapping and subsumption properties of class labels in order to better predict in large scale classification tasks. In pose estimation, more accurate predictions can be obtained when encoding the spatial relationships between joint locations [TJL14].

It is, however, an open problem how to develop scalable deep learning algorithms that can learn higher-order knowledge taking into account the output variables' dependencies. Existing approaches often rely on a two-step process [NRB11, XSU14] where a non-linear classifier that employs deep features is trained first, and its output is used to generate potentials for the structured predictor. This piece-wise training is, however, suboptimal as the deep features are learned while ignoring the dependencies between the variables of interest. For example, in object recognition, independently learned segmentation and detection features [HAG14b] might be focusing on predicting the same examples correctly, but when learned jointly, they can improve their predictive power by exploiting complementary information to fix additional mistakes.

In this chapter we extend deep learning algorithms to learn complex representations taking into

account the dependencies between the output random variables. Towards this goal, we propose a learning algorithm that is able to learn structured models with arbitrary graphs jointly with deep features that form potentials in a Markov random field (MRF). Our approach is efficient as it blends learning and inference, resulting in a single loop algorithm which makes use of GPU acceleration. We demonstrate the effectiveness of our method in the tasks of predicting words from noisy images, and tagging of Flickr photographs. We show that joint learning of deep features and MRF parameters results in big performance gains.

5.1 Learning Deep Structured Models

In this section we investigate how to learn deep features taking into account the dependencies between the output variables. Let $y \in \mathcal{Y}$ be the set of random variables $y = (y_1, \ldots, y_N)$ that we are interested in predicting. We assume the space of valid configurations to be a product space, *i.e.*, $\mathcal{Y} = \prod_{i=1}^{N} \mathcal{Y}_i$, and the domain of each individual variable y_i to be discrete, *i.e.*, $\mathcal{Y}_i = \{1, \ldots, |\mathcal{Y}_i|\}$. Given input data $x \in \mathcal{X}$ and parameters $w \in \mathbb{R}^A$ of the function $F(x, y; w) : \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^A \rightarrow \mathbb{R}$, inference amounts to finding the highest scoring configuration

$$y^* = \arg\max_y F(x, y; w).$$

Note that if F is a deep network, *i.e.*, a composite function, and there are no connections between the output variables to be predicted, inference corresponds to a forward pass to evaluate the function, followed by independently finding the largest response for each variable. This can be interpreted as inference in a graphical model with only unary potentials. However, for arbitrary graphical models it is NP-hard to find the maximizing configuration y^* since the inference program generally requires a search over a space of size $\prod_{i=1}^{N} |\mathcal{Y}_i|$. Note also that log-linear models are a special case of this program, with $F(x, y; w) = w^\top \phi(x, y)$ and $\phi(x, y)$ denoting a feature vector, computed using the input-output pair (x, y).

In this work, we consider the general setting where $F(x, y; w)$ is an arbitrary scalar-valued function of w and (x, y). In our experiments F is a function composition of non-linear base mappings such as convolutions, rectifications and pooling. We let the probability of an arbitrary con-

figuration $\hat{y}$ be given by the annealed soft-max

$$p_{(x,y)}(\hat{y}|w,\epsilon) = \frac{1}{Z_\epsilon(x,w)} \exp(F(x,\hat{y};w))^{1/\epsilon}.$$

Hereby $Z_\epsilon(x,w)$ refers to the partition function, normalizing the distribution $p_{(x,y)}$ to lie within the probability simplex Δ via $Z(x,w) = \sum_{\hat{y}\in\mathcal{Y}} \exp(F(x,\hat{y};w))^{1/\epsilon}$. The annealing parameter $\epsilon \geq 0$ is used to adjust the uniformity of the distribution. We consider general graphical models where the computation of $Z_\epsilon(x,w)$ is #P-hard.

5.1.1 Learning via gradient descent

During learning, given a training set $\mathcal{D}$ of input-output pairs $(x,y) \in \mathcal{D}$, we are interested in finding the parameters w of the model. We do so by maximizing the data likelihood, *i.e.*, minimizing the negative log-likelihood $-\ln \prod_{(x,y)\in\mathcal{D}} p_{(x,y)}(y|w,\epsilon)$ which yields

$$\min_w \sum_{(x,y)\in\mathcal{D}} \left(\epsilon \ln Z_\epsilon(x,w) - F(x,y;w)\right). \tag{5.1}$$

Note that this is equivalent to maximizing the cross-entropy between a target distribution $p_{(x,y),\text{tg}}(\hat{y}) = \delta(\hat{y} = y)$ placing all its mass on the groundtruth label, and the model distribution $p_{(x,y)}(\hat{y}|w,\epsilon)$. Hence Eq. (5.1) is equivalently obtained by $\max_w \sum_{(x,y),\hat{y}\in\mathcal{Y}} p_{(x,y),\text{tg}}(\hat{y}) \ln p_{(x,y)}(\hat{y}|w,\epsilon)$. It is easily possible to incorporate more general target distributions into Eq. (5.1). Note also that regularization can be included and $\epsilon \to 0$ recovers the general structured hinge loss objective $\min_w \sum_{(x,y)\in\mathcal{D}} \left(\max_{\hat{y}} F(x,\hat{y};w) - F(x,y;w)\right)$, since a margin term is easily incorporated.

Minimizing Eq. (5.1) w.r.t. w requires computation of the gradient $\frac{\partial}{\partial w} \sum_{(x,y)} -\ln p_{(x,y)}(y|w,\epsilon)$, which is given by a transformed difference between the distributions of the model $p_{(x,y)}(\hat{y}|w,\epsilon)$ and the target $p_{(x,y),\text{tg}}(\hat{y})$:

$$\sum_{(x,y)\in\mathcal{D}} \sum_{\hat{y}\in\mathcal{Y}} \frac{\partial}{\partial w} F(x,\hat{y};w) \left(p_{(x,y)}(\hat{y}|w,\epsilon) - p_{(x,y),\text{tg}}(\hat{y})\right). \tag{5.2}$$

A gradient descent algorithm for minimizing Eq. (5.1) will iterate between the following steps: (i) For a given w evaluate the function F, (ii) compute the model distribution $p_{(x,y)}(\hat{y}|w,\epsilon)$, (iii) propagate the difference between the model and target distribution using a backward pass (resembling the chain rule for composite functions) and (iv) update the parameters w. This is summarized in Fig. 5.1.

Algorithm: Deep Structured Learning

Repeat until stopping criteria

1. Forward pass to compute $F(x, \hat{y}; w)$
2. Obtain $p_{(x,y)}(\hat{y}|w, \epsilon)$ via a soft-max
3. Backward pass via chain rule to obtain gradient
4. Update parameters w

Figure 5.1: Gradient descent for learning deep structured models.

$$\min_{w} \sum_{(x,y)\in\mathcal{D}} \left(\max_{b_{(x,y)}\in\mathcal{C}_{(x,y)}} \left\{ \sum_{r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) f_r(x, \hat{y}_r; w) + \sum_{r} \epsilon c_r H(b_{(x,y),r}) \right\} - F(x, y; w) \right)$$

Figure 5.2: The approximated non-linear structured prediction task.

5.1.2 Approximate Learning

Note that for general graphical models the exact computation of $p_{(x,y)}(\hat{y}|w, \epsilon)$ is not possible since the state-space size $|\mathcal{Y}| = \prod_{i=1}^{N} |\mathcal{Y}_i|$ is exponential in the number of variables. As a consequence it is intractable to compute the exact gradient of the cost-function given in Eq. (5.2) and one has to resort to approximate solutions.

Inspired by approximations used for log-linear models, we make use of the following identity [WJ08, KF09]:

$$\epsilon \ln Z_\epsilon(x, w) = \max_{p_{(x,y)}(\hat{y})\in\Delta} \mathbb{E}[F(x, \hat{y}; w)] + \epsilon H(p_{(x,y)}), \tag{5.3}$$

where $\mathbb{E}$ denotes an expectation over the distribution $p_{(x,y)}(\hat{y})$ and H refers to its entropy.

For most applications, $F(x, y; w)$ decomposes into a sum of functions, each depending on a local subset of variables y_r, *i.e.*,

$$F(x, y; w) = \sum_{r\in\mathcal{R}} f_r(x, y_r; w).$$

Hereby r is a restriction of the variable tuple $y = (y_1, \ldots, y_N)$ to the subset $r \subseteq \{1, \ldots, N\}$, i.e., $y_r = (y_i)_{i \in r}$. All subsets r required to compute the model function F are summarized in the set $\mathcal{R}$. Importantly we note that each local composite function $f_r(x, y_r; w)$ can depend non-linearly on the parameters w.

Plugging this decomposition into Eq. (5.3), we equivalently get the log-partition function $\epsilon \ln Z_\epsilon(x, w)$ via

$$\max_{p_{(x,y)}(\hat{y}) \in \Delta} \sum_{r, \hat{y}_r} p_{(x,y),r}(\hat{y}_r) f_r(x, \hat{y}_r; w) + \epsilon H(p_{(x,y)}),$$

where we use marginals $p_{(x,y),r}(\hat{y}_r) = \sum_{y \setminus y_r} p_{(x,y)}(y)$.

Despite the assumed locality of the scoring function, the learning task remains computationally challenging since the entropy $H(p_{(x,y)})$ can only be computed exactly for a very small set of models, *e.g.*, models for which the dependencies of the joint distribution $p_{(x,y)}(y)$ are equivalently characterized by a low tree-width graph. In addition, the marginalization constraints are exponential in size.

To deal with both issues a common solution in log-linear models is to approximate the true marginals $p_{(x,y),r}$ with local beliefs $b_{(x,y),r}$ that are not required to fulfill marginalization constraints globally, but only locally [WJ08]. That is marginals $b_{(x,y),r}$ are not required to arise from a common joint distribution $p_{(x,y)}$. In addition, we approximate the entropy via the fractional entropy [WH03], *i.e.*, $H(p_{(x,y)}) \approx \sum_r c_r H(b_{(x,y),r})$. Counting numbers c_r are employed to weight the marginal entropies. Putting all this together, we obtain the following approximation for $\epsilon \ln Z_\epsilon(x, w)$:

$$\max_{b_{(x,y)} \in \mathcal{C}_{(x,y)}} \sum_{r, \hat{y}_r} b_{(x,y),r}(\hat{y}_r) f_r(x, \hat{y}_r; w) + \sum_r \epsilon c_r H(b_{(x,y),r}). \tag{5.4}$$

where the beliefs are constrained to the local polytope

$$\mathcal{C}_{(x,y)} = \begin{cases} \forall r & b_{(x,y),r} \in \Delta \\ \forall r, \hat{y}_r, p \in P(r) & \sum_{\hat{y}_p \setminus \hat{y}_r} b_{(x,y),p}(\hat{y}_p) = b_{(x,y),r}(\hat{y}_r), \end{cases}$$

with $P(r)$ the set of parents of region r, *i.e.*, $P(r) \subseteq \{p \in \mathcal{R} : r \subset p\}$, which subsumes those regions for which we want the marginalization constraint to hold. Conversely, we define the set of children as $C(r) = \{c \in \mathcal{R} : r \in P(c)\}$.

Algorithm: Efficient Deep Structured Learning

Repeat until stopping criteria

1. Forward pass to compute $f_r(x, \hat{y}_r; w)\ \forall (x, y), r, y_r$

2. Compute approximate beliefs $b_{(x,y),r} \propto \exp \frac{\hat{f}_r(x,\hat{y}_r;w,\lambda)}{\epsilon c_r}$ by iterating for a fixed number of times over r:

$\forall (x, y), p \in P(r), \hat{y}_r$

$$\mu_{(x,y),p\to r}(\hat{y}_r) = \epsilon c_p \ln \sum_{\hat{y}_p \setminus \hat{y}_r} \exp \frac{f_p(x, \hat{y}_p; w) - \sum_{p' \in P(p)} \lambda_{(x,y),p\to p'}(\hat{y}_p) + \sum_{r' \in C(p)\setminus r} \lambda_{(x,y),r'\to p}(\hat{y}_{r'})}{\epsilon c_p}$$

$$\lambda_{(x,y),r\to p}(\hat{y}_r) \propto \frac{c_p}{c_r + \sum_{p \in P(r)} c_p} \left(f_r(x, \hat{y}_r; w) + \sum_{c \in C(r)} \lambda_{(x,y),c\to r}(\hat{y}_c) + \sum_{p \in P(r)} \mu_{(x,y),p\to r}(\hat{y}_r) \right) - \mu_{(x,y),p\to r}(\hat{y}_r)$$

3. Backward pass via chain-rule to obtain gradient $g = \sum_{(x,y),r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) \nabla_w f_r(x, \hat{y}_r; w) - \nabla_w \overline{F}(w)$

4. Update parameters w using stepsize η via $w \leftarrow w - \eta g$

Figure 5.3: Efficient learning algorithm that blends learning and inference.

We can thus rewrite the learning problem by plugging the approximations derived in Eq. (5.4) into Eq. (5.1). This gives rise to the new approximated learning program depicted in Fig. 5.2.

To iteratively update the parameters for the non-smooth approximated cost function given in Fig. 5.2 we require a sub-gradient w.r.t. w, which in turn requires to solve the maximization w.r.t. the beliefs b exactly. This is a non-trivial task in itself as inference in general graphical models is NP-hard. Iterative message passing algorithms [Pea88, YFW05, WJW05, WYM07, SMG08, MGW09] are typically employed for approximate inference. Importantly, note that combining the procedure outlined in Fig. 5.1 with iterative message passing to approximate $p_{(x,y)}(\hat{y}|w, \epsilon)$ results in a double-loop algorithm which would be slow for many graphical models of interest.

5.1.3 Efficient Approximate Learning by Blending Learning and Inference

In this section we propose a more efficient algorithm that is based on the principle of blending learning (*i.e.*, parameter updates) and inference. Thus we are interested in only performing a

single message passing iteration before updating the parameters w. Note that simply reducing the number of iterations is generally not an option as the obtained beliefs $b_{(x,y),r}$ are by no means accurate. However, assuming all counting numbers c_r to be positive, we can derive an algorithm that is able to interleave minimization w.r.t. w and maximization of the beliefs b. Such a procedure is more efficient as we are able to update the parameters w much more frequently.

To interleave both programs we first convert maximization of the beliefs into a minimization by employing the dual program as detailed for general scoring functions in the following claim, which was discussed for log-linear models in seminal work by [TCK05]. This conversion is possible since the maximization problem is concave in $b_{(x,y)}$ if $\forall r$, $\epsilon c_r \geq 0$.

Claim 1. *Assume $\epsilon c_r \geq 0 \; \forall r$, and let $\overline{F}(w) = \sum_{(x,y)\in\mathcal{D}} F(x,y;w)$ denote the sum of empirical function observations. Let $\lambda_{(x,y),r\to p}(\hat{y}_r)$ be the Lagrange multipliers for each marginalization constraint $\sum_{\hat{y}_p\setminus\hat{y}_r} b_{(x,y),p}(\hat{y}_p) = b_{(x,y),r}(\hat{y}_r)$ within the polytope $\mathcal{C}_{(x,y)}$. Then the approximated general structured prediction task shown in Fig. 5.2 is equivalent to*

$$\min_{w,\lambda} \sum_{(x,y),r} \epsilon c_r \ln \sum_{\hat{y}_r} \exp \frac{\hat{f}_r(x,\hat{y}_r;w,\lambda)}{\epsilon c_r} - \overline{F}(w), \tag{5.5}$$

where we employed the re-parameterization score $\hat{f}_r(x,\hat{y}_r;w,\lambda) = f_r(x,\hat{y}_r;w) + \sum_{c\in C(r)} \lambda_{(x,y),c\to r}(\hat{y}_c) - \sum_{p\in P(r)} \lambda_{(x,y),r\to p}(\hat{y}_r)$.

Proof: To obtain the dual of the maximization w.r.t. $b_{(x,y)}$ we utilize its Lagrangian

$$L_{(x,y)} = \sum_{r,\hat{y}_r} b_{(x,y),r}(\hat{y}_r) \hat{f}_r(x,\hat{y}_r;w,\lambda) + \sum_r \epsilon c_r H(b_{(x,y),r}).$$

Maximization of the Lagrangian w.r.t. the primal variables b is possible by employing the relationship stated in Eq. (5.3) locally $\forall r$. We then obtain the dual function being the first term in Eq. (5.5). For strict convexity, *i.e.*, $\epsilon c_r > 0$, we reconstruct the beliefs to be proportional to the exponentiated, loss-augmented re-parameterization score, *i.e.*, formally $b_{(x,y),r} \propto \exp \frac{\hat{f}_r(x,\hat{y}_r;w,\lambda)}{\epsilon c_r}$. For $\epsilon c_r = 0$ the beliefs correspond to a uniform distribution over the set of maximizers of the loss-augmented re-parameterization score $\hat{f}_r(x,\hat{y}_r;w,\lambda)$. ■

It is important to note that by applying duality we managed to convert the min-max task in Fig. 5.2 into a single minimization as shown in Eq. (5.5). Performing block coordinate descent

Figure 5.4: Samples from the Word50 dataset. Note the high degree of rotation, scaling and translation.

updates to minimize Eq. (5.5), we are therefore able to interleave both, updating the weights (*i.e.*, learning) and the messages (*i.e.*, inference). This results in a more efficient algorithm, as inference does not have to be run until convergence, even a single update of the messages suffices. We note that this is possible only if $\epsilon c_r \geq 0 \; \forall r$. Strictly speaking, we require concavity only within the set of feasible beliefs $\mathcal{C}_{(x,y)}$. However, for simplicity of the derivations and descriptions we neglected such an extension.

Fig. 5.3 summarizes our efficient deep structured prediction algorithm which iterates between the following steps. Given parameters w we perform a standard forward pass to compute $f_r(x, \hat{y}_r; w)$ for all regions r. Depending on the model, computation of f_r can sometimes be carried out more efficiently via a single convolutional neural network which combines all the data. We then iterate through all regions r and use block-coordinate descent to find the globally optimal value of Eq. (5.5) w.r.t. $\lambda_{(x,y),r \to p}(\hat{y}_r) \; \forall (x, y), \hat{y}_r, p \in P(r)$. This can be done in closed form and therefore is computed very efficiently [GJ07, SMG08, HS10, Sch13]. We then compute the gradient using a standard backward pass before we jointly update all the parameters w by performing a step of size η along the negative gradient.

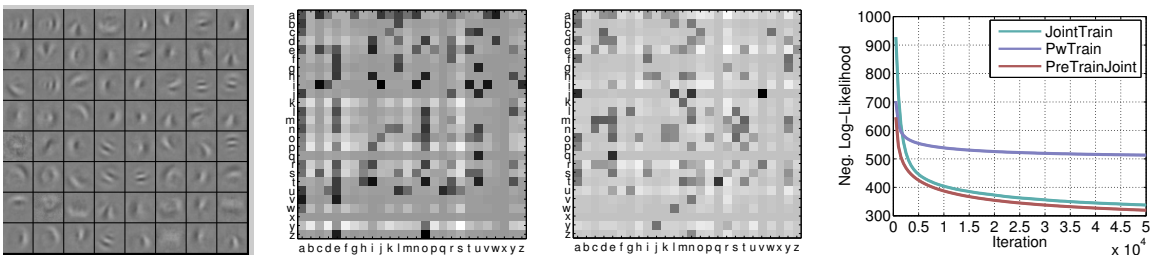
5.1.4 Implementation Details

We implemented the general algorithm presented in Fig. 5.3 in C++ as a library for Linux, Windows and Mac platforms. It supports usage of the GPU for the forward and backward pass using both, standard linear algebra packages and manually tuned GPU-kernels. In addition to standard

Graph	MLP	Method	$H_1 = 128$	$H_1 = 256$	$H_1 = 512$	$H_1 = 768$	$H_1 = 1024$
1st order Markov	One Layer	Unary only	8.60 / 61.32	10.80 / 64.41	12.50 / 65.69	12.95 / 66.66	13.40 / 67.02
		JointTrain	16.80 / 65.28	25.20 / 70.75	**31.80** / 74.90	33.05 / 76.42	34.30 / 77.02
		PwTrain	12.70 / 64.35	18.00 / 68.27	22.80 / 71.29	23.25 / 72.62	26.30 / 73.96
		PreTrainJoint	**20.65 / 67.42**	**25.70 / 71.65**	31.70 / **75.56**	**34.50 / 77.14**	**35.85 / 78.05**
2nd order Markov	One Layer	JointTrain	25.50 / 67.13	34.60 / 73.19	45.55 / 79.60	**51.55 / 82.37**	**54.05 / 83.57**
		PwTrain	10.05 / 58.90	14.10 / 63.44	18.10 / 67.31	20.40 / 70.14	22.20 / 71.25
		PreTrainJoint	**28.15 / 69.07**	**36.85 / 75.21**	**45.75 / 80.09**	50.10 / 82.30	52.25 / 83.39
		$H_1 = 512$	$H_2 = 32$	$H_2 = 64$	$H_2 = 128$	$H_2 = 256$	$H_2 = 512$
1st order Markov	Two Layer	Unary only	15.25 / 69.04	18.15 / 70.66	19.00 / 71.43	19.20 / 72.06	20.40 / 72.51
		JointTrain	35.95 / 76.92	43.80 / 81.64	44.75 / 82.22	46.00 / 82.96	**47.70** / 83.64
		PwTrain	34.85 / 79.11	38.95 / 80.93	42.75 / 82.38	45.10 / 83.67	45.75 / 83.88
		PreTrainJoint	**42.25 / 81.10**	**44.85 / 82.96**	**46.85 / 83.50**	**47.95 / 84.21**	47.05 / **84.08**
2nd order Markov	Two Layer	JointTrain	54.65 / 83.98	61.80 / 87.30	66.15 / 89.09	64.85 / 88.93	68.00 / 89.96
		PwTrain	39.95 / 81.14	48.25 / 84.45	52.65 / 86.24	57.10 / 87.61	62.90 / 89.49
		PreTrainJoint	**62.60 / 88.03**	**65.80 / 89.32**	**68.75 / 90.47**	**68.60 / 90.42**	**69.35 / 90.75**

Table 5.1: Word / Character accuracy. Performance improves as (1) joint-training is employed, (2) the model is more structured, and (3) deeper unary classifiers are utilized. The number of hidden units for the first and second layer are denoted as H_1 and H_2 respectively.

gradient descent, we allow specification of both mini-batches, moments and different regularizers like 2-norm and ∞-norm. Between iterations the step-size can be reduced based on either the negative log-likelihood or validation set performance. Our function F is specified using a directed a-cyclic graph. Hence we support an arbitrarily nested function structure composed of data, parameters and function prototypes (convolution, affine function aka fully connected, dropout, local response normalization, pooling, rectified linear, sigmoid and softmax units). The aforementioned library is accompanied by a program performing learning, inference and gradient checks. To accommodate for large datasets it reads data from HDF5 storage while a second thread simultaneously performs the computation. This is useful since we can prepare the data for the next pass while conducting the computation. Google protocol buffers are employed to effectively specify the function F without the need to modify any source code. The library is released on http://alexander-schwing.de.



Unary weights distance-1 edges distance-2 edges Neg. Log-Likelihood

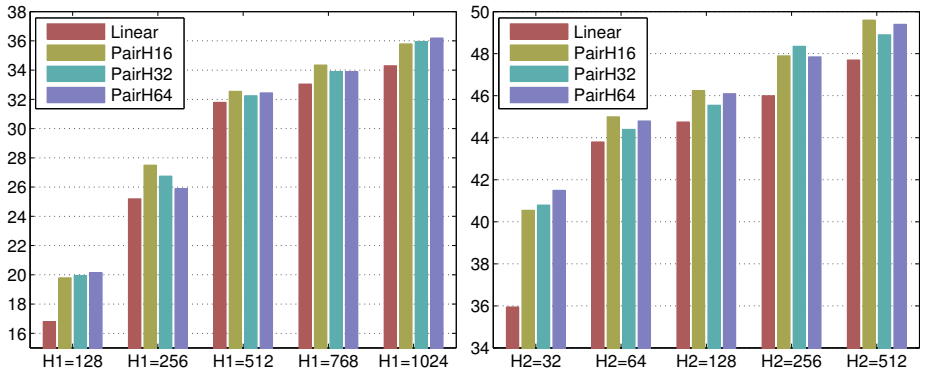
Figure 5.5: (left) Subset of the learned unary weights. Pairwise weights (middle two panels), the darker, the larger the weight. (right) Negative log-likelihood for different learning approaches.

5.2 Experimental Evaluation

We demonstrate the performance of our model on two tasks: word recognition and image classification. We investigate four strategies to learn the model parameters. '**Unary only**' denotes training only unary classifiers while ignoring the structure of the graphical model, *i.e.*, pairwise weights are equal to 0. '**JointTrain**' initializes all weights at random and trains them jointly. '**PwTrain**' uses piecewise training by first training the unary potentials and then keeping them fixed when learning the pairwise potentials. '**PreTrainJoint**' pre-trains the unaries but jointly optimizes pairwise weights as well as unary weights in a second step.

5.2.1 Word Recognition: Word50

Our first task consists of word recognition from noisy images. Towards this goal, we created a challenging dataset by randomly selecting 50 words, each consisting of five characters. We then generated writing variations of each word as follows: we took the lower case characters from the Chars74K dataset [CBV09], and inserted them in random background image patches (similar to [LEC07]) by alpha matting, *i.e.*, characters have transparency. To increase the difficulty, we perturbed each character image of size 28×28 by scaling, rotation and translation. As shown in Fig. 5.4 the task is very challenging, some characters are fairly difficult to recognize even for humans. We denote the resulting dataset '**Word50**.' The training, validation and test sets have $10,000$, $2,000$ and $2,000$ variations of words respectively.



One-Layer MLP Chain Two-Layer MLP Chain

Figure 5.6: Learning non-linear pairwise functions: Word recognition as a function of the number of hidden units for the unary potential. Colors represent different number of hidden units for the pairwise potentials. The y-axis shows the word accuracy of using Linear function, or 16 (PairH16), 32 (PairH32), and 64 (PairH64) hidden units for the pairwise function.

We experimented with graphical models composed of unary and pairwise regions defined over five random variables, one per character. We encode unary potentials $f_r(x, y_i; w_u)$ using multi-layer perceptrons (MLPs) with rectified linear units (ReLU). Unless otherwise stated, we define all pairwise interactions via

$$f_r(x, y_i, y_j; w_p) = \sum_{mn} W_{mn} \cdot \delta(y_i = m, y_j = n), \tag{5.6}$$

where $r = \{i, j\}$, $w_p = \{W\}$, W_{mn} is the element of matrix W, and δ refers to the indicator function.

For all experiments, we share all unary weights across the nodes of the graphical model as well as all pairwise weights for all edges. Note that due to the use of ReLU units, the negative log-likelihood is non-smooth, non-linear and non-convex w.r.t. w. Because of the non-smoothness of F, we utilize momentum based sub-gradient descent methods to estimate the weights. In particular, we use a mini-batch size of 100, a step size of 0.01 and a momentum of 0.95. If the unary potential is pre-trained, the initial step size is reduced to 0.001. All the unary classifiers are trained with $100,000$ iterations over mini-batches. For all experiments, the validation set is only used to decrease the step size, *i.e.*, if the accuracy on the validation set decreases, we reduce the step size





female/indoor/portrait	sky/plant life/tree	water/animals/sea	animals/dog/indoor	indoor/flower/plant life
female/indoor/portrait	sky/plant life/tree	water/animals/sky	animals/dog	$\emptyset$

Figure 5.7: Flickr test set images and a subset of the assigned tags as well as our predictions (bottom row).

by 0.5. We use $\epsilon = 1$, set $c_r = 1$ for all regions r, and perform 10 message passing iterations to compute the marginal beliefs $b_{(x,y),r}$ at step 2 in Fig. 5.3 when dealing with loopy models.

We experiment with two graphical models, Markov models of first (*i.e.*, there are links only between y_i and y_{i+1}) and second order (*i.e.*, there are links between y_i and y_{i+1}, y_{i+2}) as well as two types of unary potentials with varying degree of structure. We report two metrics, the average character and word accuracy, which correspond to Hamming loss and zero-one loss respectively. Tab. 5.1 depicts the results for the different models, learning strategies and number of hidden units. We observe the following trends.

Joint training helps: Joint training with pre-trained unary classifiers (PreTrainJoint) outperforms all the other approaches in almost all cases. Piecewise training (PwTrain), unable to adapt the non-linearities while learning pairwise weights, is worst than joint training.

Structure helps: Adding structure to the model is key to capture complex dependencies. As shown in Tab. 5.1, more structured models (*i.e.*, second order Markov model) consistently improves performance.

Deep helps: We tested our models using one layer and two-layer perceptrons with both short-range and long-range connections in the MRF. For the two-layer MLP, the number of hidden units in the first layer is fixed to $H_1 = 512$, and we varied the number of hidden units H_2 in the sec-

ond layer. As shown in Tab. 5.1, the deeper and the more structured the model is, the better the performance we achieve. As expected, performance also increases with the number of hidden units.

Efficiency: Using GPUs, it takes on average 0.064s per iteration for the 1st order Markov model and 0.104s for the 2nd order Markov model. The time employed for training one layer *vs*. the multi-layer models is approximately the same. Note that our approach is very efficient, as this is the time per iteration to train 831,166 weights.

Learned parameters: As shown in the left column of Fig. 5.5, the learned unary weights resemble character strokes. The middle two panels show the learned pairwise weights for distance-1 edges (*i.e*., edges with only neighboring connections) and distance-2 edges (*i.e*., edges connecting every other variable). For example, it shows that 'q' is likely to be followed by 'u,' and 'e' is likely to be distance-2 away from 'q' in this dataset. On the right-most panel, we also show the negative log-likelihood as a function of the number of joint training iterations. PreTrainJoint can achieve the lowest cost value, while PwTrain has the highest value.

Non-linear pairwise functions: To further demonstrate the generality of our approach, we replaced the linear pairwise function in Eq. (5.6) by a one-layer MLP, while keeping the other settings identical. For this experiment we utilize a 1st order Markov model. As shown in Fig. 5.6, our model attains best performance when using a non-linear pairwise function. We found 16 to 64 hidden units for the non-linear pairwise function to be sufficient for modeling the bi-gram combinations in this dataset. In this case the largest model has 974,846 weights and training takes on average 0.068s per iteration.

5.2.2 Image Tagging: Flickr

We next evaluate the importance of blending learning and inference. Towards this goal, we make use of the Flickr dataset, which consists of $10,000$ training and $10,000$ test images from Flickr. The task is to predict which of 38 possible tags should be assigned to each image. Fig. 5.7 shows

Method	Mean error
Unary only	9.36
PwTrain	7.70
PreTrainJoint	**7.25**

Table 5.2: Flickr Hamming loss: Joint training of deep features and the MRF improves performance.

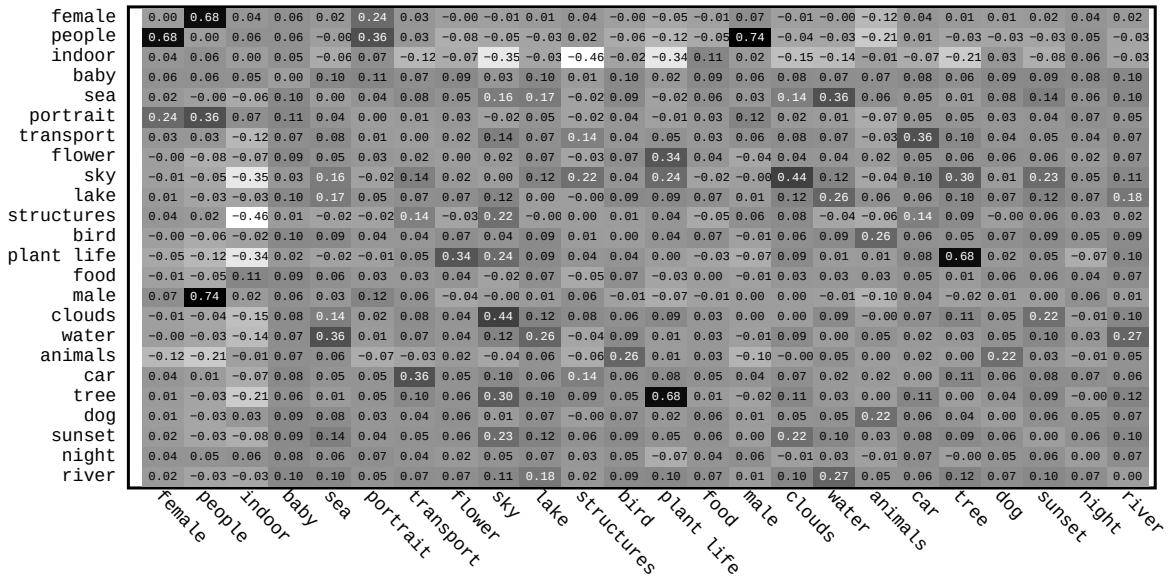


Figure 5.8: Correlation matrix (*i.e.*, pairwise potentials) learned on the Flickr dataset.

some examples. The graphical model has 38 binary random variables, each denoting the presence/absence of a particular tag. We define the non-linear unaries $f_r(x, y_i; w_u)$ using the 8-layer deep-net architecture from [KSH13], followed by a 76-dimensional top layer. Hence the function is composed out of two subsequent stacks of convolution, rectified linear (ReLU), pooling and local response normalization units. Those are followed by three convolution–ReLU function pairs. Afterwards pooling is applied before two fully-connected–ReLU–dropout combinations are employed to yield the input into a fully connected layer which finally computes the unary potentials. We employ pairwise potentials similar to Eq. (5.6) which now fully model the correlations between

any pair of output variables. This amounts to a total of $57,182,408$ parameters arising from the convolutional units, fully connected units and corresponding biases as well as the pairwise weights.

We use a momentum based sub-gradient method for training with a mini-batch size of 300, a step size of 0.0001, a momentum of 0.95 and set $\epsilon = 1$ and $c_r = 1\ \forall r$. We initialized the deep-net parameters using a model pre-trained on ImageNet [DDS09]. Our error metric is the classification error, *i.e.*, Hamming loss.

Joint training helps: Results on the test set are summarized in Tab. 5.2. Similar to the Word50 dataset we observe that joint training is beneficial. We provide examples for perfect (two left-most images), roughly accurate and failing predictions (right image) in Fig. 5.7.

Learned pairwise weights: In Fig. 5.8 we illustrate the learned correlations for a subset of the 38 classes. We observe that the class 'people' correlates highly with 'female,' 'male,' and 'portrait.' The 'indoor' tag does not co-occur with 'sky,' 'structures,' 'plant life' and 'tree.' 'Sea' appears typically with 'water,' 'clouds,' 'lake' and 'sky.'

Efficiency of Blending: To illustrate that blending is indeed beneficial we compare the negative log-likelihood and the training error as a function of run-time in Fig. 5.9. The standard approach is limited to 20 iterations of message passing to avoid time-consuming, repeated computation of a stopping criterion involving both the approximated log-partition function and its dual. As show in Fig. 5.9 blending learning and inference speeds up parameter estimation significantly. For larger graphical models, we expect the differences to be even more significant.

5.3 Discussion

Joint training of neural networks and graphical models: Neural Networks have been incorporated as unary potentials in graphical models. One of the earliest works by [Bri90] jointly optimizes a system consisting of multilayer perceptrons and hidden Markov models for speech recognition. For document processing systems, [BBL97] propose Graph Transformer Networks to

jointly optimize sub-tasks, such as word segmentation and character recognition. Several works [CWB11, PBX09, MPW12, DA10, PF10, MF08] have extended the linear unary potential in MRFs to incorporate non-linearities. However, they assume that exact inference can be performed either via a forward-backward pass within the graphical model or dynamic programming. In contrast, in this chapter we present learning algorithms for general graphical models, where inference is NP-hard. Moreover, all the previous works (except [DA10]) do not consider max-margin loss during training which is incorporated into our framework by choosing $\epsilon = 0$. More recently, [LZ14] use a hinge loss to learn the unary term defined as a neural net, but keep the pairwise potentials fixed (*i.e.*, no joint training). [Dom13] considers non-linear structured prediction and decomposes the learning problem into a subset of logistic regressors, which require the parameter updates to be run till convergence before updating the messages. [TJL14] also jointly train convolutional neural networks and a graphical model for pose estimation. However, the MRF inference procedure is approximated by their Spatial-Model which ignores the partition function. [JNS12, JNR13] showed the benefits of a combination of structured models with classification trees. Since the submission of our work, [SU15, ZJR15] proposed the use of joint training using a double loop algorithm when efficient mean field updates are possible. State-of-the-art was achieved when using the semantic segmentation graphical model of [CPK15].

Blending learning and inference: In this chapter we defined learning to be a $\min$-$\max$ task. The blending strategy, which was previously employed for learning log-linear models by [MSJ10, HU10], amounts to converting the maximization task into a minimization problem using its dual. Subsequently we make use of block-coordinate descent strategies to obtain a more efficient algorithm. Importantly any order of block-updates is possible. It remains an open problem to find the optimal tradeoff.

5.4 Conclusion

We have proposed an efficient algorithm to learn deep models enriched to capture the dependencies between the output variables. Our experiments on word prediction from noisy images and image tagging showed that the deeper and the more structured the model, the better the performance we

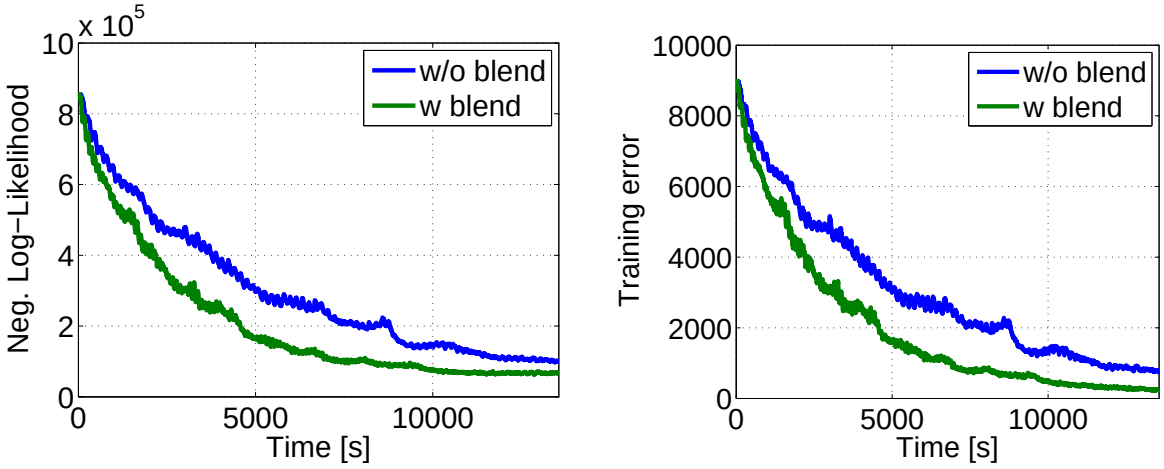


Figure 5.9: Blending learning and inference speeds-up training significantly.

achieve. Furthermore, joint learning of all weights outperforms all other strategies.

Joint training of deep convolutional neural networks and conditional random field has been employred by recent works [ZJR15, LSR15, LLL15]. They have demonstrated outstanding performance on the task of semantic image segmentation.

CHAPTER 6

Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs

Deep Convolutional Neural Networks (DCNNs) had been the method of choice for document recognition since [LBB98], but have only recently become the mainstream of high-level vision research. Over the past two years DCNNs have pushed the performance of computer vision systems to soaring heights on a broad array of high-level problems, including image classification [KSH13, SEZ13, SZ14, SLJ14, PKS14], object detection [GDD14], fine-grained categorization [ZDG14], among others. A common theme in these works is that DCNNs trained in an end-to-end manner deliver strikingly better results than systems relying on carefully engineered representations, such as SIFT or HOG features. This success can be partially attributed to the built-in invariance of DCNNs to local image transformations, which underpins their ability to learn hierarchical abstractions of data [ZF14]. While this invariance is clearly desirable for high-level vision tasks, it can hamper low-level tasks, such as pose estimation [CY14, TJL14] and semantic segmentation - where we want precise localization, rather than abstraction of spatial details.

There are two technical hurdles in the application of DCNNs to image labeling tasks: signal downsampling, and spatial 'insensitivity' (invariance). The first problem relates to the reduction of signal resolution incurred by the repeated combination of max-pooling and downsampling ('striding') performed at every layer of standard DCNNs [KSH13, SZ14, SLJ14]. Instead, as in [PKS14], we employ the 'atrous' (with holes) algorithm originally developed for efficiently computing the undecimated discrete wavelet transform [Mal99]. This allows efficient dense computation of DCNN responses in a scheme substantially simpler than earlier solutions to this problem [GCM13, SEZ13].

The second problem relates to the fact that obtaining object-centric decisions from a classifier requires invariance to spatial transformations, inherently limiting the spatial accuracy of the DCNN model. We boost our model's ability to capture fine details by employing a fully-connected Conditional Random Field (CRF). Conditional Random Fields have been broadly used in semantic segmentation to combine class scores computed by multi-way classifiers with the low-level information captured by the local interactions of pixels and edges [RKB04, SWR09] or superpixels [LLB11]. Even though works of increased sophistication have been proposed to model the hierarchical dependency [HZC04, LRK09, LVZ11] and/or high-order dependencies of segments [DOI12, GBW10, KLT09, CPY13, WSL15], we use the fully connected pairwise CRF proposed by [KK11] for its efficient computation, and ability to capture fine edge details while also catering for long range dependencies. That model was shown in [KK11] to largely improve the performance of a boosting-based pixel-level classifier, and in our work we demonstrate that it leads to state-of-the-art results when coupled with a DCNN-based pixel-level classifier.

The three main advantages of our "DeepLab" system are (i) speed: by virtue of the 'atrous' algorithm, our dense DCNN operates at 8 fps, while Mean Field Inference for the fully-connected CRF requires 0.5 second, (ii) accuracy: we obtain state-of-the-art results on the PASCAL semantic segmentation challenge, outperforming the second-best approach of [MYS14] by a margin of 7.2% and (iii) simplicity: our system is composed of a cascade of two fairly well-established modules, DCNNs and CRFs.

6.1 Convolutional Neural Networks for Dense Image Labeling

Herein we describe how we have re-purposed and finetuned the publicly available Imagenet-pretrained state-of-art 16-layer classification network of [SZ14] (VGG-16) into an efficient and effective dense feature extractor for our dense semantic image segmentation system.

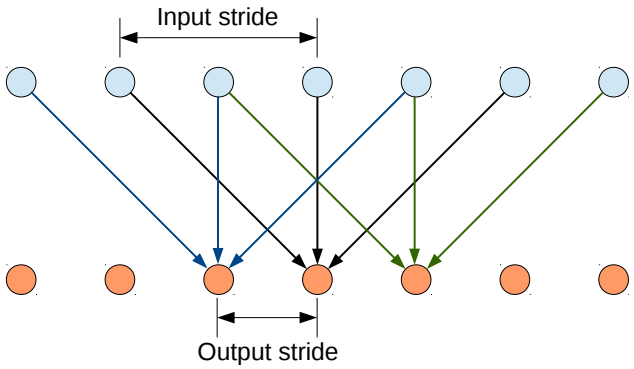


Figure 6.1: Illustration of the hole algorithm in 1-D, when *kernel_size = 3*, *input_stride = 2*, and *output_stride = 1*.

6.1.1 Efficient Dense Sliding Window Feature Extraction with the Hole Algorithm

Dense spatial score evaluation is instrumental in the success of our dense CNN feature extractor. As a first step to implement this, we convert the fully-connected layers of VGG-16 into convolutional ones and run the network in a convolutional fashion on the image at its original resolution. However this is not enough as it yields very sparsely computed detection scores (with a stride of 32 pixels). To compute scores more densely at our target stride of 8 pixels, we develop a variation of the method previously employed by [GCM13, SEZ13]. We skip subsampling after the last two max-pooling layers in the network of [SZ14] and modify the convolutional filters in the layers that follow them by introducing zeros to increase their length ($2\times$ in the last three convolutional layers and $4\times$ in the first fully connected layer). We can implement this more efficiently by keeping the filters intact and instead sparsely sample the feature maps on which they are applied on using an input stride of 2 or 4 pixels, respectively. This approach, illustrated in Fig. 6.1 is known as the 'hole algorithm' ('atrous algorithm') and has been developed before for efficient computation of the undecimated wavelet transform [Mal99]. We have implemented this within the Caffe framework [JSD14] by adding to the *im2col* function (it converts multi-channel feature maps to vectorized patches) the option to sparsely sample the underlying feature map. This approach is generally applicable and allows us to efficiently compute dense CNN feature maps at any target subsampling rate without introducing any approximations.

We finetune the model weights of the Imagenet-pretrained VGG-16 network to adapt it to

the image classification task in a straightforward fashion, following the procedure of [LSD14]. We replace the 1000-way Imagenet classifier in the last layer of VGG-16 with a 21-way one. Our loss function is the sum of cross-entropy terms for each spatial position in the CNN output map (subsampled by 8 compared to the original image). All positions and labels are equally weighted in the overall loss function. Our targets are the ground truth labels (subsampled by 8). We optimize the objective function with respect to the weights at all network layers by the standard SGD procedure of [KSH13].

During testing, we need class score maps at the original image resolution. As illustrated in Figure 6.2 and further elaborated in Section 6.2.1, the class score maps (corresponding to log-probabilities) are quite smooth, which allows us to use simple bilinear interpolation to increase their resolution by a factor of 8 at a negligible computational cost. Note that the method of [LSD14] does not use the hole algorithm and produces very coarse scores (subsampled by a factor of 32) at the CNN output. This forced them to use learned upsampling layers, significantly increasing the complexity and training time of their system: Fine-tuning our network on PASCAL VOC 2012 takes about 10 hours, while they report a training time of several days (both timings on a modern GPU).

6.1.2 Controlling the Receptive Field Size and Accelerating Dense Computation with Convolutional Nets

Another key ingredient in re-purposing our network for dense score computation is explicitly controlling the network's receptive field size. Most recent DCNN-based image recognition methods rely on networks pre-trained on the Imagenet large-scale classification task. These networks typically have large receptive field size: in the case of the VGG-16 net we consider, its receptive field is 224×224 (with zero-padding) and 404×404 pixels if the net is applied convolutionally. After converting the network to a fully convolutional one, the first fully connected layer has 4,096 filters of large 7×7 spatial size and becomes the computational bottleneck in our dense score map computation.

We have addressed this practical problem by spatially subsampling (by simple decimation) the

first FC layer to 4×4 (or 3×3) spatial size. This has reduced the receptive field of the network down to 128×128 (with zero-padding) or 308×308 (in convolutional mode) and has reduced computation time for the first FC layer by $2-3$ times. Using our Caffe-based implementation and a Titan GPU, the resulting VGG-derived network is very efficient: Given a 306×306 input image, it produces 39×39 dense raw feature scores at the top of the network at a rate of about 8 frames/sec during testing. The speed during training is 3 frames/sec. We have also successfully experimented with reducing the number of channels at the fully connected layers from 4,096 down to 1,024, considerably further decreasing computation time and memory footprint without sacrificing performance, as detailed in Section 7.2. Using smaller networks such as [KSH13] could allow video-rate test-time dense feature computation even on light-weight GPUs.

6.2 Detailed Boundary Recovery: Fully-Connected Conditional Random Fields and Multi-scale Prediction

6.2.1 Deep Convolutional Networks and the Localization Challenge

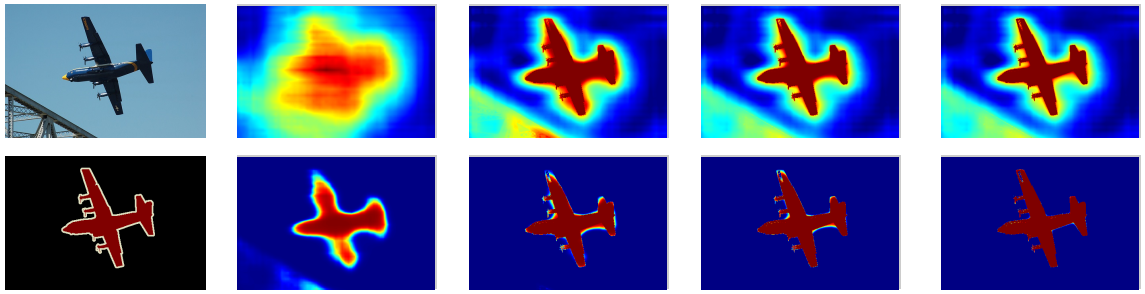
As illustrated in Figure 6.2, DCNN score maps can reliably predict the presence and rough position of objects in an image but are less well suited for pin-pointing their exact outline. There is a natural trade-off between classification accuracy and localization accuracy with convolutional networks: Deeper models with multiple max-pooling layers have proven most successful in classification tasks, however their increased invariance and large receptive fields make the problem of inferring position from the scores at their top output levels more challenging.

Recent work has pursued two directions to address this localization challenge. The first approach is to harness information from multiple layers in the convolutional network in order to better estimate the object boundaries [LSD14, EF14]. The second approach is to employ a super-pixel representation, essentially delegating the localization task to a low-level segmentation method. This route is followed by the very successful recent method of [MYS14].

In Section 6.2.2, we pursue a novel alternative direction based on coupling the recognition capacity of DCNNs and the fine-grained localization accuracy of fully connected CRFs and show that

it is remarkably successful in addressing the localization challenge, producing accurate semantic segmentation results and recovering object boundaries at a level of detail that is well beyond the reach of existing methods.

6.2.2 Fully-Connected Conditional Random Fields for Accurate Localization



Image/G.T. DCNN output CRF Iteration 1 CRF Iteration 2 CRF Iteration 10

Figure 6.2: Score map (input before softmax function) and belief map (output of softmax function) for Aeroplane. We show the score (1st row) and belief (2nd row) maps after each mean field iteration. The output of last DCNN layer is used as input to the mean field inference. Best viewed in color.

Traditionally, conditional random fields (CRFs) have been employed to smooth noisy segmentation maps [RKB04, KLT09]. Typically these models contain energy terms that couple neighboring nodes, favoring same-label assignments to spatially proximal pixels. Qualitatively, the primary function of these short-range CRFs has been to clean up the spurious predictions of weak classifiers built on top of local hand-engineered features.

Compared to these weaker classifiers, modern DCNN architectures such as the one we use in this work produce score maps and semantic label predictions which are qualitatively different. As illustrated in Figure 6.2, the score maps are typically quite smooth and produce homogeneous classification results. In this regime, using short-range CRFs can be detrimental, as our goal should be to recover detailed local structure rather than further smooth it. Using contrast-sensitive potentials [RKB04] in conjunction to local-range CRFs can potentially improve localization but still miss thin-structures and typically requires solving an expensive discrete optimization problem.

To overcome these limitations of short-range CRFs, we integrate into our system the fully connected CRF model of [KK11]. The model employs the energy function

$$E(\boldsymbol{x}) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j) \tag{6.1}$$

where $\boldsymbol{x}$ is the label assignment for pixels. We use as unary potential $\theta_i(x_i) = -\log P(x_i)$, where $P(x_i)$ is the label assignment probability at pixel i as computed by DCNN. The pairwise potential is $\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{K} w_m \cdot k^m(\boldsymbol{f}_i, \boldsymbol{f}_j)$, where $\mu(x_i, x_j) = 1$ if $x_i \neq x_j$, and zero otherwise (*i.e.*, Potts Model). There is one pairwise term for each pair of pixels i and j in the image no matter how far from each other they lie, *i.e.* the model's factor graph is fully connected. Each k^m is the Gaussian kernel depends on features (denoted as $\boldsymbol{f}$) extracted for pixel i and j and is weighted by parameter w_m. We adopt bilateral position and color terms, specifically, the kernels are

$$w_1 \exp\Big(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2}\Big) + w_2 \exp\Big(-\frac{||p_i - p_j||^2}{2\sigma_\gamma^2}\Big) \tag{6.2}$$

where the first kernel depends on both pixel positions (denoted as p) and pixel color intensities (denoted as I), and the second kernel only depends on pixel positions. The hyper parameters σ_α, σ_β and σ_γ control the "scale" of the Gaussian kernels.

Crucially, this model is amenable to efficient approximate probabilistic inference [KK11]. The message passing updates under a fully decomposable mean field approximation $b(\boldsymbol{x}) = \prod_i b_i(x_i)$ can be expressed as convolutions with a Gaussian kernel in feature space. High-dimensional filtering algorithms [ABD10] significantly speed-up this computation resulting in an algorithm that is very fast in practice, less that 0.5 sec on average for Pascal VOC images using the publicly available implementation of [KK11].

6.2.3 Multi-Scale Prediction

Following the promising recent results of [HAG14a, LSD14] we have also explored a multi-scale prediction method to increase the boundary localization accuracy. Specifically, we attach to the input image and the output of each of the first four max pooling layers a two-layer MLP (first layer: 128 3x3 convolutional filters, second layer: 128 1x1 convolutional filters) whose feature

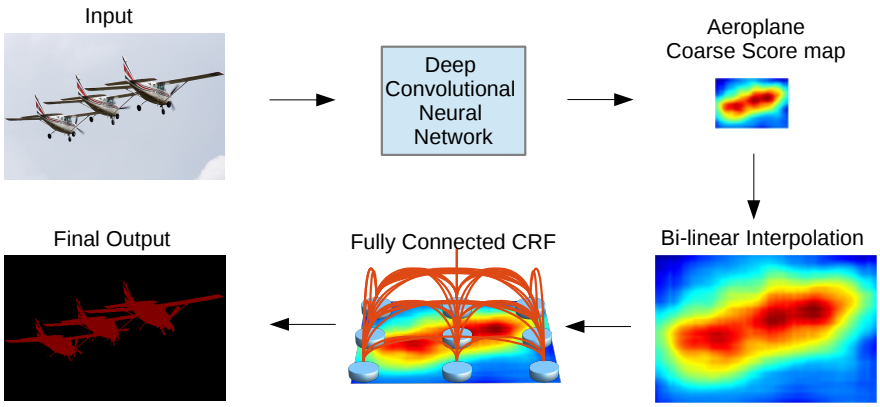


Figure 6.3: Model Illustration. The coarse score map from Deep Convolutional Neural Network (with fully convolutional layers) is upsampled by bi-linear interpolation. A fully connected CRF is applied to refine the segmentation result. Best viewed in color.

map is concatenated to the main network's last layer feature map. The aggregate feature map fed into the softmax layer is thus enhanced by 5 * 128 = 640 channels. We only adjust the newly added weights, keeping the other network parameters to the values learned by the method of Section 6.1. As discussed in the experimental section, introducing these extra direct connections from fine-resolution layers improves localization performance, yet the effect is not as dramatic as the one obtained with the fully-connected CRF.

6.3 Experimental Evaluation

Dataset We test our DeepLab model on the PASCAL VOC 2012 segmentation benchmark [EEG14], consisting of 20 foreground object classes and one background class. The original dataset contains $1,464$, $1,449$, and $1,456$ images for training, validation, and testing, respectively. The dataset is augmented by the extra annotations provided by [HAB11], resulting in $10,582$ training images. The performance is measured in terms of pixel intersection-over-union (IOU) averaged across the 21 classes.

Training We adopt the simplest form of piecewise training, decoupling the DCNN and CRF training stages, assuming the unary terms provided by the DCNN are fixed during CRF training.

Method	mean IOU (%)
DeepLab	59.80
DeepLab-CRF	63.74
DeepLab-MSc	61.30
DeepLab-MSc-CRF	65.21
DeepLab-7x7	64.38
DeepLab-CRF-7x7	67.64
DeepLab-LargeFOV	62.25
DeepLab-CRF-LargeFOV	67.64
DeepLab-MSc-LargeFOV	64.21
DeepLab-MSc-CRF-LargeFOV	68.70

(a)

Method	mean IOU (%)
MSRA-CFM	61.8
FCN-8s	62.2
TTI-Zoomout-16	64.4
DeepLab-CRF	66.4
DeepLab-MSc-CRF	67.1
DeepLab-CRF-7x7	70.3
DeepLab-CRF-LargeFOV	70.3
DeepLab-MSc-CRF-LargeFOV	71.6

(b)

Table 6.1: (a) Performance of our proposed models on the PASCAL VOC 2012 'val' set (with training in the augmented 'train' set). The best performance is achieved by exploiting both multi-scale features and large field-of-view. (b) Performance of our proposed models (with training in the augmented 'trainval' set) compared to other state-of-art methods on the PASCAL VOC 2012 'test' set.

For DCNN training we employ the VGG-16 network which has been pre-trained on ImageNet. We fine-tuned the VGG-16 network on the VOC 21-way pixel-classification task by stochastic gradient descent on the cross-entropy loss function, as described in Section 6.1.1. We use a mini-batch of 20 images and initial learning rate of 0.001 (0.01 for the final classifier layer), multiplying the learning rate by 0.1 at every 2000 iterations. We use momentum of 0.9 and a weight decay of 0.0005.

After the DCNN has been fine-tuned, we cross-validate the parameters of the fully connected CRF model in Eq. (6.2) along the lines of [KK11]. We use the default values of $w_2 = 3$ and $\sigma_\gamma = 3$ and we search for the best values of w_1, σ_α, and σ_β by cross-validation on a small subset of the validation set (we use 100 images). We employ coarse-to-fine search scheme. Specifically, the initial search range of the parameters are $w_1 \in [5, 10]$, $\sigma_\alpha \in [50 : 10 : 100]$ and $\sigma_\beta \in [3 : 1 : 10]$ (MATLAB notation), and then we refine the search step sizes around the first round's best values. We fix the number of mean field iterations to 10 for all reported experiments.

Evaluation on Validation set We conduct the majority of our evaluations on the PASCAL 'val' set, training our model on the augmented PASCAL 'train' set. As shown in Tab. 6.1 (a), incorporating the fully connected CRF to our model (denoted by DeepLab-CRF) yields a substantial performance boost, about 4% improvement over DeepLab. We note that the work of [KK11] improved the 27.6% result of TextonBoost [SWR09] to 29.1%, which makes the improvement we report here (from 59.8% to 63.7%) all the more impressive.

Turning to qualitative results, we provide visual comparisons between DeepLab and DeepLab-CRF in Fig. 7.6. Employing a fully connected CRF significantly improves the results, allowing the model to accurately capture intricate object boundaries.

Multi-Scale features We also exploit the features from the intermediate layers, similar to [HAG14a, LSD14]. As shown in Tab. 6.1 (a), adding the multi-scale features to our DeepLab model (denoted as DeepLab-MSc) improves about 1.5% performance, and further incorporating the fully connected CRF (denoted as DeepLab-MSc-CRF) yields about 4% improvement. The qualitative comparisons between DeepLab and DeepLab-MSc are shown in Fig. 6.4. Leveraging the multi-scale features can slightly refine the object boundaries.

Field of View The 'atrous algorithm' we employed allows us to arbitrarily control the Field-of-View (FOV) of the models by adjusting the input stride, as illustrated in Fig. 6.1. In Tab. 6.2, we experiment with several kernel sizes and input strides at the first fully connected layer. The method, DeepLab-CRF-7x7, is the direct modification from VGG-16 net, where the kernel size = 7×7 and input stride = 4. This model yields performance of 67.64% on the 'val' set, but it is relatively slow (1.44 images per second during training). We have improved model speed to 2.9 images per second by reducing the kernel size to 4×4. We have experimented with two such network variants with different FOV sizes, DeepLab-CRF and DeepLab-CRF-4x4; the latter has large FOV (*i.e.*, large input stride) and attains better performance. Finally, we employ kernel size 3×3 and input stride = 12, and further change the filter sizes from 4096 to 1024 for the last two layers. Interestingly, the resulting model, DeepLab-CRF-LargeFOV, matches the performance of the expensive DeepLab-CRF-7x7. At the same time, it is 3.36 times faster to run and has significantly fewer parameters

Method	kernel size	input stride	receptive field	# parameters	mean IOU (%)	Training speed (img/sec)
DeepLab-CRF-7x7	7×7	4	224	134.3M	67.64	1.44
DeepLab-CRF	4×4	4	128	65.1M	63.74	2.90
DeepLab-CRF-4x4	4×4	8	224	65.1M	67.14	2.90
DeepLab-CRF-LargeFOV	3×3	12	224	20.5M	67.64	4.84

Table 6.2: Effect of Field-Of-View. We show the performance (after CRF) and training speed on the PASCAL VOC 2012 'val' set as the function of (1) the kernel size of first fully connected layer, (2) the input stride value employed in the atrous algorithm.

(20.5M instead of 134.3M).

The performance of several model variants is summarized in Tab. 6.1, showing the benefit of exploiting multi-scale features and large FOV.

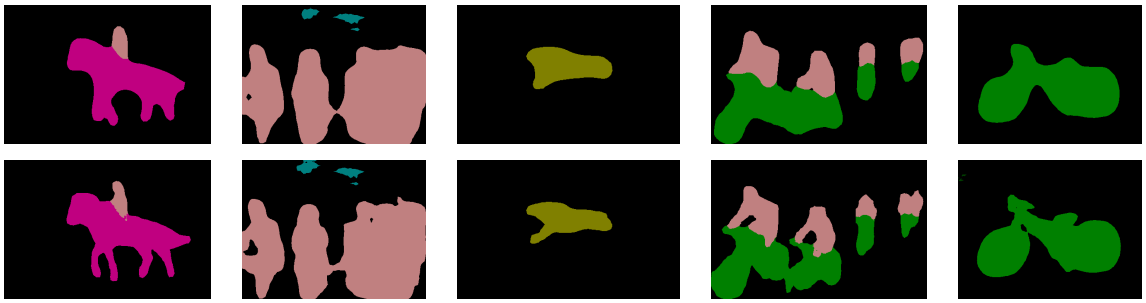
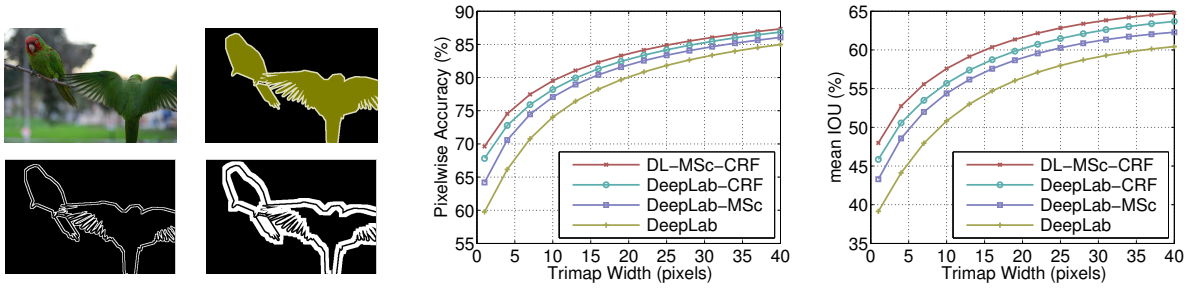


Figure 6.4: Incorporating multi-scale features improves the boundary segmentation. We show the results obtained by DeepLab and DeepLab-MSc in the first and second row, respectively. Best viewed in color.

Mean Pixel IOU along Object Boundaries To quantify the accuracy of the proposed model near object boundaries, we evaluate the segmentation accuracy with an experiment similar to [KLT09, KK11]. Specifically, we use the 'void' label annotated in val set, which usually occurs around object boundaries. We compute the mean IOU for those pixels that are located within a narrow band (called trimap) of 'void' labels. As shown in Fig. 6.5, exploiting the multi-scale features from the intermediate layers and refining the segmentation results by a fully connected CRF significantly improve the results around object boundaries.



(a) (b) (c)

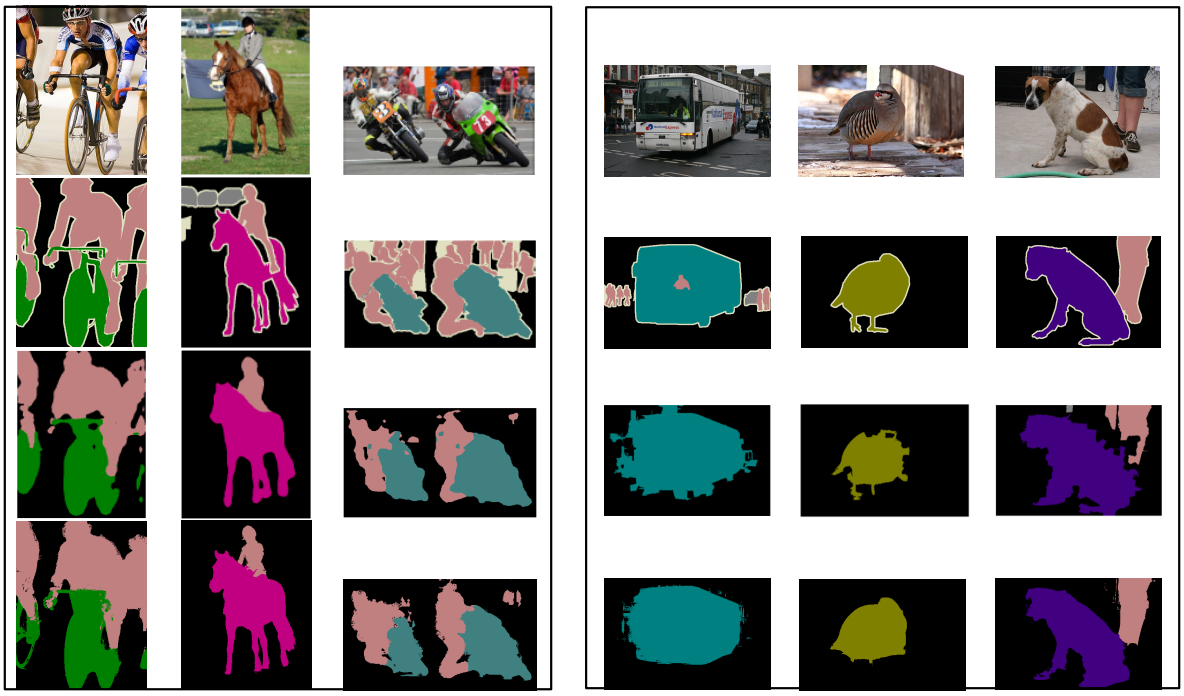
Figure 6.5: (a) Some trimap examples (top-left: image. top-right: ground-truth. bottom-left: trimap of 2 pixels. bottom-right: trimap of 10 pixels). Quality of segmentation result within a band around the object boundaries for the proposed methods. (b) Pixelwise accuracy. (c) Pixel mean IOU.

Comparison with State-of-art In Fig. 6.6, we qualitatively compare our proposed model, DeepLab-CRF, with two state-of-art models: FCN-8s [LSD14] and TTI-Zoomout-16 [MYS14] on the 'val' set (the results are extracted from their papers). Our model is able to capture the intricate object boundaries.

Reproducibility We have implemented the proposed methods by extending the excellent Caffe framework [JSD14]. We share our source code, configuration files, and trained models that allow reproducing the results in this chapter at a companion web site `https://bitbucket.org/deeplab/deeplab-public`.

Test set results Having set our model choices on the validation set, we evaluate our model variants on the PASCAL VOC 2012 official 'test' set. As shown in Tab. 6.3, our DeepLab-CRF and DeepLab-MSc-CRF models achieve performance of 66.4% and 67.1% mean IOU[1], respectively. Our models outperform all the other state-of-the-art models (specifically, TTI-Zoomout-16 [MYS14], FCN-8s [LSD14], and MSRA-CFM [DHS14]). When we increase the FOV of the models, DeepLab-CRF-LargeFOV yields performance of 70.3%, the same as DeepLab-CRF-7x7,

[1] `http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=6`



(a) FCN-8s vs. DeepLab-CRF (b) TTI-Zoomout-16 vs. DeepLab-CRF

Figure 6.6: Comparisons with state-of-the-art models on the val set. First row: images. Second row: ground truths. Third row: other recent models (Left: FCN-8s, Right: TTI-Zoomout-16). Fourth row: our DeepLab-CRF. Best viewed in color.

Method	bkg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
MSRA-CFM	-	75.7	26.7	69.5	48.8	65.6	81.0	69.2	73.3	30.0	68.7	51.5	69.1	68.1	71.7	67.5	50.4	66.5	44.4	58.9	53.5	61.8
FCN-8s	-	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
TTI-Zoomout-16	89.8	81.9	35.1	78.2	57.4	56.5	80.5	74.0	79.8	22.4	69.6	53.7	74.0	76.0	76.6	68.8	44.3	70.2	40.2	68.9	55.3	64.4
DeepLab-CRF	92.1	78.4	33.1	78.2	55.6	65.3	81.3	75.5	78.6	25.3	69.2	52.7	75.2	69.0	79.1	77.6	54.7	78.3	45.1	73.3	56.2	66.4
DeepLab-MSc-CRF	92.6	80.4	36.8	77.4	55.2	66.4	81.5	77.5	78.9	27.1	68.2	52.7	74.3	69.6	79.4	79.0	56.9	78.8	45.2	72.7	59.3	67.1
DeepLab-CRF-7x7	92.8	83.9	36.6	77.5	58.4	**68.0**	84.6	**79.7**	83.1	29.5	**74.6**	59.3	78.9	76.0	82.1	80.6	**60.3**	81.7	49.2	**78.0**	60.7	70.3
DeepLab-CRF-LargeFOV	92.6	83.5	36.6	**82.5**	62.3	66.5	**85.4**	78.5	**83.7**	30.4	72.9	**60.4**	78.5	75.5	82.1	79.7	58.2	82.0	48.8	73.7	63.3	70.3
DeepLab-MSc-CRF-LargeFOV	**93.1**	**84.4**	**54.5**	81.5	**63.6**	65.9	85.1	79.1	83.4	**30.7**	74.1	59.8	**79.0**	**76.1**	**83.2**	**80.8**	59.7	**82.2**	**50.4**	73.1	**63.7**	**71.6**

Table 6.3: Labeling IOU (%) on the PASCAL VOC 2012 test set, using the trainval set for training.

while its training speed is faster. Furthermore, our best model, DeepLab-MSc-CRF-LargeFOV, attains the best performance of 71.6% by employing both multi-scale features and large FOV.

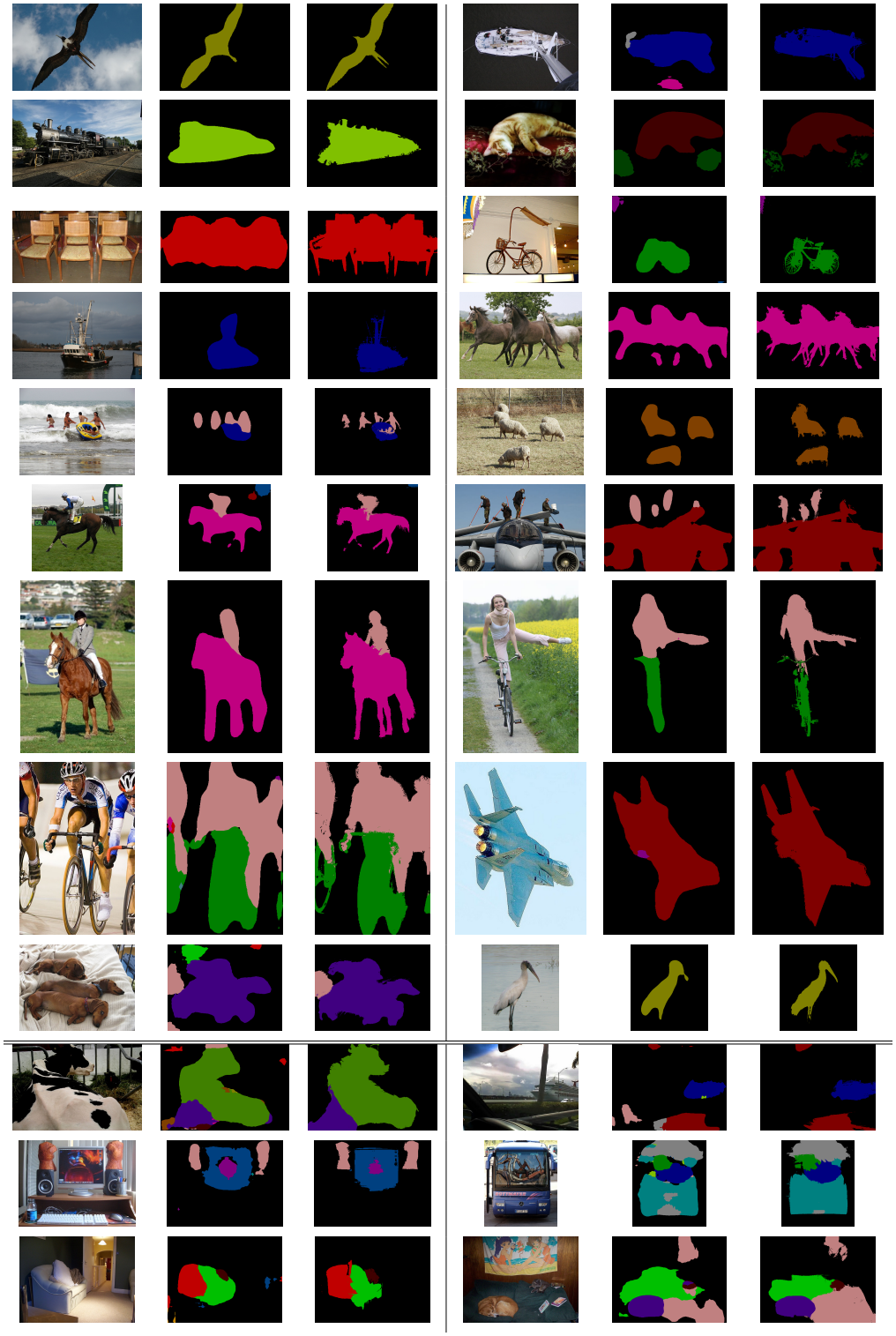


Figure 6.7: Visualization results on VOC 2012-val. For each row, we show the input image, the segmentation result delivered by the DCNN (DeepLab), and the refined segmentation result of the Fully Connected CRF (DeepLab-CRF). We show our failure modes in the last three rows. Best viewed in color.

6.4 Discussion

Our system works directly on the pixel representation, similarly to [LSD14]. This is in contrast to the two-stage approaches that are now most common in semantic segmentation with DCNNs: such techniques typically use a cascade of bottom-up image segmentation and DCNN-based region classification, which makes the system commit to potential errors of the front-end segmentation system. For instance, the bounding box proposals and masked regions delivered by [APB14, USG13] are used in [GDD14] and [HAG14b] as inputs to a DCNN to introduce shape information into the classification process. Similarly, the authors of [MYS14] rely on a superpixel representation. A celebrated non-DCNN precursor to these works is the second order pooling method of [CCB12] which also assigns labels to the regions proposals delivered by [CS12]. Understanding the perils of committing to a single segmentation, the authors of [CLP14] build on [YBS13] to explore a diverse set of CRF-based segmentation proposals, computed also by [CS12]. These segmentation proposals are then re-ranked according to a DCNN trained in particular for this reranking task. Even though this approach explicitly tries to handle the temperamental nature of a front-end segmentation algorithm, there is still no explicit exploitation of the DCNN scores in the CRF-based segmentation algorithm: the DCNN is only applied post-hoc, while it would make sense to directly try to use its results *during* segmentation.

Moving towards works that lie closer to our approach, several other researchers have considered the use of convolutionally computed DCNN features for dense image labeling. Among the first have been [FCN13] who apply DCNNs at multiple image resolutions and then employ a segmentation tree to smooth the prediction results; more recently, [HAG14a] propose to concatenate the computed inter-mediate feature maps within the DCNNs for pixel classification, and [DHS14] propose to pool the inter-mediate feature maps by region proposals. Even though these works still employ segmentation algorithms that are decoupled from the DCNN classifier's results, we believe it is advantageous that segmentation is only used at a later stage, avoiding the commitment to premature decisions.

More recently, the segmentation-free techniques of [LSD14, EF14] directly apply DCNNs to the whole image in a sliding window fashion, replacing the last fully connected layers of a DCNN

by convolutional layers. In order to deal with the spatial localization issues outlined in the beginning of the introduction, [LSD14] upsample and concatenate the scores from inter-mediate feature maps, while [EF14] refine the prediction result from coarse to fine by propagating the coarse results to another DCNN.

The main difference between our model and other state-of-the-art models is the combination of pixel-level CRFs and DCNN-based 'unary terms'. Focusing on the closest works in this direction, [CLP14] use CRFs as a proposal mechanism for a DCNN-based reranking system, while [FCN13] treat superpixels as nodes for a local pairwise CRF and use graph-cuts for discrete inference; as such their results can be limited by errors in superpixel computations, while ignoring long-range superpixel dependencies. Our approach instead treats every pixel as a CRF node, exploits long-range dependencies, and uses CRF inference to directly optimize a DCNN-driven cost function. We note that mean field had been extensively studied for traditional image segmentation/edge detection tasks, *e.g.*, [GG91, GY91, KDF08], but recently [KK11] showed that the inference can be very efficient for fully connected CRF and particularly effective in the context of semantic segmentation.

After the first version of our manuscript was made publicly available, it came to our attention that two other groups have independently and concurrently pursued a very similar direction, combining DCNNs and densely connected CRFs [BUS14, ZJR15]. There are several differences in technical aspects of the respective models. In terms of applications, [BUS14] focus on the problem of material classification. Similarly to us, [ZJR15] evaluate their system on the problem of semantic image segmentation but their results on the PASCAL VOC 2012 benchmark are somewhat inferior to ours. We refer the interested reader to these papers for different perspectives on the interplay of DCNNs and CRFs.

6.5 Conclusion

Our work combines ideas from deep convolutional neural networks and fully-connected conditional random fields, yielding a novel method able to produce semantically accurate predictions and detailed segmentation maps, while being computationally efficient. Our experimental results

show that the proposed method significantly advances the state-of-art in the challenging PASCAL VOC 2012 semantic image segmentation task.

CHAPTER 7

Weakly- and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation

Semantic image segmentation refers to the problem of assigning a semantic label (such as "person", "car" or "dog") to every pixel in the image. Various approaches have been tried over the years, but according to the results on the challenging Pascal VOC 2012 segmentation benchmark, the best performing methods all use some kind of Deep Convolutional Neural Network (DCNN) [BUS14, CPK15, DHS14, FCN13, LSD14, MYS14, ZJR15].

In this chapter, we work with the DeepLab-CRF approach of [CPK15, ZJR15]. This combines a DCNN with a fully connected Conditional Random Field (CRF) [KK11], in order to get high resolution segmentations. This model achieves state-of-art results on the challenging PASCAL VOC segmentation benchmark [EEG14], delivering a mean intersection-over-union (IOU) score exceeding 70%.

A key bottleneck in building this class of DCNN-based segmentation models is that they typically require pixel-level annotated images during training. Acquiring such data is an expensive, time-consuming annotation effort. Weak annotations, in the form of bounding boxes (*i.e.*, coarse object locations) or image-level labels (*i.e.*, information about which object classes are present) are far easier to collect than detailed pixel-level annotations. We develop new methods for training DCNN image segmentation models from weak annotations, either alone or in combination with a small number of strong annotations. Extensive experiments, in which we achieve performance up to 69.0%, demonstrate the effectiveness of the proposed techniques.

According to [LMB14], collecting bounding boxes around each class instance in the image is about 15 times faster/cheaper than labeling images at the pixel level. We demonstrate that it is

possible to learn a DeepLab-CRF model delivering 62.2% IOU on the PASCAL VOC 2012 test set by training it on a simple foreground/background segmentation of the bounding box annotations.

An even cheaper form of data to collect is image-level labels, which specify the presence or absence of semantic classes, but not the object locations. Most existing approaches for training semantic segmentation models from this kind of very weak labels use multiple instance learning (MIL) techniques. However, even recent weakly-supervised methods such as [LSD14] deliver significantly inferior results compared to their fully-supervised counterparts, only achieving 25.7%. Including additional trainable objectness [CZL14] or segmentation [APB14] modules that largely increase the system complexity, [PC15] has improved performance to 40.6%, which still significantly lags performance of fully-supervised systems.

We develop novel online Expectation-Maximization (EM) methods for training DCNN semantic segmentation models from weakly annotated data. The proposed algorithms alternate between estimating the latent pixel labels (subject to the weak annotation constraints), and optimizing the DCNN parameters using stochastic gradient descent (SGD). When we only have access to image-level annotated training data, we achieve 39.6%, close to [PC15] but without relying on any external objectness or segmentation module. More importantly, our EM approach also excels in the semi-supervised scenario which is very important in practice. Having access to a small number of strongly (pixel-level) annotated images and a large number of weakly (bounding box or image-level) annotated images, the proposed algorithm can almost match the performance of the fully-supervised system. For example, having access to 2.9k pixel-level images and 9k image-level annotated images yields 68.5%, only 2% inferior the performance of the system trained with all 12k images strongly annotated at the pixel level. Finally, we show that using additional weak or strong annotations from the MS-COCO dataset can further improve results, yielding 73.9% on the PASCAL VOC 2012 benchmark.

Contributions In summary, our main contributions are:

1. We present EM algorithms for training with image-level or bounding box annotation, applicable to both the weakly-supervised and semi-supervised settings.

2. We show that our approach achieves excellent performance when combining a small number of pixel-level annotated images with a large number of image-level or bounding box annotated images, nearly matching the results achieved when all training images have pixel-level annotations.

3. We show that combining weak or strong annotations across datasets yields further improvements. In particular, we reach 73.9% IOU performance on PASCAL VOC 2012 by combining annotations from the PASCAL and MS-COCO datasets.

7.1 Proposed Methods

We build on the DeepLab model for semantic image segmentation proposed in [CPK15]. This uses a DCNN to predict the label distribution per pixel, followed by a fully-connected (dense) CRF [KK11] to smooth the predictions while preserving image edges. In this chapter, we focus for simplicity on methods for training the DCNN parameters from weak labels, only using the CRF at test time. Additional gains can be obtained by integrated end-to-end training of the DCNN and CRF parameters [ZJR15].

Notation We denote by $\mathbf{x}$ the image values and $\mathbf{y}$ the segmentation map. In particular, $y_m \in \{0, \ldots, L\}$ is the pixel label at position $m \in \{1, \ldots, M\}$, assuming that we have the background as well as L possible foreground labels and M is the number of pixels. Note that these pixel-level labels may not be visible in the training set. We encode the set of image-level labels by $\mathbf{z}$, with $z_l = 1$, if the l-th label is present anywhere in the image, *i.e.*, if $\sum_m [y_m = l] > 0$.

7.1.1 Pixel-level annotations

In the fully supervised case illustrated in Fig. 7.1, the objective function is

$$J(\boldsymbol{\theta}) = \log P(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) = \sum_{m=1}^{M} \log P(y_m|\mathbf{x}; \boldsymbol{\theta}) \,, \tag{7.1}$$

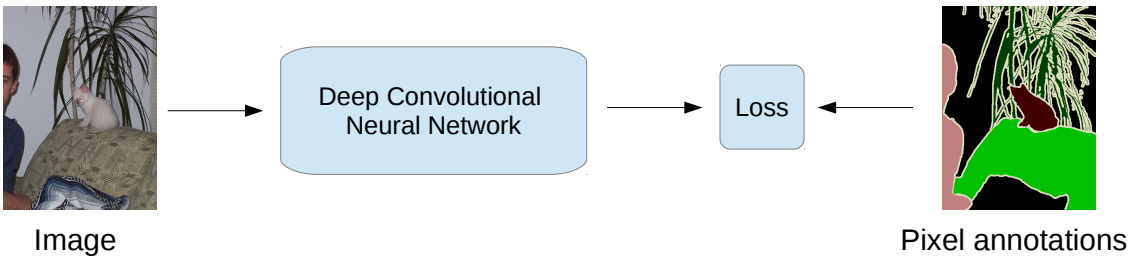


Figure 7.1: DeepLab model training from fully annotated images.

where $\boldsymbol{\theta}$ is the vector of DCNN parameters. The per-pixel label distributions are computed by

$$P(y_m|\mathbf{x};\boldsymbol{\theta}) \propto \exp(f_m(y_m|\mathbf{x};\boldsymbol{\theta}))\,, \tag{7.2}$$

where $f_m(y_m|\mathbf{x};\boldsymbol{\theta})$ is the output of the DCNN at pixel m. We optimize $J(\boldsymbol{\theta})$ by mini-batch SGD.

7.1.2 Image-level annotations

When only image-level annotation is available, we can observe the image values $\mathbf{x}$ and the image-level labels $\mathbf{z}$, but the pixel-level segmentations $\mathbf{y}$ are latent variables. We have the following probabilistic graphical model:

$$P(\mathbf{x},\mathbf{y},\mathbf{z};\boldsymbol{\theta}) = P(\mathbf{x})\left(\prod_{m=1}^{M} P(y_m|\mathbf{x};\boldsymbol{\theta})\right) P(\mathbf{z}|\mathbf{y})\,. \tag{7.3}$$

We pursue an EM-approach in order to learn the model parameters $\boldsymbol{\theta}$ from training data. If we ignore terms that do not depend on $\boldsymbol{\theta}$, the expected complete-data log-likelihood given the previous parameter estimate $\boldsymbol{\theta}'$ is

$$Q(\boldsymbol{\theta};\boldsymbol{\theta}') = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x},\mathbf{z};\boldsymbol{\theta}') \log P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) \approx \log P(\hat{\mathbf{y}}|\mathbf{x};\boldsymbol{\theta})\,, \tag{7.4}$$

where we adopt a hard-EM approximation, estimating in the E-step of the algorithm the latent segmentation by

$$\begin{aligned}
\hat{\mathbf{y}} &= \operatorname*{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}')P(\mathbf{z}|\mathbf{y}) && (7.5)\\
&= \operatorname*{argmax}_{\mathbf{y}} \log P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}') + \log P(\mathbf{z}|\mathbf{y}) && (7.6)\\
&= \operatorname*{argmax}_{\mathbf{y}} \left(\sum_{m=1}^{M} f_m(y_m|\mathbf{x};\boldsymbol{\theta}') + \log P(\mathbf{z}|\mathbf{y})\right)\,. && (7.7)
\end{aligned}$$

Algorithm 1 Weakly-Supervised EM (fixed bias version)

Input: Initial CNN parameters $\boldsymbol{\theta}'$, potential parameters b_l, $l \in \{0, \ldots, L\}$, image $\mathbf{x}$, image-level label set $\mathbf{z}$.

E-Step: For each image position m

1: $\hat{f}_m(l) = f_m(l|\mathbf{x}; \boldsymbol{\theta}') + b_l$, if $z_l = 1$

2: $\hat{f}_m(l) = f_m(l|\mathbf{x}; \boldsymbol{\theta}')$, if $z_l = 0$

3: $\hat{y}_m = \text{argmax}_l \, \hat{f}_m(l)$

M-Step:

4: $Q(\boldsymbol{\theta}; \boldsymbol{\theta}') = \log P(\hat{\mathbf{y}}|\mathbf{x}, \boldsymbol{\theta}) = \sum_{m=1}^{M} \log P(\hat{y}_m|\mathbf{x}, \boldsymbol{\theta})$

5: Compute $\nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \boldsymbol{\theta}')$ and use SGD to update $\boldsymbol{\theta}'$.

In the M-step of the algorithm, we optimize $Q(\boldsymbol{\theta}; \boldsymbol{\theta}') \approx \log P(\hat{\mathbf{y}}|\mathbf{x}; \boldsymbol{\theta})$ by mini-batch SGD similarly to (7.1), treating $\hat{\mathbf{y}}$ as ground truth segmentation.

To completely identify the E-step (7.7), we need to specify the observation model $P(\mathbf{z}|\mathbf{y})$. We have experimented with two variants, *EM-Fixed* and *EM-Adapt*.

EM-Fixed In this variant, we assume that $\log P(\mathbf{z}|\mathbf{y})$ factorizes over pixel positions as

$$\log P(\mathbf{z}|\mathbf{y}) = \sum_{m=1}^{M} \phi(y_m, \mathbf{z}) + (\text{const}) \,, \tag{7.8}$$

allowing us to estimate the E-step segmentation at each pixel separately

$$\hat{y}_m = \underset{y_m}{\text{argmax}} \, \hat{f}_m(y_m) \doteq f_m(y_m|\mathbf{x}; \boldsymbol{\theta}') + \phi(y_m, \mathbf{z}) \,. \tag{7.9}$$

We assume that

$$\phi(y_m = l, \mathbf{z}) \quad = \quad \begin{cases} b_l & \text{if } z_l = 1 \\ 0 & \text{if } z_l = 0 \end{cases} \tag{7.10}$$

We set the parameters $b_l = b_{\text{fg}}$, if $l > 0$ and $b_0 = b_{\text{bg}}$, with $b_{\text{fg}} > b_{\text{bg}} > 0$. Intuitively, this potential encourages a pixel to be assigned to one of the image-level labels $\mathbf{z}$. We choose $b_{\text{fg}} > b_{\text{bg}}$, boosting present foreground classes more than the background, to encourage full object coverage and avoid a degenerate solution of all pixels being assigned to background. The procedure is summarized in Algorithm 1 and illustrated in Fig. 7.2.

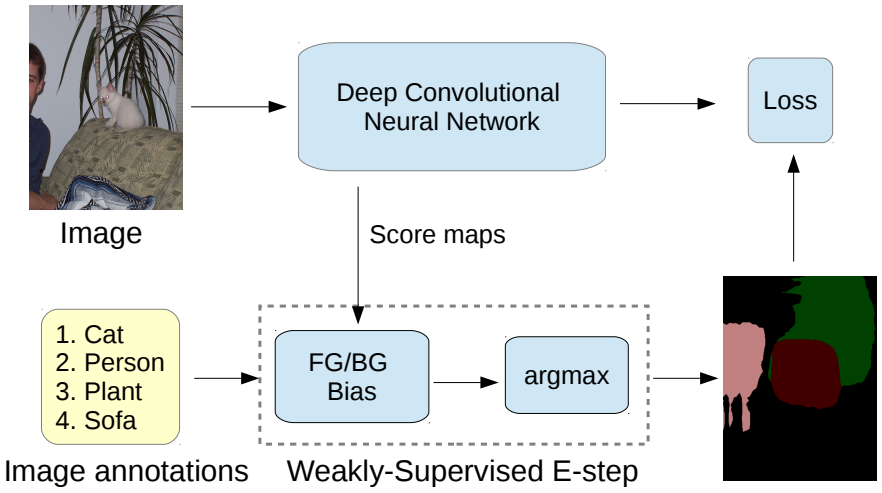


Figure 7.2: DeepLab model training using image-level labels.

EM-Adapt In this method, we assume that $\log P(\mathbf{z}|\mathbf{y}) = \phi(\mathbf{y}, \mathbf{z}) + (\text{const})$, where $\phi(\mathbf{y}, \mathbf{z})$ takes the form of a cardinality potential [LZ14, PK12, TSZ12]. In particular, we encourage *at least* a ρ_l portion of the image area to be assigned to class l, if $z_l = 1$, and enforce that no pixel is assigned to class l, if $z_l = 0$. We set the parameters $\rho_l = \rho_{\text{fg}}$, if $l > 0$ and $\rho_0 = \rho_{\text{bg}}$. Similar area or label constraints appear in [DOI12, KF05].

In practice, we employ a variant of Algorithm 1. We adaptively set the image- and class-dependent biases b_l so as the prescribed proportion of the image area is assigned to the background or foreground object classes. This acts as a powerful constraint that explicitly prevents the background score from prevailing in the whole image, also promoting higher foreground object coverage.

EM-Adapt: E-Step with Cardinality Constraints: Details of our EM-Adapt Algorithm Herein we provide more background and a detailed description of our *EM-Adapt* algorithm for weakly-supervised training with image-level annotations.

As a reminder, $\mathbf{y}$ is the latent segmentation map, with $y_m \in \{0, \dots, L\}$ denoting the label at position $m \in \{1, \dots, M\}$. The image-level annotation is encoded in $\mathbf{z}$, with $z_l = 1$, if the l-th label is present anywhere in the image.

We assume that $\log P(\mathbf{z}|\mathbf{y}) = \phi(\mathbf{y}, \mathbf{z}) + (\text{const})$. We employ a cardinality potential $\phi(\mathbf{y}, \mathbf{z})$

which encourages at least a ρ_l portion of the image area to be assigned to class l, if $z_l = 1$, and enforce that no pixel is assigned to class l, if $z_l = 0$. We set the parameters $\rho_l = \rho_{\text{fg}}$, if $l > 0$ and $\rho_0 = \rho_{\text{bg}}$.

While dedicated algorithms exist for optimizing energy functions under such cardinality potentials [TSZ12, PK12, LZ14], we opt for a simpler alternative that approximately enforces these area constraints and works well in practice. We use a variant of the *EM-Fixed* algorithm described before, updating the segmentations in the E-Step by $\hat{y}_m = \text{argmax}_l \, \hat{f}_m(l) \doteq f_m(l|\mathbf{x}; \boldsymbol{\theta}') + b_l$. The key difference in the *EM-Adapt* variant is that the biases b_l are *adaptively* set so as the prescribed proportion of the image area is assigned to the background or foreground object classes that are present in the image.

When only one label l is present (*i.e.* $z_l = 1$, $\sum_{l'=0}^{L} z_{l'} = 1$), one can easily enforce the constraint that at least ρ_l of the image area is assigned to label l as follows: (1) Set $b_{l'} = 0, l' \neq l$. (2) Compute the maximum score at each position, $f_m^{\max} = \max_{l'=0}^{L} f_m(l'|\mathbf{x}; \boldsymbol{\theta}')$. (3) Set b_l equal to the ρ_l-th percentile of the score difference $d_m = f_m^{\max} - f_m(l|\mathbf{x}; \boldsymbol{\theta}')$. The cost of this algorithm is $\mathcal{O}(M)$ (linear w.r.t. the number of pixels).

When more than one labels are present in the image (*i.e.* $\sum_{l'=0}^{L} z_{l'} > 1$), we employ the procedure above sequentially for each label that $z_l > 1$ (we first visit the background label, then in random order each of the foreground labels which are present in the image). We set $b_l = -\infty$, if $z_l = 0$, to suppress the labels that are not present in the image.

EM vs. MIL It is instructive to compare our EM-based approach with two recent Multiple Instance Learning (MIL) methods for learning semantic image segmentation models [PSL14, PC15]. The method in [PSL14] defines an MIL classification objective based on the per-class spatial maximum of the local label distributions of (7.2), $\hat{P}(l|\mathbf{x}; \boldsymbol{\theta}) \doteq \max_m P(y_m = l|\mathbf{x}; \boldsymbol{\theta})$, and [PC15] adopts a softmax function. While this approach has worked well for image classification tasks [OBL14, PKS14], it is less suited for segmentation as it does not promote full object coverage: The DCNN becomes tuned to focus on the most distinctive object parts (*e.g.*, human face) instead of capturing the whole object (*e.g.*, human body).

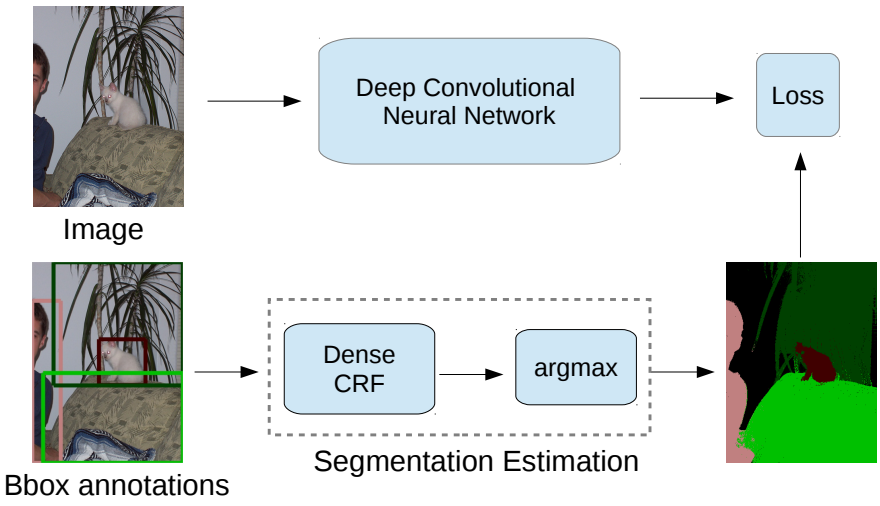


Figure 7.3: DeepLab model training from bounding boxes.

7.1.3 Bounding Box Annotations

We explore three alternative methods for training our segmentation model from labeled bounding boxes.

The first *Bbox-Rect* method amounts to simply considering each pixel within the bounding box as positive example for the respective object class. Ambiguities are resolved by assigning pixels that belong to multiple bounding boxes to the one that has the smallest area.

The bounding boxes fully surround objects but also contain background pixels that contaminate the training set with false positive examples for the respective object classes. To filter out these background pixels, we have also explored a second *Bbox-Seg* method in which we perform automatic foreground/background segmentation. To perform this segmentation, we use the same CRF as in DeepLab. More specifically, we constrain the center area of the bounding box ($\alpha\%$ of pixels within the box) to be foreground, while we constrain pixels outside the bounding box to be background. We implement this by appropriately setting the unary terms of the CRF. We then infer the labels for pixels in between. We cross-validate the CRF parameters to maximize segmentation accuracy in a small held-out set of fully-annotated images. This approach is similar to the grabcut method of [RKB04]. Examples of estimated segmentations with the two methods are shown in Fig. 7.4.

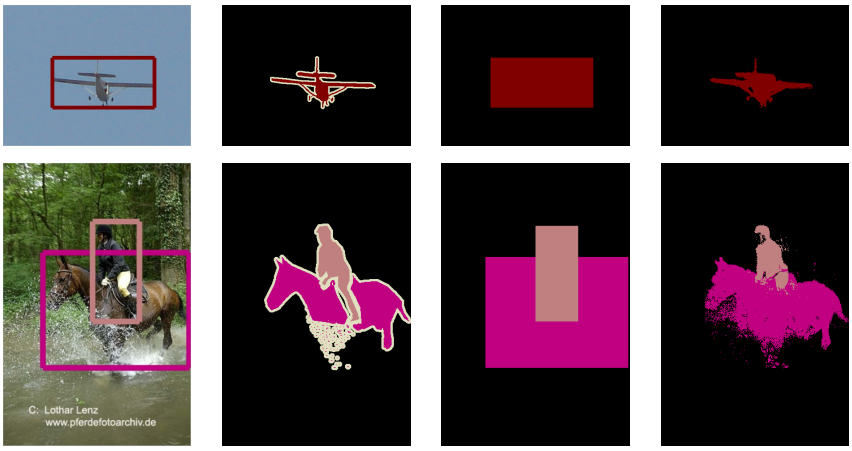


Figure 7.4: Estimated segmentation from bounding box annotation.

The two methods above, illustrated in Fig. 7.3, estimate segmentation maps from the bounding box annotation as a pre-processing step, then employ the training procedure of Sec. 7.1.1, treating these estimated labels as ground-truth.

Our third *Bbox-EM-Fixed* method is an EM algorithm that allows us to refine the estimated segmentation maps throughout training. The method is a variant of the *EM-Fixed* algorithm in Sec. 7.1.2, in which we boost the present foreground object scores only within the bounding box area.

7.1.4 Mixed strong and weak annotations

In practice, we often have access to a large number of weakly image-level annotated images and can only afford to procure detailed pixel-level annotations for a small fraction of these images. We handle this hybrid training scenario by combining the methods presented in the previous sections, as illustrated in Figure 7.5. In SGD training of our deep CNN models, we bundle to each mini-batch a fixed proportion of strongly/weakly annotated images, and employ our EM algorithm in estimating at each iteration the latent semantic segmentations for the weakly annotated images.

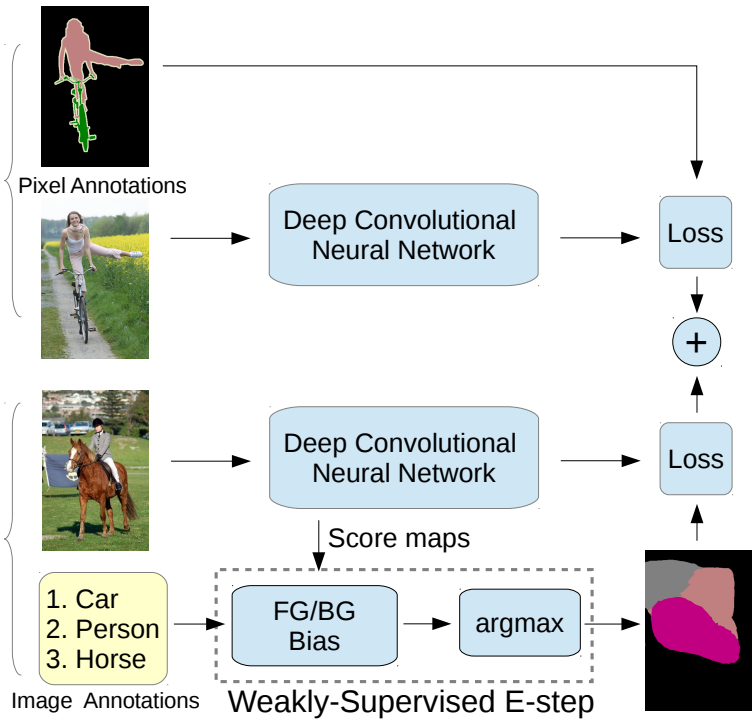


Figure 7.5: DeepLab model training on a union of full (strong labels) and image-level (weak labels) annotations.

7.2 Experimental Evaluation

7.2.1 Experimental Protocol

Datasets The proposed training methods are evaluated on the PASCAL VOC 2012 segmentation benchmark [EEG14], consisting of 20 foreground object classes and one background class. The segmentation part of the original PASCAL VOC 2012 dataset contains 1464 (*train*), 1449 (*val*), and 1456 (*test*) images for training, validation, and test, respectively. We also use the extra annotations provided by [HAB11], resulting in augmented sets of $10,582$ (*train_aug*) and $12,031$ (*trainval_aug*) images. We have also experimented with the large MS-COCO 2014 dataset [LMB14], which contains $123,287$ images in its *trainval* set. The MS-COCO 2014 dataset has 80 foreground object classes and one background class and is also annotated at the pixel level.

The performance is measured in terms of pixel intersection-over-union (IOU) averaged across the 21 classes. We first evaluate our proposed methods on the PASCAL VOC 2012 *val* set. We

then report our results on the official PASCAL VOC 2012 benchmark *test* set (whose annotations are not released). We also compare our *test* set results with other competing methods.

Reproducibility We have implemented the proposed methods by extending the excellent Caffe framework [JSD14]. We share our source code, configuration files, and trained models that allow reproducing the results in this chapter at a companion web site `https://bitbucket.org/deeplab/deeplab-public`.

Weak annotations In order to simulate the situations where only weak annotations are available and to have fair comparisons (e.g., use the same images for all settings), we generate the weak annotations from the pixel-level annotations. The image-level labels are easily generated by summarizing the pixel-level annotations, while the bounding box annotations are produced by drawing rectangles tightly containing each object instance (PASCAL VOC 2012 also provides instance-level annotations) in the dataset.

Network architectures We have experimented with the two DCNN architectures of [CPK15], with parameters initialized from the VGG-16 ImageNet pretrained model of [SZ14]. They differ in the receptive field of view (FOV) size. We have found that large FOV (224×224) performs best when at least some training images are annotated at the pixel level, whereas small FOV (128×128) performs better when only image-level annotations are available.

Training We employ our proposed training methods to learn the DCNN component of the DeepLab-CRF model of [CPK15]. In SGD training, we use a mini-batch of 20-30 images and initial learning rate of 0.001 (0.01 for the final classifier layer), multiplying the learning rate by 0.1 after a fixed number of iterations. We use momentum of 0.9 and a weight decay of 0.0005. Fine-tuning our network on PASCAL VOC 2012 takes about 12 hours on a NVIDIA Tesla K40 GPU.

Similarly to [CPK15], we decouple the DCNN and Dense CRF training stages and learn the CRF parameters by cross validation to maximize IOU segmentation accuracy in a held-out set of 100 Pascal *val* fully-annotated images. We use 10 mean-field iterations for Dense CRF inference

Method	#Strong	#Weak	val IOU
EM-Fixed (Weak)	-	10,582	20.8
EM-Adapt (Weak)	-	10,582	38.2
EM-Fixed (Semi)	200	10,382	47.6
	500	10,082	56.9
	750	9,832	59.8
	1,000	9,582	62.0
	1,464	5,000	63.2
	1,464	9,118	64.6
Strong	1,464	-	62.5
	10,582	-	67.6

Table 7.1: VOC 2012 *val* performance for varying number of pixel-level (strong) and image-level (weak) annotations (Sec. 7.2.3).

[KK11]. Note that the IOU scores are typically 3-5% worse if we don't use the CRF for post-processing of the results.

7.2.2 Pixel-level annotations

We have first reproduced the results of [CPK15]. Training the DeepLab-CRF model with strong pixel-level annotations on PASCAL VOC 2012, we achieve a mean IOU score of 67.6% on *val* and 70.3% on *test*; see method *DeepLab-CRF-LargeFOV* in [CPK15, Table 1].

7.2.3 Image-level annotations

Validation results We evaluate our proposed methods in training the DeepLab-CRF model using image-level weak annotations from the 10,582 PASCAL VOC 2012 *train_aug* set, generated as described in Sec. 7.2.1 above. We report the *val* performance of our two weakly-supervised EM variants described in Sec. 7.1.2. In the *EM-Fixed* variant we use $b_{\text{fg}} = 5$ and $b_{\text{bg}} = 3$ as fixed foreground and background biases. We found the results to be quite sensitive to the difference $b_{\text{fg}} - b_{\text{bg}}$ but not very sensitive to their absolute values. In the adaptive *EM-Adapt* variant we constrain at least $\rho_{\text{bg}} = 40\%$ of the image area to be assigned to background and at least $\rho_{\text{fg}} = 20\%$ of the image area to be assigned to foreground (as specified by the weak label set).

Method	#Strong	#Weak	test IOU
MIL-FCN [PSL14]	-	10k	25.7
MIL-sppxl [PC15]	-	760k	35.8
MIL-obj [PC15]	BING	760k	37.0
MIL-seg [PC15]	MCG	760k	40.6
EM-Adapt (Weak)	-	12k	39.6
EM-Fixed (Semi)	1.4k	10k	66.2
	2.9k	9k	68.5
Strong [CPK15]	12k	-	70.3

Table 7.2: VOC 2012 *test* performance for varying number of pixel-level (strong) and image-level (weak) annotations (Sec. 7.2.3).

We also examine using weak image-level annotations in addition to a varying number of pixel-level annotations, within the semi-supervised learning scheme of Sec. 7.1.4. In this *Semi* setting we employ strong annotations of a subset of PASCAL VOC 2012 *train* set and use the weak image-level labels from another non-overlapping subset of the *train_aug* set. We perform segmentation inference for the images that only have image-level labels by means of *EM-Fixed*, which we have found to perform better than *EM-Adapt* in the semi-supervised training setting.

The results are summarized in Table 7.1. We see that the EM-Adapt algorithm works much better than the EM-Fixed algorithm when we only have access to image level annotations, 20.8% vs. 38.2% validation IOU. Using 1,464 pixel-level and 9,118 image-level annotations in the EM-Fixed semi-supervised setting significantly improves performance, yielding 64.6%. Note that image-level annotations are helpful, as training only with the 1,464 pixel-level annotations only yields 62.5%.

Test results In Table 7.2 we report our *test* results. We compare the proposed methods with the recent MIL-based approaches of [PSL14, PC15], which also report results obtained with image-level annotations on the VOC benchmark. Our EM-Adapt method yields 39.6%, which improves over MIL-FCN [PSL14] by a large 13.9% margin. As [PC15] shows, MIL can become more competitive if additional segmentation information is introduced: Using low-level superpixels, MIL-sppxl [PC15] yields 35.8% and is still inferior to our EM algorithm. Only if augmented

with BING [CZL14] or MCG [APB14] can MIL obtain results comparable to ours (MIL-obj: 37.0%, MIL-seg: 40.6%) [PC15]. Note, however, that both BING and MCG have been trained with bounding box or pixel-annotated data on the PASCAL *train* set, and thus both MIL-obj and MIL-seg indirectly rely on bounding box or pixel-level PASCAL annotations.

The more interesting finding of this experiment is that including very few strongly annotated images in the semi-supervised setting significantly improves the performance compared to the pure weakly-supervised baseline. For example, using 2.9k pixel-level annotations along with 9k image-level annotations in the semi-supervised setting yields 68.5%. We would like to highlight that this result surpasses all techniques which are not based on the DCNN+CRF pipeline of [CPK15] (see Tab. 7.6), even if trained with all available pixel-level annotations.

7.2.4 Bounding box annotations

Validation results In this experiment, we train the DeepLab-CRF model using bounding box annotations from the *train_aug* set. We estimate the training set segmentations in a pre-processing step using the *Bbox-Rect* and *Bbox-Seg* methods described in Sec. 7.1.3. We assume that we also have access to 100 fully-annotated PASCAL VOC 2012 *val* images which we have used to cross-validate the value of the single *Bbox-Seg* parameter α (percentage of the center bounding box area constrained to be foreground). We varied α from 20% to 80%, finding that $\alpha = 20\%$ maximizes accuracy in terms of IOU in recovering the ground truth foreground from the bounding box. We also examine the effect of combining these weak bounding box annotations with strong pixel-level annotations, using the semi-supervised learning methods of Sec. 7.1.4.

The results are summarized in Table 7.3. When using only bounding box annotations, we see that *Bbox-Seg* improves over *Bbox-Rect* by 8.1%, and gets within 7.0% of the strong pixel-level annotation result. We observe that combining 1,464 strong pixel-level annotations with weak bounding box annotations yields 65.1%, only 2.5% worse than the strong pixel-level annotation result. In the semi-supervised learning settings and 1,464 strong annotations, *Semi-Bbox-EM-Fixed* and *Semi-Bbox-Seg* perform similarly.

Method	#Strong	#Box	val IOU
Bbox-Rect (Weak)	-	10,582	52.5
Bbox-EM-Fixed (Weak)	-	10,582	54.1
Bbox-Seg (Weak)	-	10,582	60.6
Bbox-Rect (Semi)	1,464	9,118	62.1
Bbox-EM-Fixed (Semi)	1,464	9,118	64.8
Bbox-Seg (Semi)	1,464	9,118	65.1
Strong	1,464	-	62.5
	10,582	-	67.6

Table 7.3: VOC 2012 *val* performance for varying number of pixel-level (strong) and bounding box (weak) annotations (Sec. 7.2.4).

Method	#Strong	#Box	test IOU
BoxSup [DHS15]	MCG	10k	64.6
BoxSup [DHS15]	1.4k (+MCG)	9k	66.2
Bbox-Rect (Weak)	-	12k	54.2
Bbox-Seg (Weak)	-	12k	62.2
Bbox-Seg (Semi)	1.4k	10k	66.6
Bbox-EM-Fixed (Semi)	1.4k	10k	66.6
Bbox-Seg (Semi)	2.9k	9k	68.0
Bbox-EM-Fixed (Semi)	2.9k	9k	69.0
Strong [CPK15]	12k	-	70.3

Table 7.4: VOC 2012 *test* performance for varying number of pixel-level (strong) and bounding box (weak) annotations (Sec. 7.2.4).

Test results In Table 7.4 we report our *test* results. We compare the proposed methods with the very recent BoxSup approach of [DHS15], which also uses bounding box annotations on the VOC 2012 segmentation benchmark. Comparing our alternative Bbox-Rect (54.2%) and Bbox-Seg (62.2%) methods, we see that simple foreground-background segmentation provides much better segmentation masks for DCNN training than using the raw bounding boxes. BoxSup does 2.4% better, however it employs the MCG segmentation proposal mechanism [APB14], which has been trained with pixel-annotated data on the PASCAL *train* set; it thus indirectly relies on pixel-level annotations.

When we also have access to pixel-level annotated images, our performance improves to 66.6%

(1.4k strong annotations) or 69.0% (2.9k strong annotations). In this semi-supervised setting we outperform BoxSup (66.6% *vs*. 66.2% with 1.4k strong annotations), although we do not use MCG. Interestingly, Bbox-EM-Fixed improves over Bbox-Seg as we add more strong annotations, and it performs 1.0% better (69.0% *vs*. 68.0%) with 2.9k strong annotations. This shows that the E-step of our EM algorithm can estimate the object masks better than the foreground-background segmentation pre-processing step when enough pixel-level annotated images are available.

Comparing with Sec. 7.2.3, note that 2.9k strong + 9k image-level annotations yield 68.5% (Tab. 7.2), while 2.9k strong + 9k bounding box annotations yield 69.0% (Tab. 7.3). This finding suggests that bounding box annotations add little value over image-level annotations when a sufficient number of pixel-level annotations is also available.

7.2.5 Exploiting Annotations Across Datasets

Validation results We present experiments leveraging the 81-label MS-COCO dataset as an additional source of data in learning the DeepLab model for the 21-label PASCAL VOC 2012 segmentation task. We consider three scenarios:

- *Cross-Pretrain (Strong)*: Pre-train DeepLab on MS-COCO, then replace the top-level network weights and fine-tune on Pascal VOC 2012, using pixel-level annotation in both datasets.
- *Cross-Joint (Strong)*: Jointly train DeepLab on Pascal VOC 2012 and MS-COCO, sharing the top-level network weights for the common classes, using pixel-level annotation in both datasets.
- *Cross-Joint (Semi)*: Jointly train DeepLab on Pascal VOC 2012 and MS-COCO, sharing the top-level network weights for the common classes, using the pixel-level labels from PASCAL and varying the number of pixel- and image-level labels from MS-COCO.

In all cases we use strong pixel-level annotations for all 10,582 *train_aug* PASCAL images.

We report our results on the PASCAL VOC 2012 *val* in Tab. 7.5, also including for comparison our best PASCAL-only 67.6% result exploiting all 10,582 strong annotations as a baseline. When we employ the weak MS-COCO annotations (*EM-Fixed (Semi)*) we obtain 67.7% IOU, which

Method	#Strong COCO	#Weak COCO	val IOU
PASCAL-only	-	-	67.6
EM-Fixed (Semi)	-	123,287	67.7
Cross-Joint (Semi)	5,000	118,287	70.0
Cross-Joint (Strong)	5,000	-	68.7
Cross-Pretrain (Strong)	123,287	-	71.0
Cross-Joint (Strong)	123,287	-	71.7

Table 7.5: VOC 2012 *val* performance using strong annotations for all 10,582 *train_aug* PASCAL images and a varying number of strong and weak MS-COCO annotations (Sec. 7.2.5).

Method	test IOU
MSRA-CFM [DHS14]	61.8
FCN-8s [LSD14]	62.2
Hypercolumn [HAG14a]	62.6
TTI-Zoomout-16 [MYS14]	64.4
DeepLab-CRF-LargeFOV [CPK15]	70.3
BoxSup (Semi, with weak COCO) [DHS15]	71.0
DeepLab-CRF-LargeFOV (Multi-scale net) [CPK15]	71.6
Oxford_TVG_CRF_RNN_VOC [ZJR15]	72.0
Oxford_TVG_CRF_RNN_COCO [ZJR15]	74.7
Cross-Pretrain (Strong)	72.7
Cross-Joint (Strong)	73.0
Cross-Pretrain (Strong, Multi-scale net)	73.6
Cross-Joint (Strong, Multi-scale net)	73.9

Table 7.6: VOC 2012 *test* performance using PASCAL and MS-COCO annotations (Sec. 7.2.5).

does not improve over the PASCAL-only baseline. However, using strong labels from 5,000 MS-COCO images (4.0% of the MS-COCO dataset) and weak labels from the remaining MS-COCO images in the *Cross-Joint (Semi)* semi-supervised scenario yields 70.0%, a significant 2.4% boost over the baseline. This *Cross-Joint (Semi)* result is also 1.3% better than the 68.7% performance obtained using only the 5,000 strong and no weak annotations from MS-COCO. As expected, our best results are obtained by using all 123,287 strong MS-COCO annotations, 71.0% for *Cross-Pretrain (Strong)* and 71.7% for *Cross-Joint (Strong)*. We observe that cross-dataset augmentation improves by 4.1% over the best PASCAL-only result. Using only a small portion of pixel-level annotations and a large portion of image-level annotations in the semi-supervised setting reaps

about half of this benefit.

Test results We report our PASCAL VOC 2012 *test* results in Tab. 7.6. We include results of other leading models from the PASCAL leaderboard. All our models have been trained with pixel-level annotated images on the PASCAL *trainval_aug* and the MS-COCO 2014 *trainval* datasets.

Methods based on the DCNN+CRF pipeline of DeepLab-CRF [CPK15] are the most competitive, with performance surpassing 70%, even when only trained on PASCAL data. Leveraging the MS-COCO annotations brings about 2% improvement. Our top model yields 73.9%, using the multi-scale network architecture of [CPK15]. This is the second best result reported on the PASCAL VOC 2012 segmentation task, only surpassed by [ZJR15], which also use joint training on PASCAL and MS-COCO data, but get improved performance (74.7%) due to end-to-end learning of the DCNN and CRF parameters.

7.2.6 Qualitative Segmentation Results

In Fig. 7.6 we provide visual comparisons of the results obtained by the DeepLab-CRF model learned with some of the proposed training methods.

7.2.7 Effect of Field-Of-View

In this section, we explore the effect of Field-Of-View (FOV) when training the DeepLab-CRF model with the proposed methods. Similar to [CPK15], we also employ the 'atrous' algorithm [Mal99] in the DeepLab model. The 'atrous' algorithm enables us to arbitrary control the model's FOV by adjusting the input stride (which is equivalent to injecting zeros between filter weights) at the first fully connected layer of VGG-16 net [SZ14]. Applying a large value of input stride increases the effective kernel size, and thus enlarges the model's FOV (see [CPK15] for details).

Experimental protocol We employ the same experimental protocol as mentioned before. Models trained with the proposed training methods and different values of FOV are evaluated on the PASCAL VOC 2012 *val* set.

EM-Adapt Assuming only image-level labels are available, we first experiment with the *EM-Adapt (Weak)* method when the value of FOV varies. Specifically, we explore the setting where the kernel size is 3×3 with various FOV values. The selection of kernel size 3×3 is based on the discovery by [CPK15]: employing a kernel size of 3×3 at the first fully connected layer can attain the same performance as using the kernel size of 7×7, while being 3.4 times faster during training. As shown in Tab. 7.7, we find that our proposed model can yield the performance of 39.2% with FOV 96×96, but the performance degrades by 9% when large FOV 224×224 is employed. The original DeepLab model employed by [CPK15] has a kernel size of 4×4 and input stride of 4. Its performance, shown in the first row of Tab. 7.7, is similar to the performance obtained by using a kernel size of 3×3 and input stride of 6. Both cases have the same FOV value of 128×128.

Network architectures In the following experiments, we compare two network architectures trained with the proposed methods. The first network is the same as the one originally employed by [CPK15] (kernel size 4×4 and input stride 4, resulting in a FOV size 128×128). The second network we employ has FOV 224×224 (with kernel size of 3×3 and an input stride of 12). We refer to the first network as 'DeepLab-CRF with small FOV', and the second network as 'DeepLab-CRF with large FOV'.

Image-level annotations In Tab. 7.8, we experiment with the cases where weak image-level annotations as well as a varying number of pixel-level annotations are available. Similar to the results in Tab. 7.7, the DeepLab-CRF with small FOV performs better than that with large FOV when a small amount of supervision is leveraged. Interestingly, when there are more than 750 pixel-level annotations are available in the semi-supervised setting, employing large FOV yields better performance than using small FOV.

Bounding box annotations In Tab. 7.9, we report the results when weak bounding box annotations in addition to a varying number of pixel-level annotations are exploited. we found that DeepLab-CRF with small FOV attains better performance when trained with the three methods: Bbox-Rect (Weak), Bbox-EM-Fixed (Weak), and Bbox-Rect (Semi- 1464 strong), whereas the

kernel size	input stride	receptive field	val IOU (%)
4×4	4	128×128	38.2
3×3	2	64×64	37.3
3×3	4	96×96	39.2
3×3	6	128×128	38.3
3×3	8	160×160	38.1
3×3	10	192×192	32.6
3×3	12	224×224	30.2

Table 7.7: Effect of Field-Of-View. The validation performance obtained by DeepLab-CRF trained with the method EM-Adapt (Weak) as the value of FOV varies.

model DeepLab-CRF with large FOV is better in all the other cases.

Annotations across datasets In Tab. 7.10, we show the results when training the models with the strong pixel-level annotations from PASCAL VOC 2012 *train_aug* set in conjunction with the extra annotations from MS-COCO [LMB14] dataset (in the form of either weak image-level annotations or strong pixel-level annotations). Interestingly, employing large FOV consistently improves over using small FOV in all cases by at least 3%.

Reported results Note that we report the results of the best architecture for each setup, in Sec. 7.2.3, Sec. 7.2.4, and Sec. 7.2.5.

7.2.8 Detailed test results

In Tab. 7.11, Tab. 7.12, and Tab. 7.13, we show more detailed results on PASCAL VOC 2012 *test* set for all the reported methods in this chapter.

7.3 Discussion

Training segmentation models with only image-level labels has been a challenging problem in the literature [DBF02, VT07, VFB12, XSU14]. Our work is most related to other recent DCNN models such as [PSL14, PC15], who also study the weakly supervised setting. They both de-

Method	#Strong	#Weak	w Small FOV	w Large FOV
EM-Fixed (Weak)	-	10,582	20.8	19.9
EM-Adapt (Weak)	-	10,582	38.2	30.2
EM-Fixed (Semi)	200	10,382	47.6	38.9
	500	10,082	56.9	54.2
	750	9,832	58.8	59.8
	1,000	9,582	60.5	62.0
	1,464	5,000	60.5	63.2
	1,464	9,118	61.9	64.6
Strong	1,464	-	57.6	62.5
	10,582	-	63.9	67.6

Table 7.8: Effect of Field-Of-View. VOC 2012 *val* performance for varying number of pixel-level (strong) and image-level (weak) annotations.

Method	#Strong	#Box	w Small FOV	w Large FOV
Bbox-Rect (Weak)	-	10,582	52.5	50.7
Bbox-EM-Fixed (Weak)	-	10,582	54.1	50.2
Bbox-Seg (Weak)	-	10,582	58.5	60.6
Bbox-Rect (Semi)	1,464	9.118	62.1	61.1
Bbox-EM-Fixed (Semi)	1,464	9,118	59.6	64.8
Bbox-Seg (Semi)	1,464	9,118	61.8	65.1
Strong	1,464	-	57.6	62.5
Strong	10,582	-	63.9	67.6

Table 7.9: Effect of Field-Of-View. VOC 2012 *val* performance for varying number of pixel-level (strong) and bounding box (weak) annotations.

Method	#Strong	#Weak	w Small FOV	w Large FOV
PASCAL-only	-	-	63.9	67.6
EM-Fixed (Semi)	-	123,287	64.4	67.7
Cross-Joint (Semi)	5,000	118,287	66.5	70.0
Cross-Joint (Strong)	5,000	-	64.9	68.7
Cross-Pretrain (Strong)	123,287	-	68.0	71.0
Cross-Joint (Strong)	123,287	-	68.0	71.7

Table 7.10: Effect of Field-Of-View. VOC 2012 *val* performance using strong annotations for all 10,582 *train_aug* PASCAL images and a varying number of strong and weak MS-COCO annotations.

Method	bkg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
MIL-FCN [PSL14]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	25.7
MIL-sppxl [PC15]	74.7	38.8	19.8	27.5	21.7	32.8	40.0	50.1	47.1	7.2	44.8	15.8	49.4	47.3	36.6	36.4	24.3	44.5	21.0	31.5	41.3	35.8
MIL-obj [PC15]	76.2	42.8	20.9	29.6	25.9	38.5	40.6	51.7	49.0	9.1	43.5	16.2	50.1	46.0	35.8	38.0	22.1	44.5	22.4	30.8	43.0	37.0
MIL-seg [PC15]	78.7	48.0	21.2	31.1	28.4	35.1	51.4	55.5	52.8	7.8	56.2	19.9	53.8	50.3	40.0	38.6	27.8	51.8	24.7	33.3	46.3	40.6
EM-Adapt (Weak)	76.3	37.1	21.9	41.6	26.1	38.5	50.8	44.9	48.9	16.7	40.8	29.4	47.1	45.8	54.8	28.2	30.0	44.0	29.2	34.3	46.0	39.6
EM-Fixed (Semi-1464 strong)	91.3	78.9	37.3	**81.4**	57.1	57.7	83.5	77.5	**77.6**	22.5	70.3	56.1	72.2	**74.3**	80.7	72.4	42.0	81.3	43.1	72.5	60.7	66.2
EM-Fixed (Semi-2913 strong)	**92.0**	**81.6**	**42.9**	80.5	**59.2**	**60.8**	**85.5**	**78.7**	77.3	**26.9**	**75.2**	**57.6**	**74.0**	74.2	**82.1**	**73.1**	**52.4**	**84.3**	**43.8**	**75.1**	**61.9**	**68.5**

Table 7.11: VOC 2012 *test* performance for varying number of pixel-level (strong) and image-level (weak) annotations. Links to the PASCAL evaluation server are included in the PDF.

Method	bkg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
BoxSup-box [DHS15]	-	80.3	31.3	82.1	47.4	62.6	75.4	75.0	74.5	24.5	68.3	56.4	73.7	69.4	75.2	75.1	47.4	70.8	45.7	71.1	58.8	64.6
BoxSup-semi [DHS15]	-	**82.0**	33.6	74.0	55.8	57.5	81.0	74.6	80.7	27.6	70.9	50.4	71.6	70.8	78.2	76.9	53.5	72.6	**50.1**	72.3	**64.4**	66.2
Bbox-Rect (Weak)	82.9	43.6	22.5	50.5	45.0	62.5	76.0	66.5	61.2	25.3	55.8	52.1	56.6	48.1	60.1	58.2	49.5	58.3	40.7	62.3	61.1	54.2
Bbox-Seg (Weak)	89.2	64.4	27.3	67.6	55.1	64.0	81.6	70.5	76.0	24.1	63.8	**58.2**	72.1	59.8	73.5	71.4	47.4	76.0	44.2	68.9	50.9	62.2
Bbox-Seg (Semi-1464 strong)	91.3	75.3	29.9	74.4	59.8	64.6	84.3	76.2	79.0	27.9	69.1	56.5	73.8	66.7	78.8	76.0	51.8	80.8	47.5	73.6	60.5	66.6
Bbox-EM-Fixed (Semi-1464 strong)	91.9	78.3	36.5	**86.2**	53.8	62.5	81.2	**80.0**	83.2	22.8	68.9	46.7	78.1	72.0	82.2	78.5	44.5	81.1	36.4	74.6	60.2	66.6
Bbox-Seg (Semi-2913 strong)	92.0	76.4	34.1	79.2	**61.0**	**65.6**	**85.0**	76.9	81.5	**28.5**	69.3	58.0	75.5	69.8	79.3	76.9	**54.0**	81.4	46.9	73.6	62.9	68.0
Bbox-EM-Fixed (Semi-2913 strong)	**92.5**	80.4	**41.6**	84.6	59.0	64.7	84.6	79.6	**83.5**	26.3	**71.2**	52.9	**78.3**	**72.3**	**83.3**	**79.1**	51.7	**82.1**	42.5	**75.0**	63.4	**69.0**

Table 7.12: VOC 2012 *test* performance for varying number of pixel-level (strong) and bounding box (weak) annotations. Links to the PASCAL evaluation server are included in the PDF.

Method	bkg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
MSRA-CFM [DHS14]	-	75.7	26.7	69.5	48.8	65.6	81.0	69.2	73.3	30.0	68.7	51.5	69.1	68.1	71.7	67.5	50.4	66.5	44.4	58.9	53.5	61.8
FCN-8s [LSD14]	-	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
Hypercolumn [HAG14a]	-	68.7	33.5	69.8	51.3	**70.2**	81.1	71.9	74.9	23.9	60.6	46.9	72.1	68.3	74.5	72.9	52.6	64.4	45.4	64.9	57.4	62.6
TTI-Zoomout-16 [MYS14]	89.8	81.9	35.1	78.2	57.4	56.5	80.5	74.0	79.8	22.4	69.6	53.7	74.0	76.0	76.6	68.8	44.3	70.2	40.2	68.9	55.3	64.4
CRF_RNN [ZJR15]	-	80.9	34.0	72.9	52.6	62.5	79.8	76.3	79.9	23.6	67.7	51.8	74.8	69.9	76.9	76.9	49.0	74.7	42.7	72.1	59.6	65.2
DeepLab-CRF-LargeFOV [CPK15]	92.6	83.5	36.6	82.5	62.3	66.5	85.4	78.5	83.7	30.4	72.9	60.4	78.5	75.5	82.1	79.7	58.2	82.0	48.8	73.7	63.3	70.3
Oxford_TVG_CRF_RNN [ZJR15]	-	85.5	36.7	77.2	62.9	66.7	85.9	78.1	82.5	30.1	74.8	59.2	77.3	75.0	82.8	79.7	**59.8**	78.3	50.0	76.9	65.7	70.4
BoxSup-semi-coco [DHS15]	-	86.4	35.5	79.7	65.2	65.2	84.3	78.5	83.7	30.5	76.2	62.6	79.3	76.1	82.1	81.3	57.0	78.2	55.0	72.5	68.1	71.0
DeepLab-MSc-CRF-LargeFOV [CPK15]	93.1	84.4	54.5	81.5	63.6	65.9	85.1	79.1	83.4	30.7	74.1	59.8	79.0	76.1	83.2	80.8	59.7	82.2	50.4	73.1	63.7	71.6
Oxford_TVG_CRF_RNN_COCO [ZJR15]	-	**90.4**	**55.3**	**88.7**	**68.4**	69.8	**88.3**	82.4	85.1	**32.6**	78.5	**64.4**	79.6	**81.9**	**86.4**	81.8	58.6	82.4	53.5	77.4	**70.1**	**74.7**
Cross-Pretrain (Strong)	93.4	89.1	38.3	88.1	63.3	69.7	87.1	**83.1**	85.0	29.3	76.5	56.5	79.8	77.9	85.8	82.4	57.4	84.3	54.9	**80.5**	64.1	72.7
Cross-Joint (Strong)	93.3	88.5	35.9	88.5	62.3	68.0	87.0	81.0	**86.8**	32.2	80.8	60.4	81.1	81.1	83.5	81.7	55.1	84.6	57.2	75.7	67.2	73.0
Cross-Pretrain (Strong, Multi-scale net)	**93.8**	88.7	53.1	87.7	64.4	69.5	85.9	81.6	85.3	31.0	76.4	62.0	79.8	77.3	84.6	**83.2**	59.1	**85.5**	55.9	76.5	64.3	73.6
Cross-Joint (Strong, Multi-scale net)	93.7	89.2	46.7	88.5	63.5	68.4	87.0	81.2	86.3	**32.6**	80.7	62.4	81.0	81.3	84.3	82.1	56.2	84.6	**58.3**	76.2	67.2	73.9

Table 7.13: VOC 2012 *test* performance using strong PASCAL and strong MS-COCO annotations. Links to the PASCAL evaluation server are included in the PDF.

velop MIL-based algorithms for the problem. In contrast, our model employs an EM algorithm, which similarly to [LTL13] takes into account the weak labels when inferring the latent image segmentations. Moreover, [PC15] proposed to smooth the prediction results by region proposal

algorithms (e.g., MCG [APB14]), learned on pixel-segmented images. Neither [PSL14, PC15] cover the semi-supervised setting.

Bounding box annotations have been utilized for semantic segmentation by [XDD13, ZMY14], while [GKF14] describes a scheme exploiting both image-level labels and bounding box annotations in ImageNet [DDS09]. [CFY14] attained human-level accuracy for car segmentation by using 3D bounding boxes. Bounding box annotations are also commonly used in interactive segmentation [LKR09, RKB04]; we show that such foreground/background segmentation methods can effectively estimate object segments accurate enough for training a DCNN semantic segmentation system. Working in a setting very similar to ours, [DHS15] employed MCG [APB14] (which requires training from pixel-level annotations) to infer object masks from bounding box labels during DCNN training.

7.4 Conclusion

This chapter has explored the use of weak or partial annotation in training a state of art semantic image segmentation model. Extensive experiments on the challenging PASCAL VOC 2012 dataset have shown that: (1) Using weak annotation solely at the image-level seems insufficient to train a high-quality segmentation model. (2) Using weak bounding-box annotation in conjunction with careful segmentation inference for images in the training set suffices to train a competitive model. (3) Excellent performance is obtained when combining a small number of pixel-level annotated images with a large number of weakly annotated images in a semi-supervised setting, nearly matching the results achieved when all training images have pixel-level annotations. (4) Exploiting extra weak or strong annotations from other datasets can lead to large improvements.

Figure 7.6: Qualitative DeepLab-CRF segmentation results on the PASCAL VOC 2012 *val* set. The last two rows show failure modes.

CHAPTER 8

Future Directions

Scene understanding remains an unsolved problem in computer vision. One of the key issues is how to learn good image representations. Hand-crafted features, such as HOG and SIFT, are frequently used before, while recently deep convolutional neural networks are employed to jointly learn features and learn classifiers for the tasks of interest. In order to find better representation for scene understanding, in this thesis, we have explored along three directions: (1) we propose to learn a dictionary of shape epitomes or appearance epitomes, as a compact generative representation for modeling object co-relation within edge patches and for explicitly encoding photometric and position variability of image patches, respectively, (2) we apply CRF to encode structured output correlation (*i.e.*, neighboring pixels should have high probability to be assigned the same labels) for semantic segmentation, and (3) we combine the powerful prediction ability of DCNN and the structured representation of CRF and obtain state-of-art performance on semantic segmentation benchmarks. However, there are still many challenging problems in scene understanding, and we conclude with a brief overview of possible directions for future research in this chapter.

Epitomes Currently, our proposed dictionary of epitomes mainly encode positional variability. However, objects may appear in the images with different scales. For example, there could be both cars very close to the camera (thus, they seem to be very large), and cars very far away from the camera. Besides, rotation-invariant could be another important factor to consider, in order to learn a more compact dictionary, since a vertical edge is only a rotated horizontal edge. To enrich the representational power of our dictionary of epitomes, we could consider scale and rotation during the unsupervised learning. Furthermore, it is also very interesting to combine the idea of epitomes and deep learning, as [PKS14], which propose an epitomic convolutional layer in the replacement

of one convolutional layer and one max-pooling layer. They have shown exciting results, such as faster convergence and better performance for image classification task.

Joint training of CRF and DCNN In our works, we combine both DCNN and fully connected CRF for semantic segmentation. We mainly employ a piecewise training strategy: the fully connected CRF is used as a post processing step for the DCNN output. The piecewise training strategy is not optimal, as it fails to consider the interaction between DCNN and CRF. The joint model should focus on the training samples that cannot be corrected by both CRF and DCNN, and jointly fine-tune whole model for better performance. On the other hand, [ZJR15] have shown that we can express every operation in the mean-field theory (used by fully connected CRF) as a layer in the neural network context, which enables joint training of both components (*i.e.*, fully connected CRF and DCNN). They mainly focus on the specific format of CRF, whose pairwise term employs Gaussian kernels. Gaussian kernels are used because faster bilateral filtering [ABD10] tricks can be applied for fast message passing. However, there is no reason to constrain ourselves for Gaussian kernels as pairwise term. It is interesting to explore other types of pairwise terms, such as general spatial pairwise term [LLL15]. For example, another DCNN may be used to learn the pairwise term [LSR15].

Instance Segmentation Our proposed model for semantic segmentation treats instances of the same semantic label similarly. That is, we do not try to distinguish there are several persons in the image, and only where are they presented. This result is not descriptive for better scene understanding for several practical applications. For example, for autonomous cars, we need to know how many cars or pedestrians are near the car and how can we drive to avoid any possible accident. This kind of problem is referred to as instance segmentation in the literature, and it has begun to catch intense attention recently. One possible direction for this problem is to combine detection and segmentation. Detection module will provide the coarse position of each instance, while segmentation module will provide the detailed boundary of the object. Current methods [HAG14b] employ a two-step processes, where segmentation is performed on top of the detection results. However, this is not optimal, since the final performance heavily depends on the accuracy

of the first detection module, since the errors cannot be recovered back easily in this two-step processes. It is better to consider detection and segmentation jointly, similar to [ZUS15].

RGB-D images RGB-D images provide richer information for scene understanding. From one of our works, we have shown that we can reach human-level car segmentation given 3D CAD model and depth information. [GGA14] have shown that better feature representations with DCNN can be learned with geometric embedding (an encoding for depth images for height above the ground and angle with gravity for each pixel in addition to the horizontal disparity). They have demonstrated their work on indoor instance segmentation. However, their proposed model is not trained end-to-end. That is, a Support Vector Machine is trained on top of the features extracted by DCNN. It will be very interesting if we could train the whole thing end-to-end within on deep neural network.

Multi-tasks It has been known that several scene understanding tasks are beneficial to each other. For example, detection is could be helpful for segmentation as a top-down information [LSA10], and image classification provides the global contextual information for segmentation [GBW10]. One way to handle multi-tasks is perform sequential processing, in which the following modules depend on the previous ones. However, in the context of deep learning, it is interesting to simply use one model for several correlated tasks, since it has been shown that the correlated tasks can share feature representations in the lower layers [CWB11]. Developing a shared neural network for several correlated scene understanding tasks is both interesting and challenging. For example, combining detection and segmentation as a multi-task problem could be one step toward solving instance segmentation.

REFERENCES

[ABD10] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. "Fast High-Dimensional Filtering Using the Permutohedral Lattice." In *Computer Graphics Forum*, 2010. 79, 115

[AE08] M. Aharon and M. Elad. "Sparse and redundant modeling of image content using an image-signature-dictionary." *SIAM Journal on Imaging Sciences*, **1**(3):228–247, 2008. 26

[AEB06] M. Aharon, M. Elad, and A. Bruckstein. "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation." *SP*, **54**(11):4311–4322, November 2006. 24

[AMF11] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. "Contour detection and hierarchical image segmentation." *PAMI*, 2011. 10, 11, 16

[APB14] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. "Multiscale Combinatorial Grouping." In *CVPR*, 2014. 87, 91, 103, 104, 112

[AV07] D. Arthur and S. Vassilvitskii. "K-means++: The advantages of careful seeding." In *SODA*, pp. 1027–1035, 2007. 31

[BBL97] L. Bottou, Y. Bengio, and Y. LeCun. "Global training of document processing systems using graph transformer networks." In *CVPR*, 1997. 70

[BJ01] Yuri Boykov and M-P Jolly. "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images." In *ICCV*, 2001. 42, 45

[BK04] Yuri Boykov and Vladimir Kolmogorov. "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *PAMI*, **26**(9):1124–1137, 2004. 42, 48

[BLP07] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. "Greedy Layer-Wise Training of Deep Networks." In *NIPS*, 2007. 56

[BMB11] L. Benoît, J. Mairal, F. Bach, and J. Ponce. "Sparse image representation with epitomes." In *CVPR*, pp. 2913–2920, 2011. 26

[BRF11] L. Bo, X. Ren, and D. Fox. "Hierarchical matching pursuit for image classification." In *NIPS*, 2011. 26

[Bri90] J. S. Bridle. "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters." In *NIPS*, 1990. 70

[BS97] A.J. Bell and T.J. Sejnowski. "The 'independent components' of natural scenes are edge filters." *Vision Research*, **37**:3327–3338, 1997. 26

[BSF08] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. "Segmentation and recognition using structure from motion point clouds." In *ECCV*, 2008. 43

[BUS14] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. "Material recognition in the wild with the Materials in Context Database." *arXiv:1412.0623*, 2014. 88, 90

[CBV09] T. E. de Campos, B. R. Babu, and M. Varma. "Character recognition in natural images." In *VISAPP*, 2009. 65

[CCB12] João Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. "Semantic segmentation with second-order pooling." In *ECCV*, 2012. 87

[CFY14] Liang-Chieh Chen, Sanja Fidler, Alan L. Yuille, and Raquel Urtasun. "Beat the mturkers: Automatic image labeling from weak 3d supervision." In *CVPR*, 2014. 112

[CLN11] A. Coates, H. Lee, and A. Ng. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning." In *AISTATS*, 2011. 25, 31, 36

[CLP14] Michael Cogswell, Xiao Lin, Senthil Purushwalkam, and Dhruv Batra. "Combining the Best of Graphical Models and ConvNets for Semantic Segmentation." *arXiv:1412.4313*, 2014. 87, 88

[CLV11] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. "The devil is in the details: an evaluation of recent feature encoding methods." *BMVC*, 2011. 15, 25, 26, 35, 36, 39

[Col02] M. Collins. "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms." *ACL*, 2002. 18

[CPK15] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs." In *ICLR*, 2015. 71, 90, 92, 100, 101, 102, 103, 104, 106, 107, 108, 111

[CPY13] L.-C. Chen, G. Papandreou, and A. Yuille. "Learning a Dictionary of Shape Epitomes with Applications to Image Labeling." In *ICCV*, 2013. 74

[CS12] Jo ao Carreira and Cristian Sminchisescu. "CPMC: Automatic object segmentation using constrained parametric min-cuts." *PAMI*, 2012. 87

[CWB11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. "Natural language processing (almost) from scratch." *JMLR*, 2011. 71, 116

[CY14] Xianjie Chen and Alan L Yuille. "Articulated pose estimation by a graphical model with image dependent pairwise relations." In *NIPS*, 2014. 73

[CZL14] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip H. S. Torr. "BING: Binarized Normed Gradients for Objectness Estimation at 300fps." In *CVPR*, 2014. 91, 103

[DA10] T.-M.-T. Do and T. Artieres. "Neural conditional random fields." In *AISTATS*, 2010. 71

[DBF02] Pinar Duygulu, Kobus Barnard, Joao FG de Freitas, and David A Forsyth. "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary." In *ECCV*, 2002. 109

[DDJ14] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. "Large-Scale Object Classification using Label Relation Graphs." In *ECCV*, 2014. 56

[DDS09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *CVPR*, 2009. 39, 40, 70, 112

[DHH12] M. Dikmen, D. Hoiem, and T. Huang. "A data driven method for feature transformation." In *CVPR*, 2012. 26

[DHS14] Jifeng Dai, Kaiming He, and Jian Sun. "Convolutional Feature Masking for Joint Object and Stuff Segmentation." *arXiv:1412.1283*, 2014. 84, 87, 90, 106, 111

[DHS15] Jifeng Dai, Kaiming He, and Jian Sun. "BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation." *arXiv:1503.01640*, 2015. 104, 106, 111, 112

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society*, **39**(1):1–38, 1977. 30

[DOI12] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. "Fast approximate energy minimization with label costs." *IJCV*, 2012. 10, 16, 74, 95

[Dom13] J. Domke. "Structured Learning via Logistic Regression." In *NIPS*, 2013. 71

[DT05] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection." In *CVPR*, volume 1, pp. 886–893. IEEE, 2005. 1

[DT06] James Diebel and Sebastian Thrun. "An application of markov random fields to range sensing." In *NIPS*, 2006. 44

[EEG14] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserma. "The Pascal Visual Object Classes Challenge a Retrospective." *IJCV*, 2014. 80, 90, 99

[EF01] A. Efros and W. Freeman. "Image quilting for texture synthesis and transfer." In *SIGGRAPH*, pp. 341–346, 2001. 24

[EF14] David Eigen and Rob Fergus. "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture." *arXiv:1411.4734*, 2014. 77, 87, 88

[ERF14] D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. "Understanding Deep Architectures using a Recursive Convolutional Network." In *ICLR*, 2014. 56

[EVW10] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. "The PASCAL visual object classes challenge." *IJCV*, **88**(2):303–338, 2010. 25, 34, 40

[FCN13] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. "Learning hierarchical features for scene labeling." *PAMI*, 2013. 87, 88, 90

[FD07] B.J. Frey and D. Dueck. "Clustering by passing messages between data points." *Science*, 2007. 10

[FDU12] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. "3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model." In *NIPS*, December 2012. 46

[FH12] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Distance Transforms of Sampled Functions." *Theory of Computing*, 2012. 46

[FJ03] B. Frey and N. Jojic. "Transformation-invariant clustering using the EM algorithm." *PAMI*, **25**(1):1–17, 2003. 25

[FMY13] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun. "Bottom-up Segmentation for Top-down Detection." In *CVPR*, 2013. 40

[FPC00] W. Freeman, E. Pasztor, and O. Carmichael. "Learning low-level vision." *IJCV*, **40**(1):25–47, 2000. 24

[FVS09] B. Fulkerson, A. Vedaldi, and S. Soatto. "Class segmentation and object localization with superpixel neighborhoods." *ICCV*, 2009. 7

[FZ05] D. Freedman and T. Zhang. "Interactive graph cut based segmentation with shape priors." In *CVPR*, 2005. 43

[GBW10] Josep M Gonfaus, Xavier Boix, Joost Van de Weijer, Andrew D Bagdanov, Joan Serrat, and Jordi Gonzalez. "Harmony potentials for joint classification and segmentation." In *CVPR*, 2010. 10, 22, 74, 116

[GCM13] A. Giusti, D. Ciresan, J. Masci, L.M. Gambardella, and J. Schmidhuber. "Fast image scanning with deep max-pooling convolutional neural networks." In *ICIP*, 2013. 73, 75

[GDD14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *CVPR*, 2014. 56, 73, 87

[GF09] Aleksey Golovinskiy and Thomas Funkhouser. "Min-cut based segmentation of point clouds." In *ICCV Workshops*, 2009. 42

[GFK09] S. Gould, R. Fulton, and D. Koller. "Decomposing a scene into geometric and semantically consistent regions." *ICCV*, 2009. 7, 18, 20, 23

[GG91] Davi Geiger and Federico Girosi. "Parallel and deterministic algorithms from MRFs: Surface reconstruction." *PAMI*, **13**(5):401–412, 1991. 88

[GG92] A. Gersho and R. Gray. *Vector quantization and signal compression*. Kluwer, 1992. 26

[GGA14] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. "Learning rich features from RGB-D images for object detection and segmentation." In *ECCV*, 2014. 116

[GJ07] A. Globerson and T. Jaakkola. "Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations." In *Proc. NIPS*, 2007. 63

[GKF14] Matthieu Guillaumin, Daniel Küttel, and Vittorio Ferrari. "ImageNet Auto-annotation with Segmentation Propagation." *IJCV*, **110**(3):328–348, 2014. 112

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In *CVPR*, 2012. 40, 41, 48

[Gou12] S. Gould. "Multiclass pixel labeling with non-local matching constraints." *CVPR*, 2012. 7, 20, 22, 23

[GRB08] C. Galleguillos, A. Rabinovich, and S. Belongie. "Object categorization using co-occurrence, location and appearance." *CVPR*, 2008. 7

[GRC10] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. "Geodesic star convexity for interactive image segmentation." In *CVPR*, 2010. 43, 46

[GY91] Davi Geiger and Alan Yuille. "A common framework for image segmentation." *IJCV*, **6**(3):227–243, 1991. 88

[HAB11] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. "Semantic contours from inverse detectors." In *ICCV*, 2011. 80, 99

[HAG14a] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. "Hypercolumns for Object Segmentation and Fine-grained Localization." *arXiv:1411.5752*, 2014. 79, 82, 87, 106, 111

[HAG14b] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. "Simultaneous detection and segmentation." In *ECCV*, 2014. 56, 87, 115

[HPM04] Rui Huang, Vladimir Pavlovic, and Dimitris N Metaxas. "A graphical model framework for coupling MRFs and deformable models." In *CVPR*, 2004. 43

[HS06] G. E. Hinton and R. R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science*, 2006. 56

[HS10] T. Hazan and A. Shashua. "Norm-Product Belief Propagation: Primal-Dual Message-Passing for LP-Relaxation and Approximate-Inference." *Trans. Information Theory*, 2010. 63

[HSA84] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley. "Boltzmann Machines: Constraint Satisfaction Networks that Learn." Technical report, University of Toronto, 1984. 56

[HU10] T. Hazan and R. Urtasun. "A Primal-Dual Message-Passing Algorithm for Approximated Large Scale Structured Prediction." In *NIPS*, 2010. 71

[HZC04] Xuming He, Richard S Zemel, and MA Carreira-Perpindn. "Multiscale conditional random fields for image labeling." In *CVPR*, 2004. 74

[HZR06] X. He, R. Zemel, and D. Ray. "Learning and incorporating top-down cues in image segmentation." *ECCV*, 2006. 7

[I 12] S. Avidan I. Olonetsky. "TreeCann - k-d tree Coherence Approximate Nearest Neighbor algorithm." In *ECCV*, 2012. 29

[JFK03] N. Jojic, B.J. Frey, and A. Kannan. "Epitomic analysis of appearance and shape." *ICCV*, 2003. 7, 9, 25

[JKR09] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. "What is the best multi-stage architecture for object recognition?" In *ICCV*, pp. 2146–2153, 2009. 28

[JNR13] J. Jancsary, S. Nowozin, and C. Rother. "Learning Convex QP Relaxations for Structured Prediction." In *ICML*, 2013. 71

[JNS12] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. "Regression Tree Fields – An Efficient, Non-parametric Approach to Image Labeling Problems." In *CVPR*, 2012. 71

[JSD14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding." *arXiv:1408.5093*, 2014. 56, 75, 84, 100

[JT05] F. Jurie and B. Triggs. "Creating efficient codebooks for visual recognition." *ICCV*, 2005. 11

[KAJ11] Hema Swetha Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. "Semantic labeling of 3d point clouds for indoor scenes." In *NIPS*, 2011. 43

[KBB11] P. Kontschieder, S.R. Buló, H. Bischof, and M. Pelillo. "Structured class-labels in random forests for semantic image labelling." *ICCV*, 2011. 10

[KDF08] Iasonas Kokkinos, Rachid Deriche, Olivier Faugeras, and Petros Maragos. "Computational analysis and learning for a biologically motivated model of boundary detection." *Neurocomputing*, **71**(10):1798–1812, 2008. 88

[KF05] Hendrik Kuck and Nando de Freitas. "Learning about individuals from group statistics." In *UAI*, 2005. 95

[KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. 59

[KGF12] Daniel Kuettel, Matthieu Guillaumin, and Vittorio Ferrari. "Segmentation propagation in imagenet." In *ECCV*, 2012. 42

[KH06] S. Kumar and M. Hebert. "Discriminative random fields." *IJCV*, 2006. 7

[KK11] Philipp Krähenbühl and Vladlen Koltun. "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials." In *NIPS*, 2011. 5, 7, 22, 74, 79, 81, 82, 83, 88, 90, 92, 101

[KLT09] Pushmeet Kohli, Lubor Ladicky, and Philip HS Torr. "Robust higher order potentials for enforcing label consistency." *IJCV*, 2009. 15, 74, 78, 83

[KP09] N. Komodakis and N. Paragios. "Beyond pairwise energies: Efficient optimization for higher-order MRFs." *CVPR*, 2009. 10

[KRB08] P. Kohli, J. Rihan, M. Bray, and P.H.S. Torr. "Simultaneous segmentation and pose estimation of humans using dynamic graph cuts." *IJCV*, **79**(3):285–298, 2008. 43

[KSH13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In *NIPS*, 2013. 56, 69, 73, 76, 77

[KTZ10] M. Kumar, P.H.S. Torr, and A. Zisserman. "Objcut: Efficient segmentation using top-down and bottom-up cues." *PAMI*, **32**(3):530–545, 2010. 43

[KXS13] Byungsoo Kim, Shili Xu, and Silvio Savarese. "Accurate Localization of 3D Objects from RGB-D Data using Segmentation Hypotheses." In *CVPR*, 2013. 40, 43

[LBB98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." In *Proc. IEEE*, 1998. 56, 73

[LEC07] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. "An empirical evaluation of deep architectures on problems with many factors of variation." In *Proc. ICML*, 2007. 65

[LFU13] Dahua Lin, Sanja Fidler, and Raquel Urtasun. "Holistic Scene Understanding for 3D Object Detection with RGBD cameras." In *ICCV*, 2013. 43

[LGR09] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." In *ICML*, 2009. 56

[LKR09] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. "Image segmentation with a bounding box prior." In *ICCV*, 2009. 43, 112

[LLB11] Aurélien Lucchi, Yunpeng Li, Xavier Boix, Kevin Smith, and Pascal Fua. "Are spatial and global constraints really necessary for segmentation?" In *ICCV*, 2011. 10, 14, 15, 22, 74

[LLL15] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen Change Loy, and Xiaoou Tang. "Semantic Image Segmentation via Deep Parsing Network." In *ICCV*, 2015. 72, 115

[LM01] T. Leung and J. Malik. "Representing and recognizing the visual appearance of materials using three-dimensional textons." *IJCV*, **43**(1):29–44, 2001. 25

[LMB14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft COCO: Common objects in context." In *ECCV*, 2014. 90, 99, 109

[LMP01] J. Lafferty, A. McCallum, and F. Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." *ICML*, 2001. 7

[LN11] A. Levin and B. Nadler. "Natural image denoising: Optimality and inherent bounds." In *CVPR*, pp. 2833–2840, 2011. 34

[Low04] D.G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV*, **60**(2):91–110, 2004. 1, 24

[LPT13] Joseph Lim, Hamed Pirsiavash, and Antonio Torralba. "Parsing IKEA Objects: Fine Pose Estimation." In *ICCV*, 2013. 43

[LRK09] Lubor Ladicky, Christopher Russell, Pushmeet Kohli, and Philip HS Torr. "Associative hierarchical crfs for object class image segmentation." In *ICCV*, 2009. 10, 22, 74

[LS10] Jorg Liebelt and Cordelia Schmid. "Multi-View Object Class Detection with a 3D Geometric Model." In *CVPR*, 2010. 43

[LSA10] L'ubor Ladickỳ, Paul Sturgess, Karteek Alahari, Chris Russell, and Philip HS Torr. "What, where and how many? combining object detectors and crfs." In *ECCV*. 2010. 116

[LSD14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation." *arXiv:1411.4038*, 2014. 76, 77, 79, 82, 84, 87, 88, 90, 91, 106, 111

[LSP06] Z. Lazebnik, C. Schmid, and J. Ponce. "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories." In *CVPR*, 2006. 24, 34

[LSR15] Guosheng Lin, Chunhua Shen, Ian Reid, et al. "Efficient piecewise training of deep structured models for semantic segmentation." *arXiv:1504.01013*, 2015. 72, 115

[LTL13] Wei-Lwun Lu, Jo-Anne Ting, James J Little, and Kevin P Murphy. "Learning to track and identify players from broadcast sports videos." *PAMI*, 2013. 111

[LVZ11] Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. "Pylon model for semantic segmentation." In *NIPS*, 2011. 7, 20, 23, 74

[LZ14] Y. Li and R. Zemel. "High Order Regularization for Semi-Supervised Learning of Structured Output Problems." In *ICML*, 2014. 71, 95, 96

[Mal99] S. Mallat. *A Wavelet Tour of Signal Processing*. Acad. Press, 2 edition, 1999. 73, 75, 107

[MBH10] D. Munoz, J. Bagnell, and M. Hebert. "Stacked hierarchical labeling." *ECCV*, 2010. 7, 20, 22, 23

[MF08] J. Morris and E. Fosler-Lussier. "Conditional random fields for integrating local discriminative classifiers." *IEEE Trans. Audio, Speech, and Language Processing*, 2008. 71

[MGW09] T. Meltzer, A. Globerson, and Y. Weiss. "Convergent Message Passing Algorithms: a unifying view." In *Proc. UAI*, 2009. 61

[ML09] M. Muja and D. Lowe. "Fast approximate nearest neighbors with automatic algorithm configuration." In *VISAPP*, 2009. 29

[MPW12] J. Ma, J. Peng, S. Wang, and J. Xu. "A conditional neural fields model for protein threading." *Bioinformatics*, 2012. 71

[MSJ10] O. Meshi, D. Sontag, T. Jaakkola, and A. Globerson. "Learning Efficiently with Approximate inference via Dual Losses." In *Proc. ICML*, 2010. 71

[MYS14] Mohammadreza Mostajabi, Payman Yadollahpour, and Gregory Shakhnarovich. "Feedforward semantic segmentation with zoom-out features." *arXiv:1412.0774*, 2014. 74, 77, 84, 87, 90, 106, 111

[NRB11] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. "Decision tree fields." In *ICCV*, 2011. 56

[OBL14] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. "Weakly supervised object recognition with convolutional neural networks." In *NIPS*, 2014. 96

[OF96] B.A. Olshausen and D.J. Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images." *NATURE*, **381**:607–609, 1996. 26

[PBX09] J. Peng, L. Bo, and J. Xu. "Conditional Neural Fields." In *NIPS*, 2009. 71

[PC15] Pedro Pinheiro and Ronan Collobert. "From Image-level to Pixel-level Labeling with Convolutional Networks." In *CVPR*, 2015. 91, 96, 102, 103, 109, 111, 112

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988. 61

[PF10] R. Prabhavalkar and E. Fosler-Lussier. "Backpropagation training for multilayer conditional random field based phone recognition." In *ICASSP*, 2010. 71

[PK12] Patrick Pletscher and Pushmeet Kohli. "Learning low-order models for enforcing high-order statistics." In *AISTATS*, 2012. 95, 96

[PKS14] George Papandreou, Iasonas Kokkinos, and Pierre-André Savalle. "Untangling Local and Global Deformations in Deep Convolutional Networks for Image Classification and Sliding Window Detection." *arXiv:1412.0296*, 2014. 39, 73, 96, 114

[PP97] K. Popat and R. Picard. "Cluster-based probability model and its application to image and texture processing." *IP*, **6**(2):268–284, 1997. 25

[PSL14] Deepak Pathak, Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Multi-Class Multiple Instance Learning." *arXiv:1412.7144*, 2014. 96, 102, 109, 111, 112

[RB09] S. Roth and M.J. Black. "Fields of experts." *IJCV*, 2009. 10

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "Grabcut: Interactive foreground extraction using iterated graph cuts." In *SIGGRAPH*, 2004. 41, 42, 45, 51, 74, 78, 97, 112

[RR13] X. Ren and D. Ramanan. "Histograms of Sparse Codes for Object Detection." In *CVPR*, 2013. 26

[Sch13] A. G. Schwing. *Inference and Learning Algorithms with Applications to 3D Indoor Scene Understanding*. PhD thesis, ETH Zurich, 2013. 63

[SCN07] A. Saxena, S. Chung, and A. Ng. "3-D Depth Reconstruction from a Single Still Image." *IJCV*, 2007. 44

[SEZ13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. "Overfeat: Integrated recognition, localization and detection using convolutional networks." *arXiv:1312.6229*, 2013. 73, 75

[SFP13] A. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. "Box In the Box: Joint 3D Layout and Object Reasoning from Single Images." In *ICCV*, 2013. 48, 51

[SH12] R. R. Salakhutdinov and G. E. Hinton. "An Efficient Learning Procedure for Deep Boltzmann Machines." *Neural Computation*, 2012. 56

[SHB12] R. Socher, B. Huval, B. Bhat, C. D. Manning, and A. Y. Ng. "Convolutional-Recursive Deep Learning for 3D Object Classification." In *NIPS*, 2012. 56

[SJC08] J. Shotton, M. Johnson, and R. Cipolla. "Semantic texton forests for image categorization and segmentation." *CVPR*, 2008. 15

[SLH12] Scott Satkin, Jason Lin, and Martial Hebert. "Data-Driven Scene Understanding from 3D Models." In *BMVC*, 2012. 43

[SLJ14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." *arXiv:1409.4842*, 2014. 73

[SMG08] D. Sontag, T. Meltzer, A. Globerson, and T. Jaakkola. "Tightening LP Relaxations for MAP using Message Passing." In *Proc. NIPS*, 2008. 61, 63

[SPM13] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob J. Verbeek. "Image Classification with the Fisher Vector: Theory and Practice." *IJCV*, **105**(3):222–245, 2013. 36

[SU15] A. G. Schwing and R. Urtasun. "Fully Connected Deep Structured Networks." http://arxiv.org/abs/1503.02351, 2015. 71

[SWR09] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context." *IJCV*, 2009. 7, 18, 74, 82

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv:1409.1556*, 2014. 73, 74, 75, 100, 107

[TCK05] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. "Learning Structured Prediction Models: A Large Margin Approach." In *Proc. ICML*, 2005. 62

[TGK04] B. Taskar, C. Guestrin, and D. Koller. "Max-Margin Markov Networks." In *NIPS*, 2004. 48

[TJH05] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. "Large margin methods for structured and interdependent output variables." *JMLR*, **6**:1453–1484, September 2005. 48

[TJL14] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. "Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation." In *NIPS*, 2014. 56, 71, 73

[TSZ12] Daniel Tarlow, Kevin Swersky, Richard S Zemel, Ryan P Adams, and Brendan J Frey. "Fast Exact Inference for Recursive Cardinality Models." In *UAI*, 2012. 95, 96

[USG13] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. "Selective Search for Object Recognition." *IJCV*, 2013. 87

[Vek08] Olga Veksler. "Star shape prior for graph-cut image segmentation." In *ECCV*, 2008. 43, 46

[VFB12] Alexander Vezhnevets, Vittorio Ferrari, and Joachim M Buhmann. "Weakly supervised structured output learning for semantic segmentation." In *CVPR*, 2012. 109

[VKR08] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. "Graph cut based image segmentation with connectivity priors." In *CVPR*, 2008. 43

[VT07] Jakob Verbeek and Bill Triggs. "Region classification with markov field aspect models." In *CVPR*, 2007. 109

[VZ05] M. Varma and A. Zisserman. "A statistical approach to texture classification from single images." *IJCV*, **62**(1-2):61–81, 2005. 25

[VZ12] A. Vedaldi and A. Zisserman. "Efficient additive kernels via explicit feature maps." *PAMI*, **34**(3):480–492, 2012. 34

[WH03] W. Wiegerinck and T. Heskes. "Fractional belief propagation." In *Proc. NIPS*, 2003. 60

[WJ08] M. J. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families and Variational Inference*. Foundations and Trends in Machine Learning, 2008. 59, 60

[WJP11] P. Weinzaepfel, H. Jégou, and P. Pérez. "Reconstructing an image from its local descriptors." In *CVPR*, 2011. 24, 32

[WJW05] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. "A new class of upper bounds on the log partition function." *Trans. Information Theory*, 2005. 61

[WSL15] Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan Yuille. "Towards Unified Depth and Semantic Prediction from a Single Image." In *CVPR*, 2015. 74

[WYM07] Y. Weiss, C. Yanover, and T. Meltzer. "MAP Estimation, Linear Programming and Belief Propagation with Convex Free Energies." In *Proc. UAI*, 2007. 61

[XDD13] Wei Xia, Csaba Domokos, Jian Dong, Loong-Fah Cheong, and Shuicheng Yan. "Semantic Segmentation without Annotating Segments." In *ICCV*, 2013. 112

[XH10] Xuehan Xiong and Daniel Huber. "Using context to create semantic 3d models of indoor environments." In *BMVC*, 2010. 43

[XOT13] J. Xiao, A. Owens, and A. Torralba. "SUN3D: A Database of Big Spaces Reconstructed using SfM and Object Labels." In *ICCV*, 2013. 42

[XQ09] Jianxiong Xiao and Long Quan. "Multiple view semantic segmentation for street view images." In *ICCV*, 2009. 43

[XSU14] J. Xu, A. Schwing, and R. Urtasun. "Tell Me What You See and I will Show You Where It Is." In *CVPR*, 2014. 42, 56, 109

[YBS13] Payman Yadollahpour, Dhruv Batra, and Gregory Shakhnarovich. "Discriminative reranking of diverse segmentations." In *CVPR*, 2013. 87

[YFU12] Jian Yao, Sanja Fidler, and Raquel Urtasun. "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation." In *CVPR*, 2012. 40, 43

[YFW05] J. S. Yedidia, W. T. Freeman, and Y. Weiss. "Constructing free-energy approximations and generalized belief propagation algorithms." *Trans. Information Theory*, 2005. 61

[YMU13] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. "Robust Monocular Epipolar Flow Estimation." In *CVPR*, 2013. 44

[YSM12] G. Yu, G. Sapiro, and S. Mallat. "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity." *IP*, 2012. 24

[ZCL12] L. Zhu, Y. Chen, Y. Lin, C. Lin, and A. Yuille. "Recursive Segmentation and Recognition Templates for Image Parsing." *PAMI*, 2012. 7, 8, 18, 19, 22, 23

[ZDG14] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. "Part-based R-CNNs for fine-grained category detection." In *ECCV*, 2014. 73

[ZF14] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks." In *ECCV*, 2014. 56, 73

[ZGW05] S.C. Zhu, C.E. Guo, Y. Wang, and Z. Xu. "What are textons?" *IJCV*, **62**(1-2):121–143, 2005. 25

[ZJR15] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. "Conditional Random Fields as Recurrent Neural Networks." *arXiv:1502.03240*, 2015. 71, 72, 88, 90, 92, 106, 107, 111, 115

[ZKT10] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. "Deconvolutional networks." In *CVPR*, 2010. 26

[ZML07] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. "Local features and kernels for classification of texture and object categories: A comprehensive study." *IJCV*, **73**(2):213–238, 2007. 24

[ZMY14] Jun Zhu, Junhua Mao, and Alan L Yuille. "Learning From Weakly Supervised Data by The Expectation Loss SVM (e-SVM) algorithm." In *NIPS*, 2014. 112

[ZSF15] Ziyu Zhang, Alexander G Schwing, Sanja Fidler, and Raquel Urtasun. "Monocular Object Instance Segmentation and Depth Ordering with CNNs." *arXiv:1505.03159*, 2015. 54

[ZSS13] Zeeshan Zia, Michael Stark, Bernt Schiele, and Konrad Schindler. "Detailed 3D Representations for Object Recognition and Modeling." *PAMI*, 2013. 43

[ZUS15] Yukun Zhu, Raquel Urtasun, Ruslan Salakhutdinov, and Sanja Fidler. "segDeepM: Exploiting segmentation and context in deep neural networks for object detection." *arXiv:1502.04275*, 2015. 116

[ZW11] D. Zoran and Y. Weiss. "From learning models of natural image patches to whole image restoration." *ICCV*, 2011. 10, 24

[ZZY13] Bo Zheng, Yibiao Zhao, Joey C. Yu, Katsushi Ikeuchi, and Song-Chun Zhu. "Beyond Point Clouds: Scene Understanding by Reasoning Geometry and Physics." In *CVPR*, 2013. 43