Learning Structured Prediction Models in Computer Vision

by

Fayao Liu

A thesis submitted in fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering, Computer and Mathematical Sciences
School of Computer Science

November 2015

Declaration

Acknowledgements

First and foremost, I would like to express my sincere gratitudes to my principle supervisor, Prof. Chunhua Shen. I would have never been able to finish this thesis without his guidance. During the course of my PhD study, he has always been an encouraging, inspiring and patient mentor, from whom I've learned not only methodologies, but the way to look at problems. His intelligence, creativity and keen perceptions in cutting-edge research topics have deeply impressed me. He has also set me a good example by his diligence and continuous efforts, as well as rigorous scientific attitudes. There are numerous other things I've learned from him, which will continue guiding me in my future career.

I would like to thank my co-supervisors, Prof. Anton van den Hengel and Prof. David Suter. They have showed generous patience and continuous support throughout my PhD candidature. As the director of the Australian Centre for Visual Technologies (ACVT), Anton has provided me a good platform as well as opportunities to communicate and collaborate with many talented researchers. He has also given me a lot of help on improving my language skills. I appreciate his generosity, encouragement and enlightenment.

I would like to thank Prof. Ian Reid, who showed me in person how to organize and write a research paper by telling a story. His innovative perspectives and insightful advices have helped me to improve my understandings and paper writing. It has always been pleasant and inspiring discussing with him.

I owe special thanks to Dr. Guosheng Lin, who taught me structured learning and many other things. He impressed me by his profound professional expertise and his extraordinary persistence. I'm deeply grateful for his unselfish sharing and constant supporting throughout my PhD life.

My sincere thanks go to ACVT researchers, especially Dr. Qinfeng (Javen) Shi, Dr. Peng Wang and Dr. Sakrapee (Paul) Paisitkriangkrai, for their kindness and valuable suggestions. Talking and discussing with them have always benefited me a lot.

Many thanks to all my current and previous lab mates, with whom I spent the most important years of my life together. Especially, I would like to mention Quoc-Huy Tran, Zhen Zhang, Yongrui Qin, Lina Yao, Lei Luo, Chao Zhang. I will always treasure those days spent with them. I also owe thanks to my friends I do not list here, with whom I share excitements and frustrations.

Finally, special gratitudes are attributed to my family, to whom this thesis is dedicated to.

Publications

This thesis is based on the content of the following peer-reviewed conference and journal papers:

1. Fayao Liu, Chunhua Shen, Guosheng Lin; *"Deep Convolutional Neural Fields for Depth Estimation from a Single Images"*; In proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
2. Fayao Liu, Chunhua Shen, Guosheng Lin, Ian D. Reid; *"Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields"*; Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015.
3. Fayao Liu, Guosheng Lin, Chunhua Shen; *"CRF Learning with CNN Features for Image Segmentation"*; Pattern Recognition (PR), 2015.
4. Fayao Liu, Guosheng Lin, Chunhua Shen; *"Structured Learning of Tree Potentials in CRF for Image Segmentation"*; Submitted to IEEE Transactions on Neural Networks and Learning Systems (TNNLS); Major Revision.

In addition, I have published or submitted the following papers:

1. Fayao Liu, Luping Zhou, Chunhua Shen, Jianping Yin; *"Multiple Kernel Learning in the Primal for Multimodal Alzheimer's Disease Classification"*; In IEEE Journal of Biomedical and Health Informatics (JBHI), 2014.
2. Fayao Liu, Guosheng Lin, Chunhua Shen; *"Discriminative Training of Deep Fully-connected Continuous CRFs with Task-specific Loss"*; Submitted to IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
3. Fayao Liu, Ruizhi Qiao, Chunhua Shen, Lei Luo; *"From Kernel Machines to Ensemble Learning"*; Submitted to Pattern Recognition (PR).
4. Fayao Liu, Chunhua Shen, Ian Reid, Anton van den Hengel; *"Online Unsupervised Feature Learning for Visual Tracking"*; Submitted to Computer Vision and Image Understanding (CVIU).
5. Chunhua Shen, Junae Kim, Fayao Liu, Lei Wang, Anton van den Hengel; *"Efficient Dual Approach to Distance Metric Learning"*; In IEEE Transactions on Neural Networks and Learning Systems (TNNLS), 2014.

THE UNIVERSITY OF ADELAIDE

Abstract

Faculty of Engineering, Computer and Mathematical Sciences
School of Computer Science

Doctor of Philosophy

by Fayao Liu

Most of the real world applications can be formulated as structured learning problems, in which the output domain can be arbitrary, *e.g.*, a sequence or a graph. By modelling the structures (constraints and correlations) of the output variables, structured learning provides a more general learning scheme than simple binary classification or regression models. This thesis is dedicated to learning such structured prediction models, *i.e.*, conditional random fields (CRFs) and their applications in computer vision. CRFs are popular probabilistic graphical models, which model the conditional distribution of the output variables given the observations. They play an essential role in the computer vision community and have found wide applications in various vision tasks—semantic labelling, object detection, pose estimation, to name a few. Specifically, we here focus on two challenging tasks in this thesis: image segmentation (also referred as semantic labelling) and depth estimation from single monocular images, which represent two types of CRFs models—discrete and continuous. In summary, we made three contributions in this thesis.

First, we present a new approach to exploit tree potentials in CRFs for the task of image segmentation. This method combines the advantages of both CRFs and decision trees. Different from traditional methods, in which the potential functions of CRFs are defined as a *linear* combination of some pre-defined parametric models, we formulate the unary and the pairwise potentials as nonparametric forests—ensembles of decision trees, and learn the ensemble parameters and the trees in a unified optimization problem within the large-margin framework. In this fashion, we easily achieve *nonlinear* learning of potential functions on both unary and pairwise terms in CRFs. Moreover, we learn class-wise decision trees for each object that appears in the image. We further show that this challenging optimization can be efficiently solved by combining a modified column generation and cutting-planes techniques. Experimental results on both binary and multi-class segmentation datasets demonstrate the power of the learned nonlinear nonparametric potentials.

Second, we propose to model the unary potentials of the CRFs using a convolutional neural network (CNN). The deep CNN is trained on the large-scale ImageNet dataset and transferred to image segmentation here for constructing unary potentials of superpixels. The CRFs parameters are then learned within the max-margin framework using structured support vector machines (SSVM). To fully exploit context information in inference, we construct spatially related co-occurrence pairwise potentials and incorporate them into the energy function. This prefers labellings of object pairs that frequently co-occur in a certain spatial layout and at the same time avoids implausible labellings during the inference. Extensive experiments on binary and multi-class segmentation benchmarks demonstrate the potentials of the proposed method.

Third, different from the previous two works, we address the problem of continuous CRFs learning, applied to the task of depth estimation from single images. Specifically, we formulate and learn the unary and pairwise potentials of a continuous CRFs model with CNN networks in a unified framework. We term this new method as deep convolutional neural fields, abbreviated as DCNF. It jointly explores the capacity of deep CNN and continuous CRFs. The proposed method can be used for depth estimation of general scenes with no geometric priors nor any extra information injected. Specifically, in our case, the integral of the partition function can be calculated in a closed form such that we can exactly solve the log-likelihood maximization. Moreover, solving the inference problem for predicting depths of a test image is highly efficient as closed-form solutions exist. We then further propose an equally effective model based on fully convolutional networks and a novel superpixel pooling method, which is ~ 10 times faster, to speedup the patch-wise convolutions in the deep model. With this more efficient model, we are able to design very deep networks to pursue further performance gain. Experiments on both indoor and outdoor scene datasets demonstrate that the proposed method significantly outperforms state-of-the-art depth estimation approaches. We also show experimentally that the proposed method generalizes well to depth estimations of images unrelated to the training data. This indicates the potential of our method for benefiting other vision tasks.

Dedicated to my family.

Contents

List of Figures

List of Tables

Notation

Symbol	Description
$\mathbf{1}$	Column vector with all elements being 1.
$\mathbf{0}$	Column vector with all elements being 0.
$\mathbf{I}$	Identity matrix.
$\mathbb{R}$	Domain of real numbers.
i.i.d.	Abbreviation of independent and identically distributed.
$< \cdot, \cdot >$	Inner product operation.
$\odot$	Stacking two vectors.
$\otimes$	Kronecker tensor.
$\mathrm{Tr}(\cdot)$	Trace of a matrix.
$\Vert\cdot\Vert_2$	L_2 norm.
Superscript $\top$	Transpose.
$\delta(\cdot)$	Indicator function which equals 1 if the input is true and 0 otherwise.
C	Trade-off parameter.
m	Number of examples.
$\boldsymbol{\xi}$	Vector of slack variables.
$\mathbf{w}$	Vector of model parameters.
$\mathbf{x}$	Input observation.
$\mathbf{y}$	Structured output label.
y	Scalar output label.
$\mathcal{X}$	Input domain.
$\mathcal{Y}$	Output domain.
$\mathcal{N}$	Set of nodes.
$\mathcal{S}$	Set of edges.
$\mathcal{W}$	Working set.
$\mathcal{H}$	Domain of weak learners/decision trees.
$g : \mathcal{X} \to \mathcal{Y}$	Structured prediction function.
$f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$	Scoring function.
$l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$	General loss function.
$\varDelta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$	Structured loss function.

E	Energy function.
U	Unary potential function.
V	Pairwise potential function.
Ψ	Feature mapping function.
$\Psi^{(1)}$	Unary feature mapping function.
$\Psi^{(2)}$	Pairwise feature mapping function.
Pr	Probability function.
Z	Partition function.
sgn	Sign function.
$\hbar$	A weak learner.
$\hbar^{(1)}$	A unary decision tree.
$\hbar^{(2)}$	A pairwise decision tree.
$\mathbf{H}^{(1)}$	A group of unary decision trees.
$\mathbf{H}^{(2)}$	A group of pairwise decision trees.

Chapter 1

Introduction

In recent decades, advances in the artificial intelligence (AI) have been changing people's daily lives. Smart phones, intelligent homes, autonomous vehicles (though far from being mature), all of which are revealing that we are entering an intelligent era. In the center of the various progresses stands the driving horse—machine learning. The ultimate goal of machine learning is to enable computers to learn like human beings. Towards this goal, great endeavors have been made in the past few years. Among them, two of the most notable achievements are structured learning and deep learning, with the former solving complex prediction problems, and the latter learning hierarchical feature representations. In this thesis, we focus on the structured learning topic since most of the real world applications can be formulated as structured learning problems. We also exploit deep structured learning methods that combine the benefits of both.

As an important branch of machine learning related applications, computer vision has been constantly drawing a lot of attentions from researchers. Typical computer vision tasks include image classification, semantic segmentation, object detection, 3D reconstruction, visual tracking *etc.*. We here focus on two particular vision applications: image semantic segmentation and depth estimation from single monocular images. Both of these two tasks can be regarded as the pixel map labelling problem, namely, to predict a pixel-wise label map from a given input image, which can be naturally formulated as a structured learning problem. One of the major differences between the two tasks is that the label domain of the former is discrete high level semantics and the latter is continuous low level depths. Both of them can be approached by learning (discrete or continuous) conditional random fields (CRF) models, which we will detail in this thesis.

We first introduce some general notation rules.

Notations A "$\geq$" or "$\leq$" between two vectors denotes element-wise inequality. Unless otherwise stated, we use boldfaced uppercase and lowercase letters to denote matrices and column vectors respectively.

1.1 Structured Learning

In the real world applications, most of the computer vision tasks can be formulated as structured output prediction problems. Unlike in the conventional classification and regression models, where the output of one input is represented as a single value (discrete in the classification case and continuous in the regression case), the terminology *structured* here indicates that the output is a complex multivariate object and can not be represented as a single value. Furthermore, the components of each output object are interdependent and correlated, *e.g.*, a sequence, a parsing tree or a graph. In this circumstance, to obtain the optimal prediction, we need to learn a mapping function from the given input to the structured output domain: $g : \mathcal{X} \to \mathcal{Y}$, which takes the following form:

$$\hat{\mathbf{y}} = g(\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}} \; f(\mathbf{y}, \mathbf{x}). \tag{1.1}$$

Here $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$; $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a scoring function that measures the compatibility between the input and the output. Typical structured learning tasks include speech tagging in natural language processing, protein structure prediction in bioinformatics, pixel map labelling in vision *etc.*. We in this thesis focus on applications in computer vision, specifically, image segmentation and single image depth estimation.

1.2 Conditional Random Fields

As one of the most popular structured learning models, conditional random fields (CRF) [11] model the outputs as random variables in an undirected graphical $G = (\mathcal{N}, \mathcal{S})$. Here $\mathcal{N}$, $\mathcal{S}$ are the sets of nodes and edges respectively. We here take the image segmentation task as an example to illustrate how CRF work. For an image $\mathbf{x}$, each (super-)pixel is regarded as a node, with an edge connecting each neighbouring node pair[1]. Each node p is associated with a random variable y_p depicting its semantic label, which takes values from $\{1, 2, \ldots, K\}$. Here K is the total number of categories. The labelling of the image $\mathbf{x}$ is denoted as $\mathbf{y}$, which is composed of all y_p. CRF model the conditional probability

[1]We here consider locally-connected graphs. In the case of fully-connected graphs, there exits an edge between each pair of nodes.

distribution of the image through an energy function E:

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{\mathrm{Z}(\mathbf{x})} \exp\big(-E(\mathbf{y},\mathbf{x};\mathbf{w})\big), \tag{1.2a}$$

$$\text{with } \mathrm{Z}(\mathbf{x}) = \sum_{\mathbf{y}} \exp\big(-E(\mathbf{y},\mathbf{x};\mathbf{w})\big). \tag{1.2b}$$

Here $\mathrm{Z}(\mathbf{x})$ is the normalization term, *i.e.*, partition function; $\mathbf{w}$ is a vector of model parameters.

The energy can be interpreted as the cost of assigning label $\mathbf{y}$ to the image $\mathbf{x}$. Typically, E is composed of unary potentials of all nodes and pairwise potentials of all edges[2], which takes the following form:

$$E(\mathbf{y},\mathbf{x};\mathbf{w}) = \sum_{p\in\mathcal{N}} U(y_p,\mathbf{x};\mathbf{w}) + \sum_{(p,q)\in\mathcal{S}} V(y_p,y_q,\mathbf{x};\mathbf{w}). \tag{1.3}$$

Here U, V denotes the unary and pairwise potential function respectively, both of which depend on observations $\mathbf{x}$ and the model parameters $\mathbf{w}$. Intuitively, the unary potential encodes the negative log likelihood of the p-th node taking label y_p, while the pairwise term enforces local smoothness by encouraging neighbouring nodes to take similar labels.

Generally, E is constructed as a linear form of $\mathbf{w}$, *i.e.*, $E(\mathbf{y},\mathbf{x};\mathbf{w}) = \mathbf{w}^\top\Psi(\mathbf{y},\mathbf{x})$, which implies that the potential functions in Eq. (1.3) can be represented as:

$$U(y_p,\mathbf{x};\mathbf{w}) = \big\langle \mathbf{w}^{(1)}, \Psi^{(1)}(y_p,\mathbf{x})\big\rangle, \tag{1.4}$$

$$V(y_p,y_q,\mathbf{x};\mathbf{w}) = \big\langle \mathbf{w}^{(2)}, \Psi^{(2)}(y_p,y_q,\mathbf{x})\big\rangle, \tag{1.5}$$

where $\mathbf{w} = \mathbf{w}^{(1)} \odot \mathbf{w}^{(2)}$; $\Psi = \Psi^{(1)} \odot \Psi^{(2)}$, with $\Psi^{(1)}$, $\Psi^{(2)}$ being the unary and pairwise feature mapping functions respectively.

Learning the model parameters $\mathbf{w}$ can be performed in a standard way of minimizing the negative log-likelihood of the training data. During prediction, one performs the Maximum a Posterior (MAP) inference to find an optimum labelling $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}} E(\mathbf{y},\mathbf{x};\mathbf{w}). \tag{1.6}$$

1.2.1 Limitations of Current CRF Models

As stated in Sec. 1.2, most of the current CRF models rely on a linear energy function $E(\mathbf{y},\mathbf{x};\mathbf{w}) = \mathbf{w}^\top\Psi(\mathbf{y},\mathbf{x})$, which is a linear combination of a series of parametric

[2]In more complicated cases, higher order potentials can be incorporated.

feature mappings (usually pre-defined). While nonlinear models typically yield more favorable performance, it is generally hard to conduct nonlinear potential learning in CRF. Although it is possible to train kernel CRF [12], the resulted optimization is typically expensive to solve. Moreover, kernel methods are well known to be not scalable. Nevertheless, a linear energy function formulation is intrinsically limiting CRF, which calls for more efficient non-linear CRF learning methods.

1.3 Contributions

The contributions of this thesis are on the fields of learning structured prediction models and their applications in computer vision. In Chapter 2, we review the literature background of several related topics, *i.e.*, supervised learning, structured learning, ensemble learning and deep convolutional neural networks. We present two discrete CRF learning methods for image segmentation in chapter 3 and chapter 4. We next propose a deep continuous CRF learning model for depth estimations from single monocular images. Finally in Chapter 6, we conclude the thesis and discuss directions for future work. We describe the contributions in more detail as follows:

Chapter 3

In this chapter, we present a nonlinear CRF learning method for image segmentation (semantic labelling). The unary and the pairwise potential functions are respectively composed of an ensemble of decision trees. We jointly learn the trees and the ensemble parameters in the large margin framework. This is achieved by inspecting the KKT conditions of the SSVM formulation and then applying the column generation technique. The resulted optimization problem is then efficiently solved by the cutting-plane algorithm. Our formulation also enables learning of class-wise potentials for each of the objects that appear in the image. Experimental results demonstrate that the proposed nonlinear learning of CRF potentials outperforms traditional linear and parametric models.

Chapter 4

In this chapter, we propose to incorporate CNN potentials in CRF learning for image segmentation [13]. The CNN model is pre-trained on the ImageNet dataset and transferred here to construct the unary potential for image segmentation. For the pairwise potential, we model spatially related co-occurrence correlations for better capturing the contextual information. The CRF parameters are then learned using an SSVM. Additional contributions include that we conduct comprehensive comparisons between traditional hand-crafted features and CNN features. Extensive experiments demonstrate the effectiveness and strength of the proposed method.

Chapter 5

In this chapter, we present a novel deep structured learning method, referred as deep convolutional neural field (DCNF), for depth estimations from single monocular images [14, 15]. Different from the previous two chapters, which explore discrete CRF, this chapter addresses the continuous CRF learning problem for the task of depth estimation. By jointly exploiting the capacity of continuous CRF and deep CNN, DCNF aims to estimate depths from single images without incorporating any geometric priors or extra information. We further propose a more efficient while equally effective model DCNF-FCSP based on fully convolutional networks and a novel superpixel pooling method. With this new model, we are able to design deeper CNN networks to pursue further performance gain. We experimentally demonstrate that our method outperforms state-of-the-art methods and generalizes well to general scene image depth estimations.

Chapter 2

Background Literature

This chapter reviews some background literature on the supervised learning, structured learning, ensemble learning and convolutional neural networks. These are the foundations of the following chapters. We also introduce some popular existing methods that are related to the focus of this thesis. Note that we present the conventional supervised learning methods in a way different from the standard presentation in the community, in order to give unified knowledge and facilitate the generalization to the more complicated structured learning cases. Specifically, the structured support vector machines (SSVM) introduced in Sec. 2.2.1 is a generalization of the support vector machines (SVM) described in Sec. 2.1.1 to structured learning cases. Likewise, the conditional random fields (CRF) in Sec. 2.2.2 is a generalization of the logistic regression presented in Sec. 2.1.2 to structured learning cases.

2.1 Supervised Learning

Supervised learning is the task of learning a prediction function from a set of *i.i.d* training instances, with typical examples being classification and regression. Without loss of generality, we here consider the multi-class classification problem where the output domain is $\mathcal{Y} = \{1, 2, \ldots, K\}$, with K being the number of classes. Given m labelling pairs $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$, where $\mathbf{x}^{(i)} \in \mathcal{X}$, $y^{(i)} \in \mathcal{Y}$, we aim to learn a prediction function $g : \mathcal{X} \to \mathcal{Y}$ of the following form:

$$\hat{y} = g(\mathbf{x}) = \underset{y}{\operatorname{argmax}}\; f(\mathbf{x}, y), \tag{2.1}$$

where $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a compatibility score function which measures the consistency between the input $\mathbf{x}$ and the label y. The argmax operation then chooses the prediction

y that gives the highest score as the final prediction $\hat{y}$. Typically, f takes a linear form of the model parameters $\mathbf{w}$ and a joint feature mapping Ψ:

$$f(\mathbf{x}, y) = \mathbf{w}^\top \Psi(\mathbf{x}, y). \tag{2.2}$$

This class-dependent joint feature mapping Ψ captures the attributes on which the input-output pair compatibility may depends. The design of Ψ depends on applications. Although in the linear form as defined in Eq. (2.2), it enables the use of the kernel trick [16] to yield a non-linear classifier.

To learn the model parameters $\mathbf{w}$, one generally seeks to minimize a regularized risk minimization functional:

$$\begin{aligned} \min_{\mathbf{w}} \; & J(\mathbf{w}) := C\Omega(\mathbf{w}) + R_{emp}(\mathbf{w}), \\ \text{where} \; & R_{emp}(\mathbf{w}) := \frac{1}{m}\sum_{i=1}^{m} l(y^{(i)}, g(\mathbf{x}^{(i)})). \end{aligned} \tag{2.3}$$

Here, R_{emp} is the empirical risk defined by a convex loss function l; $C\Omega(\mathbf{w})$ is a regularization term that controls the model complexity with the pre-defined trade-off parameter $C > 0$. The regularizer $\Omega(\mathbf{w})$ can be L_1, L_2 or any other types of regularizations, which lead to sparse Lasso-type, non-sparse solutions respectively.

The loss function l measures the discrepancy between the true label $y^{(i)}$ and the predicted label $\hat{y} = g(\mathbf{x}^{(i)})$. The most common type of loss function for classification is the 0/1 loss:

$$l^{0/1}(y^{(i)}, \hat{y}) = \delta(y^{(i)} \neq \hat{y}), \tag{2.4}$$

where $\delta(\cdot)$ is an indicator function which equals 1 if the input is true and 0 otherwise. Directly optimizing the 0/1 loss is generally difficult partly due to its discontinuity, therefore many approaches propose to optimize an upper bound, *e.g.*, hinge loss in support vector machines (SVM) [17] and log-loss in logistic regression. See Fig. 2.1 for an intuitive illustration. Both of these two methods lead to efficient optimization of Eq. (2.3). More importantly, they provide some new perspectives on the learning problems in that the former leads to the notion of margins while the latter enables an probabilistic interpretation of the classifier. We will detail these two methods in Sec. 2.1.1 and Sec .2.1.2 respectively. One more disadvantage of the 0/1 loss in Eq. (2.4) is that it treats all mis-classifications equivalently. A more reasonable way to deal with this is to assign different costs to different labelling pairs, which calls for more general loss functions.

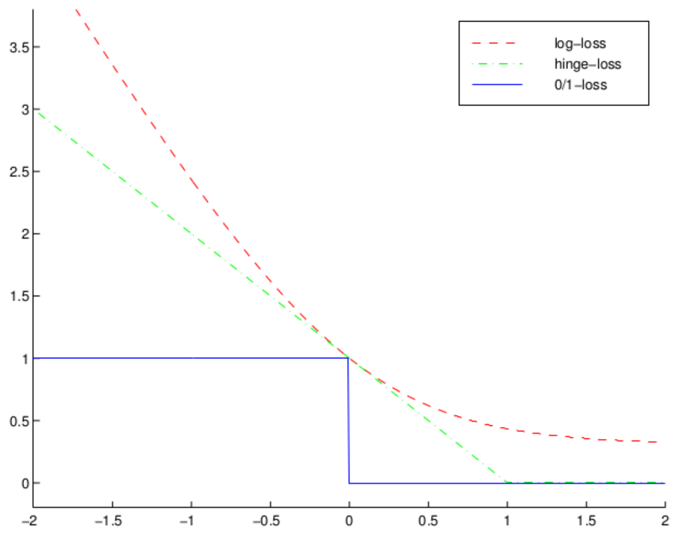


Figure 2.1: An illustration of the 0/1 loss upper bounded by the hinge loss and the log loss. The horizontal axis shows $\mathbf{w}^\top\Psi(\mathbf{x}, y) - \max_{y'\neq y}\mathbf{w}^\top\Psi(\mathbf{x}, y')$, where y is the correct label for the example $\mathbf{x}$, while the vertical axis quantifies the loss. As shown, the 0/1 loss is discontinuous, while the hinge loss is continuous; the log-loss is continuous and smooth. Figure reproduced from [1].

In general, different choices of l and Ω leads to different machine learning methods. For example, SVM optimizes a combination of the hinge loss and L_2 regularization, while the logistic regression optimizes the log-loss with L_2 regularization. For the case of regression, the least square loss with L_1 regularization leads to the Lasso algorithm [18]. When the label is no longer a single value y, but a multi-variate object $\mathbf{y}$, and l is a structured loss function, it leads to a structured model, which we will detail in Sec. 2.2.

2.1.1 Support Vector Machines

Support Vector Machines (SVM) [17] are popular max-margin based machine learning methods. Intuitively, the "margin" of an example is positive if and only if the example is correctly classified, with its magnitude measuring the prediction confidence.

Let g be the prediction function defined in Eq. (2.1) and Eq. (2.2), which we repeat here as:

$$\hat{y} = g(\mathbf{x}) = \operatorname*{argmax}_y f(\mathbf{x}, y) = \operatorname*{argmax}_y \mathbf{w}^\top\Psi(\mathbf{x}, y). \tag{2.5}$$

Hard-margin Formulation To learn the prediction function, or more specifically $\mathbf{w}$, ideally, we want the score of the correct label to be larger than the score of any other incorrect labels. In other words, we want to maximize the margin γ, *i.e.*, the smallest score difference between the correct label $y^{(i)}$ and the closest runner-up $y =$

$\operatorname{argmax}_{y\in\mathcal{Y}\backslash y^{(i)}} \mathbf{w}^\top\Psi(\mathbf{x},y)$. Adding L_2 regularization to $\mathbf{w}$, we can obtain the following hard-margin optimization problem:

$$\max_{\mathbf{w},\gamma} \ \gamma \tag{2.6a}$$

$$\text{s.t.: } \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y^{(i)}) - \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y) \geq \gamma, \forall i, \forall y \in \mathcal{Y}\backslash y^{(i)}, \tag{2.6b}$$

$$\|\mathbf{w}\|_2 = 1. \tag{2.6c}$$

The optimization problem in Eq. (2.6) is equivalent to:

$$\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|_2 \tag{2.7a}$$

$$\text{s.t.: } \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y^{(i)}) - \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y) \geq 1, \forall i, \forall y \in \mathcal{Y}\backslash y^{(i)}. \tag{2.7b}$$

Soft-margin Formulation In practical, the constraints in Eq. (2.7b) can not be all perfectly satisfied. In this case, slack variables $\xi_i(i=1,\ldots,m)$ are introduced for each example $\mathbf{x}^{(i)}$ to allow violations of the constraints [19]:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|_2 + \frac{C}{m}\sum_i \xi_i \tag{2.8a}$$

$$\begin{aligned}\text{s.t.: } & \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y^{(i)}) - \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y) \\ & \geq 1-\xi_i, \forall y \in \mathcal{Y}\backslash y^{(i)} \text{ and } \forall i = 1,\ldots,m,\end{aligned} \tag{2.8b}$$

$$\boldsymbol{\xi} \geq \mathbf{0}. \tag{2.8c}$$

The constraints in Eq. (2.8) can be simplified as

$$\mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y^{(i)}) - \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y) \geq l^{0/1}(y^{(i)},\hat{y}) - \xi_i, \forall i, \forall y, \tag{2.9}$$

where $l^{0/1}(y^{(i)},\hat{y})$ is the 0/1 loss defined in Eq. (2.4). We therefore can write Eq. (2.8) into the following equivalent form:

$$\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|_2 + \frac{C}{m}\sum_i l^{\text{hinge}}(y^{(i)},\hat{y}), \tag{2.10}$$

with $l^{\text{hinge}}(y^{(i)},\hat{y})$ being the multi-class hinge loss introduced by Crammer and Singer [19]:

$$l^{\text{hinge}}(y^{(i)},\hat{y}) = \max_y \left[l^{0/1}(y^{(i)},y) + \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y)\right] - \mathbf{w}^\top\Psi(\mathbf{x}^{(i)},y^{(i)}). \tag{2.11}$$

If the score of the correct label is larger by at least 1 than the maximum score of the wrong labels, the hinge loss equals 0. Otherwise, it scales linearly with this score difference, as illustrated in Fig. 2.1. From the above equations Eq. (2.10), Eq. (2.11), we can clearly see that SVM is optimizing the regularized risk minimization functional in Eq. (2.3) with the hinge loss and L_2 regularization.

Next, we show that the hinge loss defined in Eq. (2.11) is an upper-bound of the 0/1 loss in Eq. (2.4). From Eq. (2.5), we have:

$$\begin{aligned} &\hat{y} = \underset{y}{\operatorname{argmax}}\ \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, y) \\ \implies\ &\mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, \hat{y}) \geq \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, y^{(i)}). \end{aligned} \tag{2.12}$$

Therefore the following derivation holds:

$$\begin{aligned} l^{\text{hinge}}(y^{(i)}, \hat{y}) &= \max_y \left[l^{0/1}(y^{(i)}, y) + \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, y) \right] - \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, y^{(i)}) \\ &\geq l^{0/1}(y^{(i)}, \hat{y}) + \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, \hat{y}) - \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, y^{(i)}) \\ &\geq l^{0/1}(y^{(i)}, \hat{y}). \end{aligned} \tag{2.13}$$

This accomplishes the proof. This relation is also shown in Fig. 2.1.

2.1.2 Logistic Regression

Different from the max-margin based SVM method, logistic regression takes a probabilistic perspective by defining the score function f in Eq. (2.2) as the conditional probability distribution:

$$\Pr(y|\mathbf{x}; \mathbf{w}) = \frac{1}{\mathrm{Z}(\mathbf{x})} \exp(\mathbf{w}^\top \Psi(\mathbf{x}, y)), \tag{2.14a}$$

$$\text{with } \mathrm{Z}(\mathbf{x}) = \sum_y \exp(\mathbf{w}^\top \Psi(\mathbf{x}, y)), \tag{2.14b}$$

where $\mathrm{Z}(\mathbf{x})$ is the normalization term, *i.e.*, partition function. By defining such a probability distribution, the outputs of different candidate labels can be regarded as confidence scores and become readily comparable. The prediction is then performed by maximizing the confidence score as:

$$\hat{y} = g(\mathbf{x}) = \underset{y}{\operatorname{argmax}}\ f(\mathbf{x}, y) = \underset{y}{\operatorname{argmax}}\ \Pr(y|\mathbf{x}; w). \tag{2.15}$$

The model parameters $\mathbf{w}$ are generally learned by maximizing the conditional (log-)likelihood (or minimizing the negative (log-)likelihood) of the training data. Adding L_2 regularization, we then arrive at the optimization problem of the logistic regression method:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|_2 + \frac{C}{m}\sum_i \log \mathrm{Z}(\mathbf{x}^{(i)}) - \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, y^{(i)}). \tag{2.16}$$

It can be easily observed that Eq. (2.16) is optimizing the regularized risk minimization in Eq. (2.3) with L_2 regularization and the log loss defined as:

$$l^{\log}(y^{(i)}, \hat{y}) = -\log \Pr(\hat{y}|\mathbf{x}^{(i)}; \mathbf{w}) = \log \mathrm{Z}(\mathbf{x}^{(i)}) - \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, \hat{y}). \tag{2.17}$$

As shown in Fig. 2.1, the *log* loss is also an upper bound of the 0/1 loss.

For the multi-class classification problem we consider here, the joint feature mapping Ψ can be constructed as:

$$\Psi(\mathbf{x}, y) = [\delta(y = 1)\mathbf{f}^\top, \ldots, \delta(y = K)\mathbf{f}^\top]^\top, \tag{2.18}$$

where $\mathbf{f}$ is the feature vector associated with the example $\mathbf{x}$ and $\delta(\cdot)$ is the indicator function which equals 1 if the input is true and 0 otherwise. With this definition in Eq. (2.18), we can denote Ψ as $\Psi = \Psi_1 \odot \ldots \odot \Psi_K$, where $\odot$ stacks two column vectors and $\Psi_k(\mathbf{x}, y) = \delta(y = k)\mathbf{f}^\top$, $k = 1, \ldots, K$. Accordingly $\mathbf{w}$ can be decomposed as $\mathbf{w} = \mathbf{w}_1 \odot \ldots \odot \mathbf{w}_K$. Then the conditional probability distribution in Eq. (2.14) can be written as:

$$\Pr(y|\mathbf{x}; \mathbf{w}) = \frac{1}{\mathrm{Z}(\mathbf{x})} \prod_k \exp(\mathbf{w}_k^\top \Psi_k(\mathbf{x}, y)), \tag{2.19a}$$

$$\text{with } \mathrm{Z}(\mathbf{x}) = \sum_y \prod_k \exp(\mathbf{w}_k^\top \Psi_k(\mathbf{x}, y)). \tag{2.19b}$$

We present this new form in Eq. (2.19) so as to show the connection between the logistic regression and the conditional random fields (CRF) introduced later in Sec. 2.2.2.

2.2 Structured Learning

In the case of structured learning, the label of an example $\mathbf{x}$ can not be represented as a single value y as in the classification and regression cases, but a multi-variate complex object $\mathbf{y}$, *e.g.*, a sequence, a parsing tree or an image. Structured learning then aims to learn a structured output function $\mathbf{y} = g(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$. The prediction function takes a similar form as in Eq. (2.1):

$$\hat{\mathbf{y}} = g(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}, \mathbf{y}). \tag{2.20}$$

Likewise, f is a score function measuring the consistency or compatibility between the input $\mathbf{x}$ and the output $\mathbf{y}$. During the course of learning, two issues need to be solved: learning the model parameters of f from the training data; inferring the optimal labellings given the current learned model.

For the learning task, existing methods can be categorized into two veins: probabilistic methods and max-margin based methods. Probabilistic methods model the underline data probability distribution, and are typically solved by maximizing the likelihood of the training data. These methods require an expensive normalization calculation step to ensure a valid probability distribution. For example, the Markov Random Fields (MRF) [20] model the joint probability distribution of the input and the output variables $\Pr(\mathbf{x}, \mathbf{y})$. While the Conditional Random Fields (CRF) [11] model the conditional probability distribution given the input observations:

$$\begin{aligned} & \Pr(\mathbf{y}|\mathbf{x}) \geq 0, && (2.21a) \\ \text{with} \; & \sum_{\mathbf{y} \in \mathcal{Y}} \Pr(\mathbf{y}|\mathbf{x}) = 1, \forall \mathbf{x} \in \mathcal{X}. && (2.21b) \end{aligned}$$

Compared to the generative MRF models, CRF do not need to model the data distribution $\Pr(\mathbf{x})$ as involved in the joint probability $\Pr(\mathbf{x}, \mathbf{y})$. This can be advantageous since modelling $\Pr(\mathbf{x})$ can be difficult. By directly focusing on the discriminative problem, CRF can yield more accurate solutions. In most of the cases, the normalizations in both the generative model and the discriminative model are intractable. Hence approximations are generally required. Nevertheless, one of the primary benefits of the probabilistic methods is that they can naturally incorporate latent variables through marginalization.

In contrast to the probabilistic methods, max-margin based methods directly focus on the margin or decision boundary instead of learning a normalized data distribution. This kind of methods therefore do not need to perform the expensive calculation of the partition function or the marginal distribution. Typical methods belonging to this category are Structured SVM (SSVM) [21], Max-Margin Markov Networks [22] *etc.*. One more benefit of such methods is that kernels can be naturally incorporated as analogous in SVM, *e.g.*, kernel CRF in [12].

For the inference task, *i.e.*, solving the argmax problem in Eq. (2.20) , applied methods generally depend on different applications. For instance, the GraphCut [23] and its "α-expansions" have been generally applied to solve the inference problem in the task of image segmentation [24–26]. Note that exact inference in general graphs is intractable. Hence endeavors have been devoted to approximate inference approaches, which can be roughly categorized into two veins: deterministic methods and sampling methods. In the survey of [27], Nowozin *et al.* provide a detailed investigation of structured learning, inference and their applications in computer vision.

In this thesis, we focus on the learning task and apply off-the-shelf methods to solve the inference task according to different applications. In more detail, for learning, we learn discrete CRF with non-linear potentials in the max-margin framework for image segmentation in Chapter 3 and Chapter 4. Specifically, in Chapter 3, we learn CRF with tree potentials while in Chapter 4, we learn CRF with CNN potentials. In Chapter 5, we propose a deep structured model to jointly learn continuous CRF with CNN potentials for depth estimation, with the optimization solved by the Stochastic Gradient Descent (SGD) algorithm. For inference, in Chapter 3, we apply GraphCut [23] since the energy function is submodular. In Chapter 4, we adopt the method of [28] to solve the MAP inference. While in Chapter 5, we solve a linear equation system for the MAP inference since we have closed form solution.

Next, we will introduce two important structured learning methods, namely, SSVM and CRF in the following sections.

2.2.1 Structured SVM

Structured SVM (SSVM) [21] is a well-known max-margin based structured learning method, which generalizes the classification SVM model presented in Sec. 2.1.1 to structured output predictions. Similarly, it learns a linear score function of the form:

$$f(\mathbf{x},\mathbf{y}) = \mathbf{w}^\top \Psi(\mathbf{x},\mathbf{y}), \tag{2.22}$$

where $\mathbf{w}$ are the model parameters; $\Psi(\cdot,\cdot)$ is a joint feature mapping function, which is typically constructed based on different applications. Then the optimal labelling can be found by solving:

$$\hat{\mathbf{y}} = g(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \mathbf{w}^\top \Psi(\mathbf{x},\mathbf{y}). \tag{2.23}$$

In the case of structured learning, the loss can not be simply quantified as 0/1 loss defined in Eq. (2.4) for classifications. In contrast, it considers a more general structured loss $\Delta(\mathbf{y},\hat{\mathbf{y}})$, whose design depends on applications. For instance, Hamming loss or weighted Hamming loss is typically used in the image segmentation task. In general, we have $\Delta(\mathbf{y},\mathbf{y}) = 0$ and $\Delta(\mathbf{y},\hat{\mathbf{y}}) > 0$ for any $\hat{\mathbf{y}} \neq \mathbf{y}$.

Given a set of *i.i.d.* training examples $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$, the margin-rescaling SSVM [29] learns the model parameters $\mathbf{w}$ by solving the following optimization problem:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|_2 + \frac{C}{m}\sum_i \xi_i \tag{2.24a}$$

$$\text{s.t.: } \mathbf{w}^\top \Big[\Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \Psi(\mathbf{x}^{(i)}, \mathbf{y})\Big]$$

$$\geq \mathit{\Delta}(\mathbf{y}^{(i)}, \mathbf{y}) - \xi_i, \forall \mathbf{y} \in \mathcal{Y} \text{ and } \forall i = 1, \dots, m, \tag{2.24b}$$

$$\boldsymbol{\xi} \geq \mathbf{0}, \tag{2.24c}$$

where C is the trade-off parameter; $\boldsymbol{\xi}$ are the slack variables; $\mathit{\Delta}(\cdot,\cdot) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the structured loss function associated with the predicted label and the ground-truth label. The constraints in Eq. (2.24b) ensures that for each training example $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, the score $\mathbf{w}^\top\Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ of the correct labelling $\mathbf{y}^{(i)}$ be greater than the score of all other labellings $\mathbf{w}^\top\Psi(\mathbf{x}^{(i)}, \mathbf{y})$ by a required margin. We can see that Eq. 2.24 is a natural generalization of Eq. 2.8 to structured labels $\mathbf{y}^{(i)}$ and structured loss $\mathit{\Delta}(\cdot,\cdot)$.

Cutting-Plane Training The optimization problem in Eq. (2.24) generally involves exponentially-sized or even infinite number of constraints, *i.e.*, $O(m|\mathcal{Y}|)$, which is typically hard to solve. For the case of linearly decomposable loss function $\mathit{\Delta}$, Taskar *et al.* [22] proposed a quadratic program formulation with only a polynomial number of constraints and variables. Tsochantaridis *et al.* [29] presented a cutting-plane [30] algorithm to solve Eq. (2.24) in polynomial time for general loss functions. In [31], Joachims *et al.* further proposed an 1-slack formulation of Eq. (2.24) solved by cutting-plane methods, which is of linear time complexity in the number of training examples. The 1-slack formulation is written as:

$$\min_{\mathbf{w},\xi} \ \frac{1}{2}\|\mathbf{w}\|_2 + C\xi \tag{2.25}$$

$$\text{s.t.: } \frac{1}{m}\mathbf{w}^\top \left[\sum_{i=1}^{m} r_i \Big[\Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \Psi(\mathbf{x}^{(i)}, \mathbf{y})\Big]\right]$$

$$\geq \frac{1}{m}\sum_{i=1}^{m} r_i \mathit{\Delta}(\mathbf{y}^{(i)}, \mathbf{y}) - \xi, \forall \mathbf{y} \in \mathcal{Y} \text{ and } \forall \boldsymbol{r} \in \{0,1\}^m. \tag{2.26}$$

Starting from an empty working set, the cutting-plane algorithm iteratively constructs a working set of constraints. In each iteration, the algorithm calculates the solution $\mathbf{w}, \boldsymbol{\xi}$ over the current working set, and finds the constraint that is most violated by the current solution for each example:

$$\mathbf{y}^{(i)\star} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \ \mathbf{w}^\top\Psi(\mathbf{x}^{(i)}, \mathbf{y}) + \mathit{\Delta}(\mathbf{y}^{(i)}, \mathbf{y}). \tag{2.27}$$

This procedure terminates when no more violated constraints can be found within the desired precision. We show in Chapter 3 that a cutting-plane algorithm can be applied to speedup training of our CRF model with tree potentials.

2.2.2 Conditional Random Fields

As one of the most popular probabilistic structured learning methods, Conditional Random Fields (CRF) model the input and the output variables through an undirected graph $G = (\mathcal{N}, \mathcal{S})$ with cliques $c \in \mathcal{C}$. Here $\mathcal{N}$ denotes the set of nodes $\mathcal{N} = \{1, 2, \dots, n\}$; $\mathcal{S}$ is the set of edges with $\mathcal{S} \subset \mathcal{N} \times \mathcal{N}$; $\mathcal{C}$ is the set of cliques in G. In the case of image segmentation, $\mathcal{N}$ is the node set composed of n (super-)pixels. Each node p is associated with a random variable y_p.

CRF model the conditional probability distribution in the following log-linear form:

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x};\mathbf{w}) &= \frac{1}{\mathrm{Z}(\mathbf{x})} \exp\big(\sum_c \mathbf{w}_c^\top \Psi_c(\mathbf{x},\mathbf{y})\big) \\ &= \frac{1}{\mathrm{Z}(\mathbf{x})} \prod_c \exp\big(\mathbf{w}_c^\top \Psi_c(\mathbf{x},\mathbf{y})\big), \qquad (2.28a) \\ \text{with } \mathrm{Z}(\mathbf{x}) &= \sum_{\mathbf{y}} \prod_c \exp\big(\mathbf{w}_c^\top \Psi_c(\mathbf{x},\mathbf{y})\big), \qquad (2.28b) \end{aligned}$$

where $\mathbf{w}$ is composed of all $\mathbf{w}_c$, which are model parameters; Ψ_c is the joint feature mapping function defined on the clique c; $\mathrm{Z}(\mathbf{x})$ is the normalization term, *i.e.*, the partition function. The prediction is then performed by solving the Maximum a Posterior (MAP) inference:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \Pr(\mathbf{y}|\mathbf{x};\mathbf{w}). \qquad (2.29)$$

Comparing Eq. (2.28) with Eq. (2.19), we can see that CRF can be naturally regarded as generalization of logistic regression to more general graphs. Analogous to logistic regression, the model parameters can be learned by maximizing the conditional log-likelihood (or minimizing the negative log-likelihood) of the training data:

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|_2 + \frac{C}{m} \sum_i \log \mathrm{Z}(\mathbf{x}^{(i)}) - \mathbf{w}^\top \Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}). \qquad (2.30)$$

One can refer to the comprehensive survey of [32], which provides a broad review on CRF and its applications.

2.2.2.1 Continuous Conditional Random Fields

Thus far, all we have been dealing with are discrete problems. There exist some vision applications of which the output domain is structured and continuous, *e.g.*, image denoising, depth estimation, *etc.*. For such kind of problems, continuous CRF can be naturally applied. In the community, CRF have been extensively studied for classification (discrete) problems, while less explored for regression (continuous) problems. One of the pioneering work on continuous CRF can be attributed to [33], in which it was proposed for global ranking in document retrieval. They show that the maximum likelihood optimization can be directly solved under certain constraints, in that the partition function can be analytically calculated.

Compared to the traditional discrete CRF models, since the label domain $\mathcal{Y}$ is continuous, the partition function in Eq. (2.28) therefore becomes an integral:

$$\mathrm{Z}(\mathbf{x}) = \int_{\mathbf{y}} \prod_{c} \exp(\mathbf{w}_c^{\top} \Psi_c(\mathbf{x}, \mathbf{y}))\mathrm{d}\mathbf{y}. \tag{2.31}$$

Under certain circumstances, the integral in Eq. (2.31) can be analytically calculated. This can give rise to some benefits in the course of optimization, which we will show in Chapter 5. Specifically, we jointly train continuous CRF with CNN in a unified framework to yield state-of-the-art depth estimation performance.

2.3 Ensemble Learning

Ensemble learning [34] has been extensively studied and applied in the machine learning community. The idea is to combine multiple weak hypotheses to form the final predictor for classification or regression problems. The weak hypotheses to be combined can be any weak models, *e.g.*, decision stumps, decision trees, or models trained using different subsets of features. As for the ensemble techniques, Bayesian modelling, bootstrapping, boosting are the most commonly applied methods. The simplest ensemble method is average voting, which is to averagely combine the weak hypotheses. It has been used as a general rule to boost the performance. Though widely applied in classification/regression models, ensemble learning is less explored for structured learning problems. In the case of classification or regression problems, it is straightforward to combine scalar outputs. However, it is less clear how to combine structured predictions, *e.g.*, semantic labelling masks in the image segmentation task through ensemble techniques. In the recent work of [35], Cortes *et al.* present a broad analysis of the problem of ensemble structured prediction, including a series of algorithms with learning guarantees.

We show in Chapter 3 how to apply ensemble learning of decision trees in structured learning scenario for image segmentation. Specifically, we model the CRF potentials as ensembles of decision trees, and jointly learn the trees and the ensemble parameters in the max-margin framework. The resulted optimization problem can be efficiently solved by combining a modified column generation and cutting-plane algorithms.

2.3.1 Column Generation Boosting

Column generation (CG) is an efficient algorithm for solving large-scale linear program (LP) problems. In [36], the authors proposed an LP formulation for boosting methods solved by CG, termed as LPBoost, which can also be applied to general LP formulated ensemble learning. This is relevant to our work in Chapter 3, which addresses a structured ensemble learning problem using CG. Therefore we here introduce LPBoost [36] in detail.

LPBoost targets at the binary classification problem, in which a training example is denoted by $\mathbf{x}$ with its label being $y \in \{-1, +1\}$. A weak learner/hypothesis is denoted as $\hbar$, which performs the following mapping:

$$\hbar : \mathbf{x} \to \{-1, +1\}, \tag{2.32}$$

with each $\hbar(\cdot) \in \mathcal{H}$, *i.e.*, the domain of all weak learners' output is $\mathcal{H}$. We use $\mathbf{H}(\mathbf{x}^{(i)})$ to denote the output column vector of all possible $\hbar(\mathbf{x}^{(i)})$:

$$\mathbf{H}(\mathbf{x}^{(i)}) = [\hbar_1(\mathbf{x}^{(i)}), \hbar_2(\mathbf{x}^{(i)}), \dots, \hbar_J(\mathbf{x}^{(i)})]^\top, \tag{2.33}$$

with J being the total number of weak learners, *i.e.*, the size of the domain $\mathcal{H}$.

To perform the binary classification, LPBoost aims to learn a hyperplane that separates the positive from the negative examples. The hyperplane is represented by a normal vector $\mathbf{w}$. Then the discriminative function is written as:

$$f(\mathbf{x}) = \operatorname{sgn}\left[\mathbf{w}^\top \mathbf{H}(\mathbf{x})\right] = \operatorname{sgn}\left[\sum_{j=1}^{J} w_j \hbar_j(\mathbf{x})\right]. \tag{2.34}$$

From Eq. (2.34), we can see that LPBoost attempts to learn a set of weak learners and their linear combination coefficients $\mathbf{w}$ to perform the final prediction.

To learn the discriminative function defined in Eq. (2.34), LPBoost solves the following LP optimization problem with soft margin:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \ \mathbf{1}^\top \mathbf{w} + \frac{C}{m}\sum_i \xi_i \tag{2.35a}$$

$$\text{s.t.:}\ y^{(i)}\mathbf{w}^\top \mathbf{H}(\mathbf{x}^{(i)}) + \xi_i \geq \mathbf{1}, \forall i = 1,\dots,m, \tag{2.35b}$$

$$\mathbf{w} \geq \mathbf{0}, \boldsymbol{\xi} \geq \mathbf{0}. \tag{2.35c}$$

Here C is the trade-off parameter; $\boldsymbol{\xi}$ are slack variables. The L_1 regularization on $\mathbf{w}$ enforces sparsity, which selects a small number of weak learners.

Since the size of the domain $\mathcal{H}$ can be infinitely large, which means that the dimension of $\mathbf{w}$ can be infinitely large, it is therefore intractable to solve the optimization in Eq. (2.35) using standard LP techniques. The basic idea of CG is to maintain a working set $\mathcal{W}_\mathbf{H}$ of the weak learners, and restrict the primal problem by only considering a subset of all the possible labellings based on the current working set, which is referred as the restricted master problem. Initially, the working set is empty and no weak learner is generated. Then during each iteration, CG selects a new weak learner and adds it to the working set $\mathcal{W}_\mathbf{H}$, and then solve the restricted master problem based on the current working set. To speedup convergence, the new weak learner is selected by finding the most violated constraint in the dual. The dual of Eq. (2.35) written as:

$$\max_{\boldsymbol{\lambda}} \ \sum_i \lambda_i \tag{2.36a}$$

$$\text{s.t.:}\ \sum_i \lambda_i y^{(i)} \hbar_j(\mathbf{x}^{(i)}) \leq 1, \forall j = 1,\dots,J, \tag{2.36b}$$

$$0 \leq \lambda_i \leq \frac{C}{m}, \forall i = 1,\dots,m, \tag{2.36c}$$

where $\boldsymbol{\lambda}$ are dual variables, with each λ_i associated with a constraint in Eq. (2.35b). Then the protocol for selecting a new weak learner is to find the $\hbar$ that most violates the constraint in Eq. (2.36b):

$$\hbar^\star(\cdot) = \operatorname*{argmax}_{\hbar(\cdot)\in\mathcal{H}} \sum_i \lambda_i y^{(i)} \hbar(\mathbf{x}^{(i)}). \tag{2.37}$$

If no weak learner violates the dual constraint Eq. (2.36b), then an optimal solution is obtained. This is the main idea of the LPBoost [36] and its extension [37]. Different from the generally considered LP problem for the binary classification, we will show in Chapter 3 that how we can apply a CG algorithm to solve a quadratic program (QP) problem in the structured learning scenario for image segmentation.

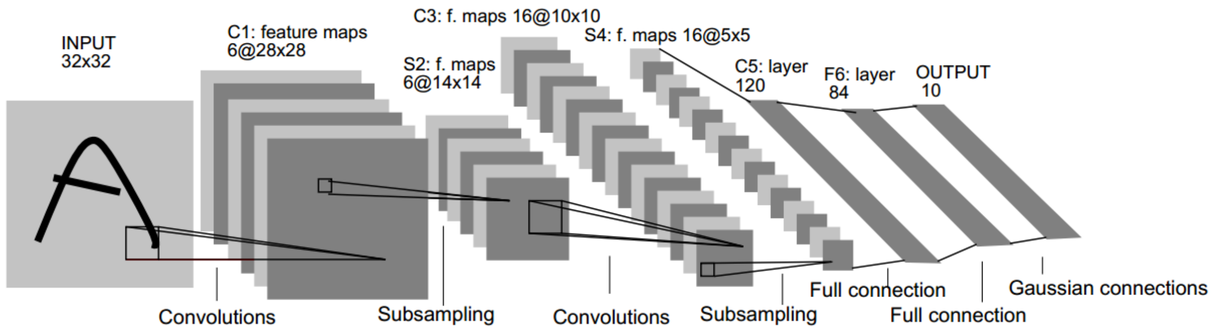


Figure 2.2: An illustration of the LeNet [2] for handwritten character recognition. Figure reproduced from [2].

2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [38] are special types of multi-layer neural networks, which were inspired by the locally sensitive property of the biological processes.

Fig. 2.3 shows an illustration of the notable LeNet [2] for handwritten character recognitions. As shown, a CNN is typically composed of multiple convolutional layers, pooling (subsampling) layers and fully connected layers.

- **Convolution** A convolutional layer performs convolutions over the input feature maps. It ensures local connectivity—neurons are connected to a small local region of the input, which is referred as the receptive field. This is intuitively originated from the fact that neighbouring image pixels are generally highly correlated. Network parameters are shared among all the receptive fields. A convolutional layer is typically followed by a nonlinear activation function for non-linearities, *e.g.*, sigmoid function ($f(x) = \tanh(x)$), rectified linear units abbreviated as ReLU ($f(x) = max\{0, x\}$).

- **Pooling** A pooling (or subsampling) layer takes blocks from the input feature maps and outputs a single value for each block. Typically used are max pooling and sum/mean pooling. It down-samples the feature maps and provides spatial invariance.

- **Fully connect** A fully-connected layer connects each of its neurons to all units in the input feature maps. Note that it can be regarded as a special case of the convolutional layer, with the receptive field size being the full size of the input feature maps. As there is only one receptive field (the whole input feature map region), there is no weight sharing in the fully connected layer, *i.e.*, each unit is connected to its own neurons.

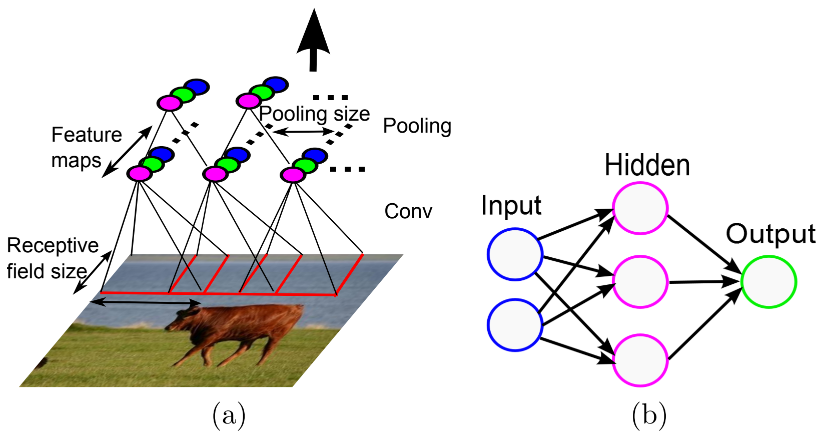


Figure 2.3: (a) An illustration of a single convolutional layer followed by a pooling layer with pooling size being 2; (b) An illustration of a fully connected layer with 3 hidden neurons.

Fig. 2.4 shows an illustration of the low-level, mid-level and high-level features learned from different layers of CNNs. As we can see, the low-level convolutional layer captures oriented edges and color patterns, while higher level layers capture component patterns or more high-level abstractions.

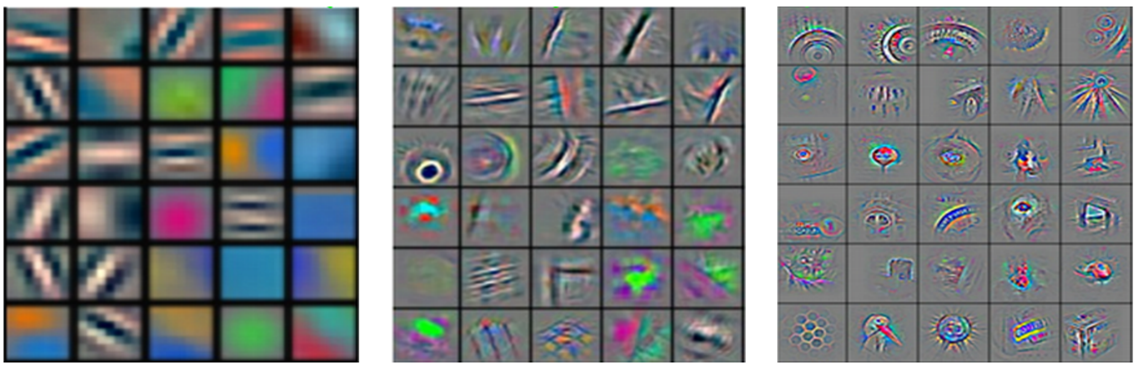
CNNs provide a general learning scheme for extracting hierarchical features from raw RGB image and have demonstrated huge potentials in various applications. However, they had been restricted to relatively small datasets in the past decades, which is mainly due to the limited computational resources and the unresolved overfitting dilemma. Until recently, the rapid development of high capacity computation power, especially GPU, makes it possible to train deep networks for large scale datasets. Moreover, the invention of the dropout [39] method provides an effective way for preventing overfittings. In [4], Krizhevsky *et al.* published the breakthrough work of training deep CNNs for ImageNet classifications. Since then, CNNs have been setting new records for diverse vision tasks, ranging from image classification [4], object detection [40] to scene labelling [41]. For more recent astounding results of CNNs related applications, one can refer to [42].

2.4.1 CNN for Structured Predictions

Though CNNs have been applied to a variety of classification problems and achieved huge success, they have been less explored for structured prediction problems. Combining CNNs training with structured learning has become an appealing direction. However, several challenges exit in this kind of combined learning models. First, training CNNs generally requires large amounts of training data to perform stochastic gradient descent (SGD). However, most of the current structured learning methods are not scalable enough. Second, the SGD based maximum likelihood learning involves calculating the gradients of the partition function, which is generally intractable. Therefore efficient approximation methods such as psudo-likelihood training or piecewise training need to be applied. Moreover, learning deep structured models calls for more efficient inference algorithms, since typically inference needs to be performed during each SGD iteration.

Existing methods of applying CNNs for structured learning typically use CNNs to model the potential functions of a CRF. A simple way is to incorporate structured information to a trained unary model as a post-processing step. These methods first train a CNN model or generate CNN features for constructing the unary potential, then incorporate spatial pairwise constraints to optimize the CRF loss. In [43], Chen *et al.* first train a fully convolutional CNN for pixel classification and then separately apply a dense CRF to refine the semantic labellings. In the work of [44], Tompson *et al.* present a hybrid architecture for combining CNNs and MRF for human pose estimation by first training a unary model and a spatial model separately and then fine-tuning them in a post-processing step. Our work in Chapter 4 also belongs to this category. Specifically, we transfer a pre-trained CNN model to extract features for constructing unary potentials, and then perform CRF learning for predicting semantic labels of images.

Recently, more endeavours are made towards joint learning of CNNs and graphical models [44–48]. Joint learning means that all the parameters of the CNNs potentials functions are learned simultaneously to optimize the CRF loss. These methods generally enable end-to-end learning, which typically produce better results than the separate learning methods. In [46], Zheng *et al.* propose to implement the mean field inference in CRF as Recurrent Neural Networks (RNNs) to facilitate the end-to-end joint learning. The work in [45] explores joint learning of MRF and deep models for predicting words from noisy images and tagging photographs. In [47], Lin *et al.* propose an efficient piecewise learning approach to jointly learn the unary and pairwise CNNs potentials in CRF for semantic segmentation. They further propose in [48] to directly learn CNNs message estimators in the message passing inference rather than learning potential functions. One of the benefits of this method is that it explicitly incorporates the expected number of inference iterations into the learning procedure, which reveals a new direction for scalable learning of deep structured models. Our work in Chapter 5 belongs to one of these early attempts. In more detail, we propose a general framework for joint training of CNNs and continuous CRF, which has the benefits of exact learning and closed-form inference solutions. The proposed method is applied for the task of depth estimations from single monocular images, and demonstrates superior performance over state-of-the-art methods.



(a) (b) (c)

Figure 2.4: An illustration of the (a) low-level, (b) mid-level and (c) high-level features learned from different layers of CNN models. Figure reproduced from [3].

Chapter 3

CRF Learning with Tree Potentials for Image Segmentation

In this chapter, we propose a new approach for image segmentation, which exploits the advantages of both conditional random fields (CRF) and decision trees. In the literature, the potential functions of CRF are mostly defined as a *linear* combination of some predefined parametric models, and then methods like SSVM are applied to learn those linear coefficients. We instead formulate the unary and pairwise potentials as nonparametric forests—ensembles of decision trees, and learn the ensemble parameters and the trees in a unified optimization problem within the max-margin framework. In this fashion, we easily achieve *nonlinear* learning of potential functions on both unary and pairwise terms in CRF. Moreover, our method enables learning class-wise decision trees for each object that appears in the image, which we call *object-aware* potentials learning. Due to the rich structure and flexibility of decision trees, our approach is more powerful in modelling complex data likelihoods and label relationships. The resulting optimization problem is very challenging because it can have exponentially many variables and constraints. We show that this challenging optimization can be efficiently solved by combining a modified column generation and cutting-planes techniques. Experimental results on both binary (Graz-02, Weizmann horse, Oxford flower) and multi-class (MSRC-21) segmentation datasets demonstrate the power of the learned nonlinear nonparametric potentials.

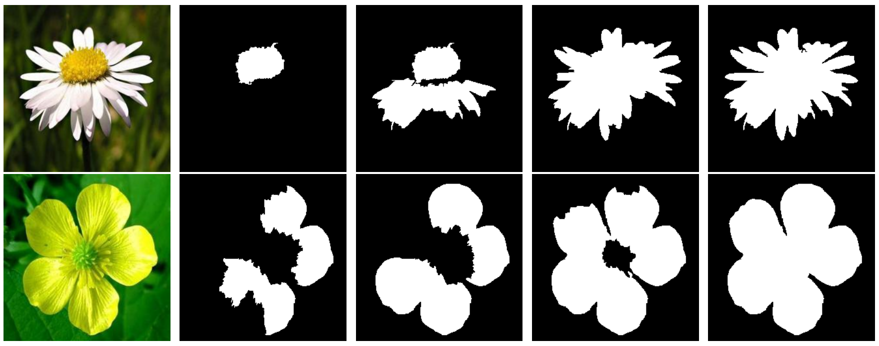


Figure 3.1: Segmentation examples produced by our model on images from the Oxford 17 Flower dataset with different column generation iterations. From left to right: Test images, 2nd, 4th, 6th, and 10th iteration.

3.1 Introduction

The goal of object segmentation is to produce a pixel level segmentation of different object categories. It is challenging as the objects may appear in various backgrounds and in different visual conditions. Early attempts have been made in geometric based methods, or use simple statistical learning models to seek an optimal labelling for each pixel (or super-pixels). Instead of looking at each pixel or its neighbourhood, looking for a joint optimal assignment for all pixels (or superpixels) has become increasingly popular via minimizing energy functions or maximizing some potential functions. As one of these methods, CRFs [49] model the conditional distribution of labels given observations, and represents the state-of-the-art in image/object segmentation [24, 26, 50–52]. The max-margin principle has also been applied to predict structured outputs, including SSVM [29], and max-margin Markov networks [22]. These three methods share similarities when viewed as optimisation problems using different loss functions. Szummer *et al.* [24] proposed to learn linear coefficients of CRF potentials using SSVM and graph cuts. To date, most of these methods assume a pre-defined parametric model for the potential functions, and typically only the linear coefficients of the parametric model are learned. This can greatly limit the flexibility of model capability of CRF, and thus calls for effective methods to incorporate nonlinear nonparametric models for learning the potential functions in CRF.

As in standard SVM, nonlinearity can be achieved by introducing nonlinear kernels for SSVM. The time complexity of nonlinear SVM is roughly $O(n^{3.5})$ with n being the number of training data examples. This time complexity is problematic for SSVM, where the number of constraints grows exponentially in the description length of the label $\mathbf{y}$. Moreover, nonlinear functions can significantly slow down the test time in most cases. Because of these reasons, currently most SSVM applications use linear kernels (or linear parametric potential functions in CRF), despite the fact that nonlinear functions usually deliver more promising prediction accuracy. In this work, we address this issue by combining CRF with nonparametric decision trees. Both CRF and decision trees have gained tremendous success in computer vision. Decision trees are capable of modelling complex relations and generalise well on test data. Unlike kernel methods, decision trees are fast to evaluate and can be used to select informative features.

In this work, we propose to use ensembles of decision trees to map the image content to both the data terms and the pairwise interaction values in CRFs. The proposed method is termed CRFTree. We formulate both the unary and pairwise potentials as *nonparametric* forests—ensembles of decision trees, and learn the ensemble parameters and the trees in a single framework. In this way, nonlinearity is easily introduced into CRF learning without confronting the kernel dilemma. Furthermore, we can learn class-wise decision trees for each object, which can be called object-aware potentials learning. Due to the rich structure and flexibility of decision trees, our approach is more powerful in modelling complex data likelihoods and label relationships. The resulting optimization problem is very challenging in the sense that it can have exponentially or even infinitely many variables and constraints. Our main contributions are thus as follows.

1. We formulate the unary and pairwise potentials as ensembles of decision trees, and show how to jointly learn the ensemble parameters and the trees in a unified optimization problem within the max-margin framework. In this fashion, we achieve nonlinear potentials learning both on the unary and pairwise terms.
2. We learn class-wise decision trees (potentials) for each object that appears in the image. In other words, our potential learning is object-aware.
3. We show how the training of the proposed CRFTree can be performed efficiently. In particular, we combine column generation and cutting-planes techniques to approximately solve the resulting optimization problem, which can involve exponentially many variables and constraints.
4. We empirically demonstrate that CRFTree outperforms existing methods for image segmentation. On both binary and multi-class segmentation datasets we show the advantages of the learned nonlinear nonparametric potentials of decision trees.

3.1.1 Related Work

We briefly review some work that is relevant to ours. A few attempts have been made to apply nonlinear kernels in SSVM. The authors of [53] and [54] developed sampled cuts based methods for training SSVM with kernels. Sampled cuts methods were originally proposed for standard kernel SVM. When applied to SSVM, performance is compromised

[26]. In [25], image-mask pair kernels are designed to exploit image-level structural information for object segmentation (their kernels are restricted to the unary term). Although not in the max-margin framework, the kernel CRF proposed in [12] incorporates kernels into the CRF learning. The authors only demonstrated the efficacy of their method on a synthetic and a small scale protein dataset. To sum up, these approaches are hampered by the heavy computation complexity. Furthermore, it is not a trivial task to design appropriate kernels for structured problems. Recently, Lucchi et al. [26] proposed a two-step solution to tackle this problem. They train a linear SSVM by using kernelized feature vectors that are obtained from training a standard non-linear kernel SVM. They experimentally demonstrated that the kernel transferred linear SVM achieved similar performance as the Gaussian SVM. However, this approach is heuristic and it cannot be shown theoretically that their formulation approximates a nonlinear SSVM. Besides, their method consumes extra usage of memory and training time since the dimension of the transformed features equals the number of support vectors, while the latter is linearly proportional to the size of the training data [55]. Moreover, compared to the above mentioned works of [25] and [26], we achieve nonlinear learning both on the unary and the pairwise terms while theirs are limited to nonlinear unary potential learning. The recent work of [56] generalizes standard boosting methods to structured learning, which shares similarities with our work here. However, our method bears critical differences from theirs: 1) We design a column generation method for non-linear potentials learning in CRF directly from the SSVM formulation. It enables the use of specialized SVM QP solvers, which is in practice faster than solving an LP problem. In contrast, [56] needs to solve an LP problem during each column generation iteration, which is generally not scalable. 2) We develop a CRF learning method for multi-class image segmentation, while [13] only shows CRF learning for binary foreground/background segmentation. 3) We learn class-wise decision trees (potentials) for each object that appears in the image. This is different from [56]. The work of decision tree fields [57] is close to ours in that they also use decision trees to model the pairwise potentials. The major difference is that in [57] the learning is implemented by optimizing the pseudo-likelihood objective function of CRF. By contrast, in our CRFTree, we learn the nonparametric potentials (represented by ensembles of trees) in the max-margin structured output learning framework, which can be seen as a generalization of SSVM.

3.2 Segmentation Using CRF Models

Given an image instance $\mathbf{x}$ and its corresponding labelling $\mathbf{y}$, CRF [49] models the conditional distribution of the form

$$\Pr(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{1}{\mathrm{Z}(\mathbf{x})}\exp(-E(\mathbf{y},\mathbf{x};\mathbf{w})). \tag{3.1}$$

where $\mathbf{w}$ are parameters and Z the partition function. The energy E of an image $\mathbf{x}$ with segmentation labels $\mathbf{y}$ over the nodes (superpixels) $\mathcal{N}$ and edges $\mathcal{S}$, takes the following form:

$$E(\mathbf{y},\mathbf{x};\mathbf{w}) = \sum_{p\in\mathcal{N}} U(y_p,\mathbf{x};\mathbf{w}) + \sum_{(p,q)\in\mathcal{S}} V(y_p,y_q,\mathbf{x};\mathbf{w}). \tag{3.2}$$

Here $\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y}$; U and V are the unary and pairwise potentials, both of which depend on the observations as well as the parameter $\mathbf{w}$. CRF seeks an optimal labeling that achieves maximum a posterior (MAP), which mainly involves a two-step process [24]: 1) Learning the model parameters from the training data; 2) Inferring a most likely label for the test data given the learned parameters. The segmentation problem thus reduced to minimizing the energy (or cost) over $\mathbf{y}$ by the learned parameters $\mathbf{w}$, which is written as:

$$\mathbf{y}^* = \operatorname*{argmin}_{\mathbf{y}\in\mathcal{Y}} E(\mathbf{y},\mathbf{x};\mathbf{w}). \tag{3.3}$$

When the energy function is submodular, this inference problem can be efficiently solved via graph cuts [24].

3.3 Learning Tree Potentials in CRF

We present the details of our method in this section by first formulating the energy functions and then showing how to learn decision tree potentials in the max-margin framework.

3.3.1 Energy Formulation

Given the energy function in Eq. (3.2), we show how to construct the unary and pairwise potentials using decision trees. We denote $\mathbf{x}_p$ as the features of superpixel p ($p = 1,\dots,n$), with its label $y_p\in\{1,\dots,K\}$, where K is the number of classes. Let $\mathcal{H}$ be a set of decision trees, which can be infinite. Each $\hbar_j^{(1)}(\cdot)\in\mathcal{H}$ takes $\mathbf{x}_p$ as input, and

$\hbar_j^{(2)}(\cdot,\cdot) \in \mathcal{H}$ takes a pair $(\mathbf{x}_p, \mathbf{x}_q)$ as input to output $\{0,1\}$. We introduce $(K+1)$ groups of decision trees, in which K groups are for the unary potential and one group for the pairwise potential. For the unary potential, the K groups of decision trees are denoted by $\mathbf{H}_c^{(1)}(c = 1, \ldots, K)$, which correspond to K categories. Each $\mathbf{H}_c^{(1)}$ is associated with the c-th class. In other words, for each class, we maintain its own unary feature mappings. Each group of decision trees for the unary potential can be written as: $\mathbf{H}_c^{(1)} = [\hbar_{c1}^{(1)}, \hbar_{c2}^{(1)}, \ldots]^\top$, which are the output of decision trees: $\hbar_{cj}^{(1)}$. All decision trees of the unary potential are denoted by $\mathbf{H}^{(1)} = [\mathbf{H}_1^{(1)}, \mathbf{H}_2^{(1)}, \ldots, \mathbf{H}_K^{(1)}]$. Accordingly, for the pairwise potential, the group of decision trees is denoted by $\mathbf{H}^{(2)}$, and $\mathbf{H}^{(2)} = [\hbar_1^{(2)}, \hbar_2^{(2)}, \ldots]^\top$ being the output of all $\hbar_j^{(2)}$. The whole set of decision trees is denoted by $\mathbf{H} = [\mathbf{H}^{(1)}, \mathbf{H}^{(2)}]$. We then construct the unary and pairwise potentials as

$$U(y_{(p)}, \mathbf{x}) = \mathbf{w}_{y_p}^{(1)\top} \mathbf{H}_{y_p}^{(1)}(\mathbf{x}_p). \tag{3.4}$$

$$V(y_{(p)}, y_{(q)}, \mathbf{x}) = \mathbf{w}^{(2)\top} \mathbf{H}^{(2)}(\mathbf{x}_p, \mathbf{x}_q)\delta(y_p \neq y_q). \tag{3.5}$$

where $\delta(\cdot)$ is an indicator function which equals 1 if the input is true and 0 otherwise. Then the energy function in Eq. (3.2) can be written as:

$$\begin{aligned} E(\mathbf{y}, \mathbf{x}; \mathbf{w}, \mathbf{H}) = & \sum_{p \in \mathcal{N}} \mathbf{w}_{y_p}^{(1)\top} \mathbf{H}_{y_p}^{(1)}(\mathbf{x}_p) \\ & + \sum_{(p,q) \in \mathcal{S}} \mathbf{w}^{(2)\top} \mathbf{H}^{(2)}(\mathbf{x}_p, \mathbf{x}_q)\delta(y_p \neq y_q). \end{aligned} \tag{3.6}$$

Next we show how to learn these decision tree potentials in the max-margin framework.

3.3.2 Learning CRF in the Max-Margin Framework

Applying the max-margin based CRF learning is to solve the following optimization:

$$\begin{aligned} \min_{\mathbf{w}, \boldsymbol{\xi}} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + \tfrac{C}{m} \sum_i \xi_i \\ \text{s.t.:} \quad & E(\mathbf{y}, \mathbf{x}_i; \mathbf{w}, \mathbf{H}) - E(\mathbf{y}_i, \mathbf{x}_i; \mathbf{w}, \mathbf{H}) \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \\ & \qquad \forall i = 1, \ldots, m, \text{ and } \forall \mathbf{y} \in \mathcal{Y}; \\ & \boldsymbol{\xi} \geq 0. \end{aligned} \tag{3.7}$$

where $\Delta : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is a loss function associated with the prediction and the true label mask. In general, we have $\Delta(\mathbf{y}, \mathbf{y}) = 0$ and $\Delta(\mathbf{y}, \mathbf{y}') > 0$ for any $\mathbf{y}' \neq \mathbf{y}$. Intuitively, the optimization in Eq. (3.7) is to encourage the energy of the ground truth label $E(\mathbf{y}_i, \mathbf{x}_i; \mathbf{w})$ to be lower than any other *incorrect* labels $E(\mathbf{y}, \mathbf{x}_i; \mathbf{w})$ by at least a margin $\Delta(\mathbf{y}_i, \mathbf{y})$.

To learn the potential functions we proposed in Sec. 3.3.1 in the max-margin framework, we introduce the following definitions. For the unary part, we define $\mathbf{w}^{(1)} = \mathbf{w}_1^{(1)} \odot \mathbf{w}_2^{(1)} \odot \ldots \odot \mathbf{w}_K^{(1)}$, where $\odot$ stacks two vectors, and

$$\Psi^{(1)}(\mathbf{y}, \mathbf{x}; \mathbf{H}^{(1)}) = \sum_{p \in \mathcal{N}} \mathbf{H}_{y_p}^{(1)}(\mathbf{x}_p) \otimes y_p. \tag{3.8}$$

where $\otimes$ denotes the tensor operation (*e.g.*, $\mathbf{x}_p \otimes y_p = [\delta(y_p = 1)\mathbf{x}_{p\top}, \ldots, \delta(y_p = K)\mathbf{x}_{p\top}]^\top$). Recall that $\mathbf{x}_p$ denotes the p-th superpixel of the image $\mathbf{x}$. Here, $\Psi^{(1)}$ acts as the unary feature mapping. Clearly we have:

$$\mathbf{w}^{(1)\top} \Psi^{(1)}(\mathbf{y}, \mathbf{x}; \mathbf{H}^{(1)}) = \sum_{p \in \mathcal{N}} U(y_p, \mathbf{x}). \tag{3.9}$$

For the pairwise part, we define the pairwise feature mapping as:

$$\Psi^{(2)}(\mathbf{y}, \mathbf{x}; \mathbf{H}^{(2)}) = \sum_{(p,q) \in \mathcal{S}} \mathbf{H}^{(2)}(\mathbf{x}_p, \mathbf{x}_q) \delta(y_p \neq y_q). \tag{3.10}$$

Then we have the following relation:

$$\mathbf{w}^{(2)\top} \Psi^{(2)}(\mathbf{y}, \mathbf{x}; \mathbf{H}^{(2)}) = \sum_{(p,q) \in \mathcal{S}} V(y_p, y_q, \mathbf{x}). \tag{3.11}$$

We further define $\mathbf{w} = \mathbf{w}^{(1)} \odot \mathbf{w}^{(2)}$, and the joint feature mapping as

$$\Psi(\mathbf{y}, \mathbf{x}; \mathbf{H}) = \Psi^{(1)}(\mathbf{y}, \mathbf{x}; \mathbf{H}^{(1)}) \odot \Psi^{(2)}(\mathbf{y}, \mathbf{x}; \mathbf{H}^{(2)}). \tag{3.12}$$

With the definitions of $\mathbf{w}$ and Ψ, the energy function can then be written as:

$$\begin{aligned} E(\mathbf{y}, \mathbf{x}; \mathbf{w}, \mathbf{H}) = &\sum_{p \in \mathcal{N}} U(y_p, \mathbf{x}; \mathbf{w}, \mathbf{H}^{(1)}) \\ &+ \sum_{(p,q) \in \mathcal{S}} V(y_p, y_q, \mathbf{x}; \mathbf{w}, \mathbf{H}^{(2)}) \\ = &\ \mathbf{w}^\top \Psi(\mathbf{y}, \mathbf{x}; \mathbf{H}). \end{aligned} \tag{3.13}$$

Now we can apply the max-margin framework to learn CRF using the proposed energy functions by rewriting the optimization problem in Eq. (3.7) as:

$$\begin{aligned}
&\min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|_2^2 + \tfrac{C}{m}\sum_i \xi_i \\
&\text{s.t.:}\ \mathbf{w}^\top\left[\Psi(\mathbf{y},\mathbf{x}_i;\mathbf{H}) - \Psi(\mathbf{y}_i,\mathbf{x}_i;\mathbf{H})\right] \geq \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i, \\
&\qquad\qquad \forall i = 1,\dots,m, \ \text{and} \ \forall \mathbf{y} \in \mathcal{Y}; \\
&\qquad \mathbf{w} \geq 0, \boldsymbol{\xi} \geq 0.
\end{aligned} \tag{3.14}$$

Note that we add the $\mathbf{w} \geq 0$ constraint to ensure submodular property of our energy functions, which we will discuss the details later in Sec. 3.3.3. Up until now, we are ready to learn $\mathbf{w}$ and Ψ (or $\mathbf{H}$) in a single optimization problem formulated in Eq. (3.14), but it is not clear how. Next we demonstrate how to solve the optimization problem in Eq. (3.14) by using column generation and cutting-plane.

3.3.3 Learning Tree Potentials Using Column Generation

We aim to learn a set of decision trees $\mathbf{H}$ and the potential parameter $\mathbf{w}$ by solving the optimization problem in Eq. (3.14). However, jointly learning $\mathbf{H}$ and $\mathbf{w}$ is generally difficult. One reason for this difficulty is that the number of constraints grows exponentially with the CRF size, and another reason is that there are infinitely many decision trees. We here propose to apply column generation techniques [36, 37] to alternatively construct the set of decision trees and solve for $\mathbf{w}$. From the point of view of column generation, the dimension of primal variable $\mathbf{w}$ is infinitely large; column generation is to iteratively select (generate) variables for solving the optimization. In our case, infinitely many dimension of $\mathbf{w}$ corresponds to infinitely many decision trees, thus we iteratively generate decision trees to solve the optimization.

Basically, we construct a working set of decision trees (denoted as $\mathcal{W}_\mathbf{H}$). For each column generation iteration we perform two steps. In the first step, we generate new decision trees and add to $\mathcal{W}_\mathbf{H}$. In the second step, we solve a restricted optimization problem in Eq. (3.14) on the current working set $\mathcal{W}_\mathbf{H}$ to obtain the solution of $\mathbf{w}$. We repeat these two steps until convergence. Next we describe how to generate decision trees in a principal way by using the dual solution of the optimization in Eq. (3.14), which is similar to the conventional column generation technique. First we derive the Lagrange dual problem of Eq. (3.14). The Lagrangian of Eq. (3.14) can be written as:

$$\begin{aligned} L = \; & \frac{1}{2}\left\|\mathbf{w}\right\|_2^2 + \tfrac{C}{m}\sum_i \xi_i - \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} \cdot \Big\{ \mathbf{w}^\top \left[\Psi(\mathbf{y},\mathbf{x}_i;\mathbf{H}) - \Psi(\mathbf{y}_i,\mathbf{x}_i;\mathbf{H})\right] \\ & - \varDelta(\mathbf{y}_i,\mathbf{y}) + \xi_i \Big\} - \boldsymbol{\theta}^\top \mathbf{w} - \boldsymbol{\beta}^\top \boldsymbol{\xi}, \end{aligned} \tag{3.15}$$

where $\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\beta}$ are Lagrange multipliers: $\boldsymbol{\lambda} \geq 0, \boldsymbol{\theta} \geq 0, \boldsymbol{\beta} \geq 0$. For ease of notation, we give the following definition:

$$\delta\Psi_i(\mathbf{y}) \equiv \Psi(\mathbf{y},\mathbf{x}_i;\mathbf{H}) - \Psi(\mathbf{y}_i,\mathbf{x}_i;\mathbf{H}). \tag{3.16}$$

At optimum, the first derivative of the Lagrangian w.r.t the primal variables must vanish,

$$\begin{aligned} \frac{\partial L}{\partial \xi_i} = 0 \implies & \frac{C}{m} - \sum_{\mathbf{y}} \lambda_{(i,\mathbf{y})} - \beta_i = 0 \\ \implies & 0 \leq \sum_{\mathbf{y}} \lambda_{(i,\mathbf{y})} \leq \frac{C}{m}; \end{aligned} \tag{3.17}$$

and,

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} = 0 \implies & \mathbf{w} - \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} \delta\Psi_i(\mathbf{y}) - \boldsymbol{\theta} = 0 \\ \implies & \mathbf{w} = \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} \delta\Psi_i(\mathbf{y}) + \boldsymbol{\theta}, \\ \implies & \mathbf{w} \geq \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} \delta\Psi_i(\mathbf{y}) \end{aligned} \tag{3.18}$$

By substituting them into Eq. (3.15), the dual problem of Eq. (3.14) can be written as:

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} \varDelta(\mathbf{y}_i,\mathbf{y}) - \frac{1}{2}\left[\sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} \left[\Psi(\mathbf{y},\mathbf{x}_i;\mathbf{H}) - \Psi(\mathbf{y}_i,\mathbf{x}_i;\mathbf{H})\right] - \boldsymbol{\theta}\right]^2 \\ \text{s.t.:} \quad & 0 \leq \textstyle\sum_{\mathbf{y}} \lambda_{(i,\mathbf{y})} \leq \tfrac{C}{m}, \forall i = 1,\dots,m; \\ & \boldsymbol{\theta} \geq 0, \boldsymbol{\lambda} \geq 0. \end{aligned}$$

Here $\boldsymbol{\theta},\boldsymbol{\lambda}$ are the dual variables. When using column generation technique, one need to find the most violated constraint in the dual. However, the constraints of the dual problem do not involve decision trees $\mathbf{H}$. Instead of examining the dual constraint, we inspect the KKT condition, which is an important difference compared to existing column generation techniques. According to the KKT condition, when at optimal, the following condition holds for the primal solution $\mathbf{w}$ and the current working set $\mathcal{W}_{\mathbf{H}}$:

$$\mathbf{w} \geq \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} [\Psi(\mathbf{y}, \mathbf{x}; \mathbf{H}) - \Psi(\mathbf{y}_i, \mathbf{x}; \mathbf{H})]. \tag{3.19}$$

All of those generated $\mathbf{H} \in \mathcal{W}_{\mathbf{H}}$ satisfy the above condition. Obviously, generating new decision trees which most violate the above condition would contribute the most to optimization of Eq. (3.14). Hence the strategy of generating new decision trees is to solve the following problem:

$$\mathbf{H}^\star = \operatorname*{argmax}_{\mathbf{H}} \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} [\Psi(\mathbf{x}_i, \mathbf{y}; \mathbf{H}) - \Psi(\mathbf{x}_i, \mathbf{y}_i; \mathbf{H})]. \tag{3.20}$$

Then $\mathbf{H}^\star$ is added to the current working set $\mathcal{W}_{\mathbf{H}}$. If $\mathbf{H}^\star$ still satisfy the condition in Eq. (3.19), the current solution of $\mathbf{H}$ and $\mathbf{w}$ are already the globally optimal one.

The optimization in Eq. (3.20) for generating new decision trees can be independently decomposed into solving the unary part and the pairwise part. Hence $\mathbf{H}^\star$ can be written as: $\mathbf{H}^\star = [\mathbf{H}^{(1)\star}, \mathbf{H}^{(2)\star}]$. For the unary part, we learn class-wise decision trees, namely, we generate K decision trees corresponding to K categories at each column generation iteration. Hence $\mathbf{H}^{(1)\star}$ is composed of K decision trees: $\mathbf{H}^{(1)\star} = [\hbar_1^{(1)\star}, \dots, \hbar_K^{(1)\star}]$. More specifically, according to the definition of $\Psi(\mathbf{y}, \mathbf{x})$ in Eq. (3.12), we solve the following K problems:

$$\begin{aligned} &\forall c = 1, \dots, K : \\ \hbar_c^{(1)\star}(\cdot) &= \operatorname*{argmax}_{\hbar \in \mathcal{H}} \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} \Bigg[\sum_{\substack{p \in \mathcal{N}, \\ y_p = c}} \hbar_{y_p}^{(1)}(\mathbf{x}_i^p) - \sum_{\substack{p \in \mathcal{N}, \\ y_i^p = c}} \hbar_{y_i^p}^{(1)}(\mathbf{x}_i^p) \Bigg] \\ &= \operatorname*{argmax}_{\hbar \in \mathcal{H}} \sum_{i,\mathbf{y}} \Bigg[\sum_{\substack{p \in \mathcal{N}, \\ y_p = c}} \underbrace{\lambda_{(i,\mathbf{y})} \hbar_{y_p}^{(1)}(\mathbf{x}_i^p)}_{\text{positive}} - \sum_{\substack{p \in \mathcal{N}, \\ y_i^p = c}} \underbrace{\lambda_{(i,\mathbf{y})} \hbar_{y_i^p}^{(1)}(\mathbf{x}_i^p)}_{\text{negative}} \Bigg]. \end{aligned} \tag{3.21}$$

To solve the above optimization problems, we here train K weighted decision tree classifiers. Specifically, when training decision trees for the c-th class, the training data is composed of those superpixels whose ground truth label or predicted label is equal to the category label c. Since the output of the decision tree is in $\{0, 1\}$ and $\lambda_{(i,\mathbf{y})} \geq 0$, the maximization in Eq. (3.21) is achieved if $\hbar_c^{(1)}$ outputs 1 for each of the superpixel p with $y_p = c$, and outputs 0 for each of the superpixel p with $y_i^p = c$. Therefore, as indicated by the horizontal curly braces in Eq. (3.21), superpixels with the predicted labels of category c are used as positive training examples, while superpixels with ground truth labels of category c are used as negative training examples. The dual solution $\boldsymbol{\lambda}$ serve as weightings of the training data. It is also worth noting that if we search over all the c categories to get the most violated constraint (by adding c under the argmax), we are learning non-object-aware decision trees, which is commonly done in most boosting methods, *e.g.*, [37], for solving multi-class problems. We will show the advantages of learning object-aware potentials in the experiment part.

For the pairwise part, we generate one decision tree in each column generation iteration, hence $\mathbf{H}^{(2)\star}$ can be written as $\mathbf{H}^{(2)\star} = [\hbar^{(2)\star}]$, the new decision tree for the pairwise part is generated as:

$$\hbar^{(2)\star}(\cdot,\cdot) = \operatorname*{argmax}_{\hbar\in\mathcal{H}} \sum_{i,\mathbf{y}} \lambda_{(i,\mathbf{y})} \Bigg[\sum_{(p,q)\in\mathcal{S}} \underbrace{\hbar^{(2)}(\mathbf{x}_p,\mathbf{x}_q)\delta(y_p \neq y_q)}_{\text{positive}} - \sum_{(p,q)\in\mathcal{S}} \underbrace{\hbar^{(2)}(\mathbf{x}_p,\mathbf{x}_q)\delta(y_i^p \neq y_i^q)}_{\text{negative}} \Bigg]. \tag{3.22}$$

Similar to the unary case, we train a weighted decision tree classifier with $\boldsymbol{\lambda}$ as training example weightings. The positive and negative training data are indicated by the horizontal curly braces in Eq. (3.22). $\hbar^{(2)}$ is the response of a decision tree applied on the pairwise features constructed by two neighbouring superpixels ($\mathbf{x}_p$, $\mathbf{x}_q$), *e.g.*, color differences or shared boundary lengths.

With the above analysis, we can now apply column generation to jointly learn the decision trees $\mathbf{H}^{(1)}, \mathbf{H}^{(2)}$ and $\mathbf{w}$. The column generation (CG) procedure iterates the following two steps:

1) Solve Eq. (3.21), Eq. (3.22) to generate decision trees $\mathbf{H}^{(1)\star}$, $\mathbf{H}^{(2)\star}$;

2) Add $\mathbf{H}^{(1)\star}$ and $\mathbf{H}^{(2)\star}$ to working set $\mathcal{W}_{\mathbf{H}}$ and resolve for the primal solution $\mathbf{w}$ and dual solution $\boldsymbol{\lambda}$.

We show some segmentation examples on the Oxford flower dataset produced by our method with different CG iterations in Fig. 3.1. As can be seen, our method refines the segmentation with the increase of CG iterations. Since this dataset is relatively simple, a few CG iterations are enough to get satisfactory results.

For solving the primal problem in the second step, it involves a large number of constraints due to the large output space $\{\mathbf{y} \in \mathcal{Y}\}$. We next show how to apply the cutting-plane technique [58] to efficiently solve this problem.

3.3.4 Speeding up Optimization Using Cutting-Plane

To apply cutting-plane for solving the optimization in Eq. (3.14), we first derive its 1-slack formulation. The 1-slack SSVM formulation was first introduced by [58]. The

Algorithm 1: CRFTree using column generation

```
1. **Input:** training examples $(\mathbf{x}_1; \mathbf{y}_1), (\mathbf{x}_2; \mathbf{y}_2), \cdots$; maximum iteration number.
2. **Initialize** $(\boldsymbol{\lambda}, \mathbf{y})$, and decision tree working set $\mathcal{W}_{\mathbf{H}} \leftarrow \emptyset$
3. **Repeat**
4.   – Find decision trees $\mathbf{H}^\star$ by solving Eq. (3.21), Eq. (3.22). Add $\mathbf{H}^\star$ to working set $\mathcal{W}_{\mathbf{H}}$.
5.   – Call Alg. 11 using working set $\mathcal{W}_{\mathbf{H}}$ to solve for $\mathbf{w}$ and $\boldsymbol{\lambda}$.
6. **Until** the maximum iteration is reached.
7. **Output:** $\mathbf{w}$, $\mathbf{H} \in \mathcal{W}_{\mathbf{H}}$.
```

Algorithm 2: Cutting-planes for solving the 1-slack primal

```
1: **Input:** cutting-plane termination threshold $\epsilon_{\mathrm{cp}}$, and inputs from Alg. 7.
2: **Initialize:** working set $\mathcal{W} \leftarrow \emptyset$; $\boldsymbol{r}$.
3: **Repeat**
4:    – $\mathcal{W} \leftarrow \mathcal{W} \cup \{(r_1, \ldots, r_m, \mathbf{y}_1^\star, \ldots, \mathbf{y}_m^\star)\}$.
5:    – Obtain primal and dual solutions $\mathbf{w}, \xi$; $\boldsymbol{\lambda}$ by solving Eq. (3.23) on $\mathcal{W}$.
6:    – **For** $i = 1, \ldots, m$
7:          Solve the inference problem in Eq. (3.24) using Graph-Cut to find the most violated $\mathbf{y}_i^\star$.
9:    – **End for**
10: **Until**
$\frac{1}{m}\mathbf{w}^\top \left[\sum_{i=1}^m r_i \left[\Psi(\mathbf{y}_i^\star, \mathbf{x}_i) - \Psi(\mathbf{y}_i, \mathbf{x}_i)\right]\right] \geq \frac{1}{m}\sum_{i=1}^m r_i \Delta(\mathbf{y}_i, \mathbf{y}_i') - \xi - \epsilon_{\mathrm{cp}}$.
11: **Output:** $\mathbf{w}, \xi$; $\boldsymbol{\lambda}, \mathcal{W}$.
```

1-slack formulation of our method can be written as:

$$
\begin{aligned}
\min_{\mathbf{w},\xi} \quad & \tfrac{1}{2}\|\mathbf{w}\|_2^2 + C\xi \\
\text{s.t.:} \quad & \frac{1}{m}\mathbf{w}^\top \left[\sum_{i=1}^m r_i \cdot \left[\Psi(\mathbf{y}, \mathbf{x}_i; \mathbf{H}) - \Psi(\mathbf{y}_i, \mathbf{x}_i; \mathbf{H})\right]\right] \\
& \geq \frac{1}{m}\sum_{i=1}^m r_i \Delta(\mathbf{y}_i, \mathbf{y}) - \xi, \forall \boldsymbol{r} \in \{0,1\}^m, \forall \mathbf{y} \in \mathcal{Y}, \\
& \mathbf{w} \geq 0, \xi \geq 0.
\end{aligned} \tag{3.23}
$$

Cutting-plane methods work by finding the most violated constraint for each example i

$$
\mathbf{y}_i^\star = \operatorname{argmin} \ \mathbf{w}^\top \Psi(\mathbf{y}, \mathbf{x}; \mathbf{H}) - \Delta(\mathbf{y}_i, \mathbf{y}) \tag{3.24}
$$

at every iteration and add it to the constraint working set. The sketch of our method is summarized in Algorithm 7, which calls Algorithm 11 to solve the 1-slack optimization.

3.3.4.1 Implementation Details

To deal with the unbalanced appearance of different categories in the dataset, we define $\Delta(\mathbf{y}_i, \mathbf{y})$ as weighted Hamming loss, which weighs errors for a given class inversely proportional to the frequency it appears in the training data, similar to [26]. In the inference problem in Eq. (3.24), when using the hamming loss as the label cost Δ, the label cost term can be absorbed into the unary part. We can apply Graph-cut to efficiently solve the inference in Eq. (3.24).

3.3.4.2 Discussions on the Submodularity

It is known that if graph cuts are to be applied to achieve globally optimum labelling in segmentation, the energy function must be submodular. For foreground/background segmentation in which a (super-)pixel label takes value in $\{0, 1\}$, we show that our method keeps this submodular property. It is commonly known that an energy function is submodular if its pairwise term satisfies: $\eta_{pq}(0, 0) + \eta_{pq}(1, 1) \leq \eta_{pq}(0, 1) + \eta_{pq}(1, 0)$. Recall that our pairwise energy is written as $\eta_{pq}(y_p, y_q) = \mathbf{w}^{(2)\top}\mathbf{H}^{(2)}(\mathbf{x}_p, \mathbf{x}_q)\delta(y_p \neq y_q)$. Clearly we have $(\eta_{pq}(0, 0) = \eta_{pq}(1, 1) = 0)$ because of the indicator function $\delta(y_p \neq y_q)$. The second thing is to ensure $\eta_{pq}(1, 0) + \eta_{pq}(0, 1) \geq 0$. Given the non-negativeness constraint we impose on $\mathbf{w}$ in our model, and the output of decision trees in our method taking values from $\{0, 1\}$, we have $\eta_{pq}(1, 0) \geq 0$ and $\eta_{pq}(0, 1) \geq 0$. We thus accomplish the proof of the submodularity of our model. In the case of multi-object segmentation, the inference is done by the α-expansion of graph cuts.

3.4 Experiments

To demonstrate the effectiveness of the proposed method, we first compare our model with some most related baseline methods, which are SVM, AdaBoost and SSVM. In Sec. 3.4.3, we show that state-of-the-art results can be achieved by incorporating the recent unsupervised feature learning method [59].

3.4.1 Experimental Setup

The datasets evaluated here include three binary (Weizmann horse, Oxford flower and Graz-02) and one multi-class dataset (MSRC-21). The Weizmann horse dataset[1] consists of 328 horse images from various backgrounds, with groundtruth masks available for each image. We use the same data split as in [25] and [60]. The Oxford 17 category flower dataset [61] is composed of 849 flower images. Those with too small foreground are removed, which leaves 753 for segmentation purpose [61]. The data split stated in [61] is used to perform the evaluation. During our experiment, images of the Weizmann horse and the Oxford flower datasets are resized to 256×256. The Graz-02 dataset[2]

[1]`http://www.msri.org/people/members/eranb/`
[2]`http://www.emt.tugraz.at/~pinz/`

contains 3 categories (bike, car and people). This dataset is considered challenging as the objects appear at various background and with different poses. We follow the evaluation protocol in [62] to use 150 for training and 150 for testing for each category. The MSRC-21 dataset [50] is a popular multi-class segmentation benchmark with 591 images containing objects from 21 categories. We follow the standard split to divide the dataset into training/validation/test subsets.

We start with over-segmenting the images into superpixels using SLIC [63], with ~ 700 superpixels generated per image. We then extract dense SIFT descriptors and color histograms around each superpixel centroid with different block sizes (12×12, 24×24, 36×36). The dense SIFT descriptors are then quantized into bag-of-words feature using nearest neighbor search with a codebook size of 400. We construct four types of pairwise features also using different block sizes to enforce spatial smoothness, which are color difference in LUV space, color histogram difference, texture difference in terms of LBP operators as well as shared boundary length [51]. The column generation iteration number of our CRFTree is set to 50 based on a validation set. We learn tree potentials with tree depth 2. Training on the MSRC dataset takes around 16 hours on a standard PC desktop.

3.4.2 Comparing with Baseline Methods

We first compare CRFTree with some conventional methods, which are linear SVM, AdaBoost and SSVM to demonstrate the superiority of our method. For SVM and AdaBoost, each superpixel is classified independently without CRF. We mainly evaluate on the more challenging Graz-02 and MSRC dataset in this part. The regularization parameter C of SVM, SSVM and our CRFTree are selected from $\{1, 10, 100, 1000\}$ based on a validation set. We use depth-2 decision trees for training AdaBoost and our CRFTree. The maximum iteration number of AdaBoost is chosen from {50, 100, 200}. For our method, we treat the foreground and background as two categories in the binary case to learn class-wise potentials.

3.4.2.1 Graz-02

For a comprehensive evaluation, we use two measurements to quantify the performance on the Graz-02 dataset, which are intersection over union score and the pixel accuracy (including foreground and background). We report the results in Table 3.1. As can be observed, AdaBoost based on a depth-2 decision tree performs better than the linear SVM. On the other hand, structured methods which jointly consider local information and spatial consistency are able to significantly outperform the simple binary models.

Category	bike	car	people
	intersection/union (foreground, background)(%)		
SVM	67.8 (51.9, 83.8)	69.7 (46.8, 92.6)	65.0 (44.5, 85.5)
AdaBoost	71.2 (57.6, 84.9)	71.0 (49.4, 92.6)	67.7 (48.7, 86.7)
SSVM	72.2 (58.6, 85.8)	76.9 (60.0, 94.2)	70.9 (53.8, 87.9)
CRFTree	**76.4** (65.0, 87.8)	**79.5** (64.0, 95.0)	**74.2** (58.7, 89.7)
	pixel accuracy (foreground, background)(%)		
SVM	79.5 (67.4, 91.5)	77.3 (57.2, 97.3)	77.7 (63.8, 91.6)
AdaBoost	83.8 (77.3, 90.3)	80.1 (63.5, 96.6)	80.5 (69.0, 91.9)
SSVM	83.8 (76.1, 91.6)	85.5 (73.8, 97.2)	83.9 (75.8, 92.1)
CRFTree	**87.8** (83.9, 91.8)	**87.0** (76.4, 97.7)	**85.9** (78.4, 93.4)

Table 3.1: The average intersection-over-union score and average pixel accuracy comparison on the Graz-02 dataset. We include the foreground and background results in the brackets. Our method CRFTree with nonlinear and class-wise potentials learning performs better than all the baseline methods.

Method	Sa	So
Levin & Weiss [64]	**95.5**	-
Cosegmentation [65]	80.1	-
Bertelli *et al.* [25]	94.6	80.1
Kuttel *et al.* [60]	94.7	-
CRFTree (FL)	94.6	**80.4**

Table 3.2: Performance of different methods on the Weizmann Horse dataset.

By introducing nonlinear as well as class-wise potentials learning, our method is able to gain further improvement over SSVM. We show some qualitative evaluation examples of AdaBoost, SVM, SSVM and our CRFTree on Graz-02 dataset in Fig. 3.4.

3.4.2.2 MSRC-21

We learn class-wise potentials using our CRFTree for each of the 21 classes on the MSRC dataset. The compared results are summarized in Table 3.6 (upper part). Similar conclusions can be drawn as on the Graz-02 dataset and our CRFTree again outperforms all its baseline competitors. In Fig. 3.6, we show the confusion matrices of the predictions from SSVM and our CRFTree. The diagonal entries show the true positive predictions. As we can see, our CRFTree model performs generally better than SSVM.

3.4.3 Comparing with State-of-the-art Methods

To compare with state-of-the-art methods, we add features learned from the unsupervised feature learning [59] into our method. Specifically, we first learn a dictionary **B** of size 400 and patch size 6×6 based on the evaluated image dataset using Kmeans, and then use the soft threshold coding [59] to encode patches extracted from each superpixel block. The final feature vectors (we call it encoding feature here) are obtained by performing a three-level max pooling over the superpixel block.

Method	Sa	So
Nilsback *et al.* [61]	-	94.0
Bertelli *et al.* [25]	97.7	92.3
CRFTree (FL)	**98.0**	**94.2**

Table 3.3: Performance of different methods on the Oxford Flower dataset. Our method CRFTree performs better than the compared methods.

Category	bike	car	people
	intersection/union (foreground, background)(%)		
CRFTree (FL)	78.3 (67.7, 88.9)	83.0 (70.1, 95.9)	75.7 (61.0, 90.5)
	pixel accuracy (foreground, background)(%)		
CRFTree (FL)	89.1 (85.8, 92.4)	90.0 (82.1, 98.0)	86.9(80.0, 94.0)

Table 3.4: The average intersection-over-union score and average pixel accuracy of CRFTree by incorporating unsupervised feature learning method. We include the foreground and background results in the brackets.

3.4.3.1 Weizmann Horse

We quantify the performance by the global pixel-wise accuracy S_a and the foreground intersection over union score S_o, as did in [25]. S_a measures the percentage of pixels correctly classified while S_o directly reflects the segmentation quality of the foreground. The results are reported in Table 3.2. Our method performs better than the kernel structural learning method of [25], which may result from the fact that they only introduced nonlinearity into the unary part while our method achieves nonlinearity on both unary and pairwise terms. The best S_a score is obtained by [64]. However their method relies on an assumption that a perfect bounding box of the horse is available for each test image, which is not practically applicable. On the contrary, we provide a principal and general way of nonlinearly learning CRF parameters. We show some segmentation examples of our method in Fig. 3.2. As can be observed, our method produces predictions quite close to the ground-truths.

3.4.3.2 Oxford Flower

As in [25], we also use S_a and S_o to measure the performance on the Oxford flower dataset, and report the results in Table 3.3. Our method performs comparable to the original work of [61] on this dataset in terms of S_o while again outperforms the closely related state-of-the-art work of [25]. It is also worth noting that the method in [61] is very domain specific, which relies on modelling the flower's shape (center and petal), while ours is generally applicable. Examples of segmentation results on Oxford flower dataset are shown in Fig. 3.5.

Method	bike	car	people	average
Marszalek & Schimid [62]	61.8	53.8	44.1	53.2
Fulkerson *et al.* [66]	66.4	54.7	51.4	57.5
Aldavert *et al.* [67]	71.9	62.9	58.6	64.5
Kuettel *et al.* [60]	63.2	74.8	66.4	68.1
CRFTree (FL)	**80.7**	**82.4**	**75.8**	**79.5**

Table 3.5: Comparing with state-of-the-art methods on the Graz-02 dataset. We report the F-score (%) for each class and the average over classes. Our method CRFTree outperforms all the compared methods with a large margin.

3.4.3.3 Graz-02

As in the work of [62], [66], [67], [60], we also evaluate the F-score on the Graz-02 dataset besides the above mentioned intersection over union score and pixel accuracy. The F-score is defined as $F = 2pr/(p + r)$, where p is the precision and r is the recall. We summarize the results in Table 3.5 and Table 3.4. From Table 3.5, it can be seen that our method significantly outperforms all the compared methods, which fully demonstrate the power of nonlinear and class-wise potentials learning. Furthermore, compared with the results in Table 3.1, adding more features help to improve the performance.

3.4.3.4 MSRC-21

For efficient training on the MSRC dataset, we first train a linear SVM on each of the three groups of feature (bag-of-words, color histogram and encoding features) and use the per-class scores as the feature vectors to train our model by using depth-2 decision trees. The compared results with state-of-the-art work are reported in the lower part of Table 3.6. By incorporating features learned from the unsupervised feature learning method, our CRFTree gains significant improvement over the previous results which only use bag-of-words and color histogram features. Moreover, our method performs better than the related work of Lucchi *et al.* [26] which uses kernel transformed features. It has to be pointed out that we did not employ any global potentials (while in [26], they improve the global and average per-category accuracy to 82 and 76 when adding global information). If global or higher potentials are incorporated into our model, further performance promotion can be expected. We show some qualitative evaluation examples in Fig. 3.3.

3.4.4 Object-aware vs. Non-object-aware

To further demonstrate the power of the proposed method by learning object-aware potentials, we add an experiment to compare the object-aware and the non-object-aware

	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Average**	**Global**
SVM	54	92	73	41	54	80	51	67	51	41	59	41	28	8	64	17	75	41	23	20	7	47.0	63.7
AdaBoost	68	92	83	48	58	87	43	69	58	43	64	41	32	14	70	28	79	47	22	41	6	52.0	68.6
SSVM	65	92	81	42	76	84	65	70	75	54	87	62	31	14	76	31	78	61	30	25	2	57.2	70.8
CRFTree	53	87	85	59	84	90	77	82	81	54	90	57	62	22	81	59	80	71	26	49	15	**64.9**	**73.9**
CRFTree (FL)	64	94	89	80	89	91	93	81	81	74	89	78	74	50	86	79	88	85	48	69	20	**76.3**	**82.2**
Shotton *et al.* [50]	49	88	79	97	97	78	82	54	87	74	72	74	36	24	93	51	78	75	35	66	18	67	72
Ladicky *et al.* [68]	80	96	86	74	87	99	74	87	86	87	82	97	95	30	86	31	95	51	69	66	9	75	86
Gonfaus *et al.* [69]	60	78	77	91	68	88	87	76	73	77	93	97	73	57	95	81	76	81	46	56	46	75	77
Lucchi *et al.* [26]	41	77	79	87	91	86	92	65	86	65	89	61	76	48	77	91	77	82	32	48	39	70	73
Lucchi *et al.* [70]	67	89	85	93	79	93	84	75	79	87	89	92	71	46	96	79	86	76	64	77	50	**78.9**	**83.7**

Table 3.6: Segmentation results on the MSRC dataset. We report the pixel-wise accuracy for each category as well as the average per-category scores and the global pixel-wise accuracy. (1) The upper part presents the comparison with baseline methods, which all use bag-of-words and color histogram features. Our method CRFTree gains impressive improvements over SSVM while far better than simple linear models. (2) The lower part shows the results of our method incorporated with unsupervised feature learning (denoted as CRFTree (FL)) compared to state-of-the-art methods on this dataset.

	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Average**	**Global**
CRFTree (NOA)	59	86	84	56	81	83	74	71	74	49	82	51	62	13	82	66	77	75	25	41	11	62.1	71.3
CRFTree (OA)	53	87	85	59	84	90	77	82	81	54	90	57	62	22	81	59	80	71	26	49	15	64.9	73.9

Table 3.7: Compared results of the object-aware (denoted as CRFTree (OA)) and the non-object-aware (denoted as CRFTree (NOA)) models on the MSRC dataset. Using object-aware potentials learning yields better results, which demonstrates the strength of the proposed method.

models (details in Sec. 3.3.3). We use simple bag-of-words and color histogram features as in Sec. 3.4.2 and report the results in Table 3.7. As expected, our object-aware potentials learning outperforms its non-object-aware counterpart, which validates our claim and shows promising prospects in the application of multi-object segmentation.

3.5 Conclusion

In this work, we have proposed a structured learning of tree potentials method for image segmentation. The unary and pairwise potentials are ensembles of class-wise trees, with the ensemble parameters and the trees jointly learned in a unified max-margin framework. In this way, nonlinearity is easily introduced into CRF learning. We have exemplified the superiority of the proposed nonlinear potentials learning method by comparing with state-of-the-art methods on both binary and multi-class object segmentation datasets.

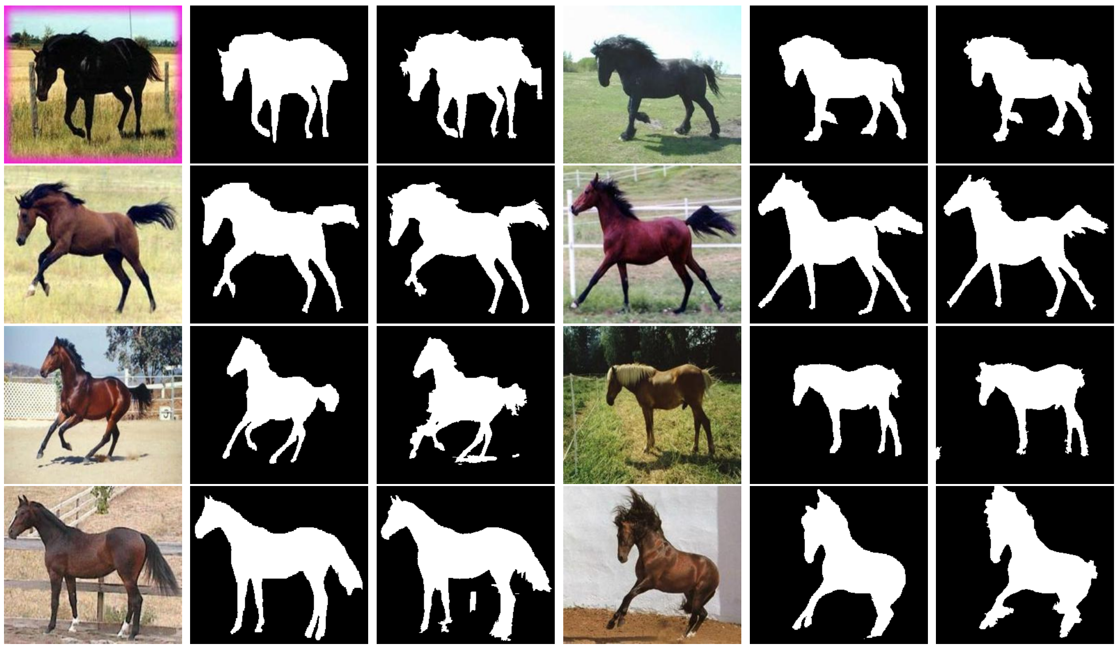


Figure 3.2: Segmentation examples on the Weizmann horse dataset. 1st and 4th columns: Test images; 2nd and 5th columns: Ground truth; 3rd and 6th columns: Predictions produced by our CRFTree method.

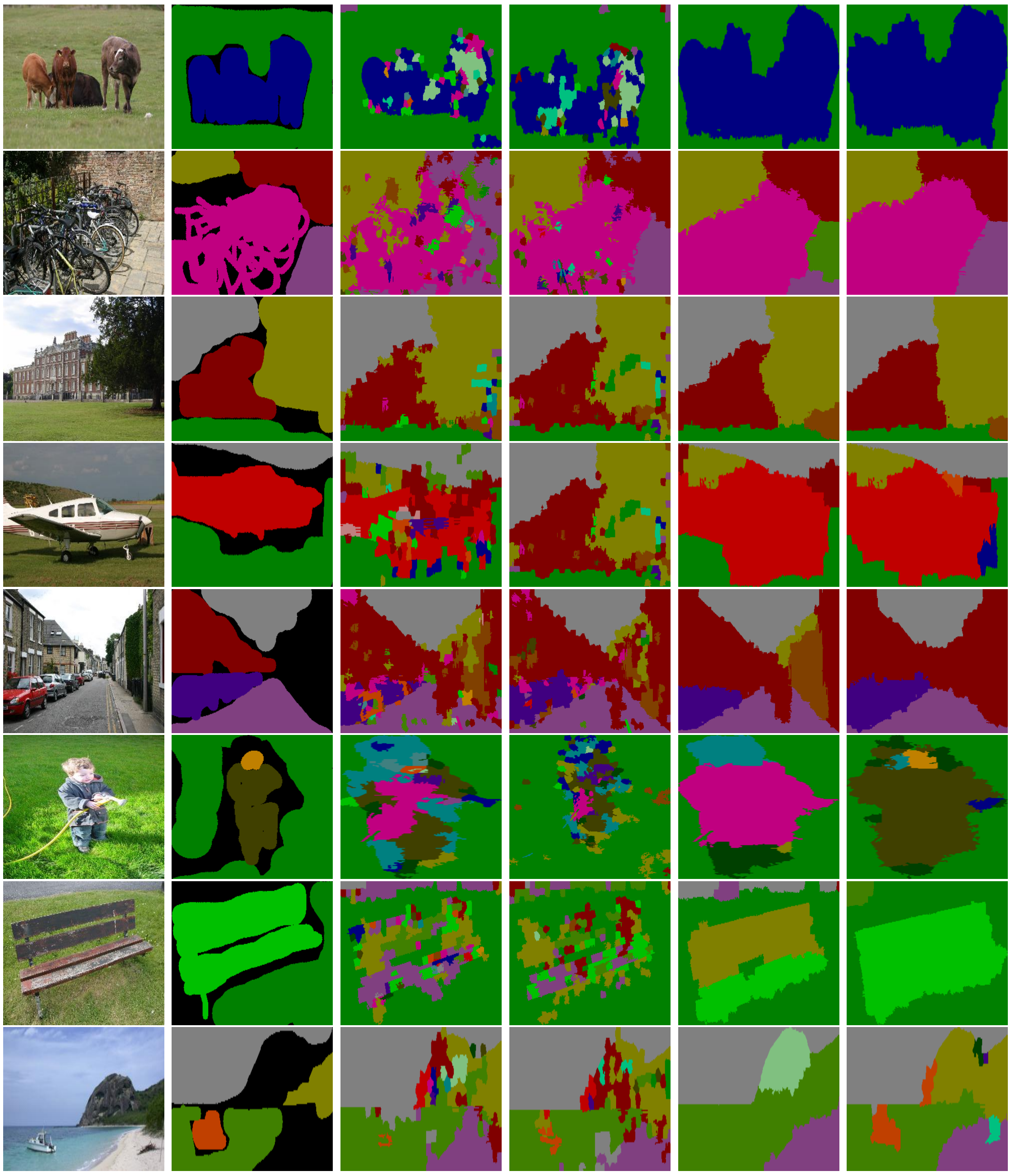


Figure 3.3: Segmentation examples on MSRC. 1st column: Test images; 2nd column: Ground truth; 3rd column: Predictions of AdaBoost; 4th column: Predictions of SVM; 5th column: Predictions of SSVM; 6th column: Predictions of CRFTree with unsupervised feature learning.

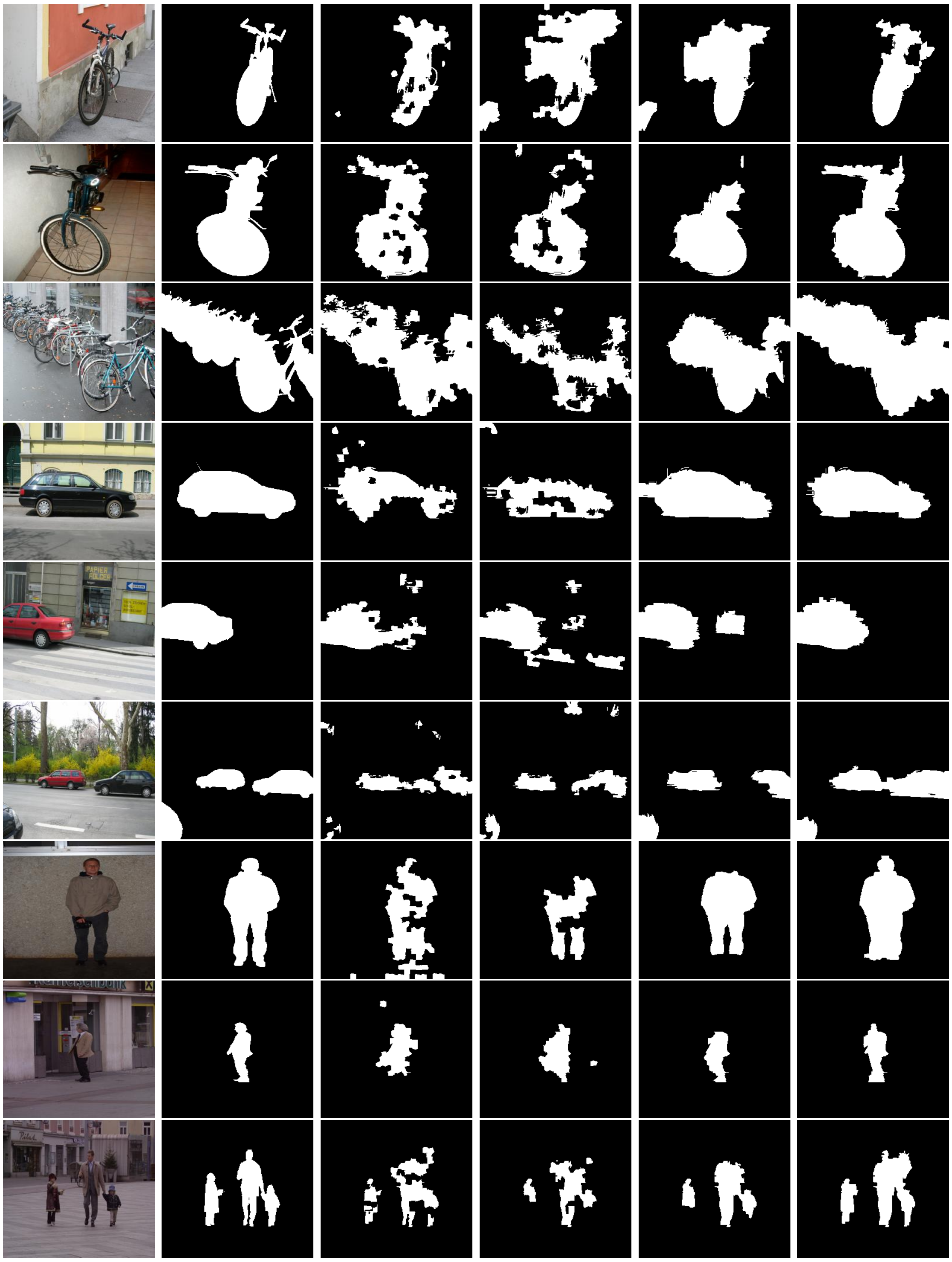


Figure 3.4: Qualitative comparison on the Graz-02 dataset. 1st column: Test images; 2nd column: Ground truth; 3rd column: Predictions of AdaBoost; 4th column: Predictions of SVM; 5th column: Predictions of SSVM; 6th column: Predictions of CRFTree. SSVM and CRFTree present more smooth boundary than AdaBoost and SVM due to the introduce of pairwise terms. Compared to SSVM, our CRFTree yields more accurate segmentation because of the non-linearity property.

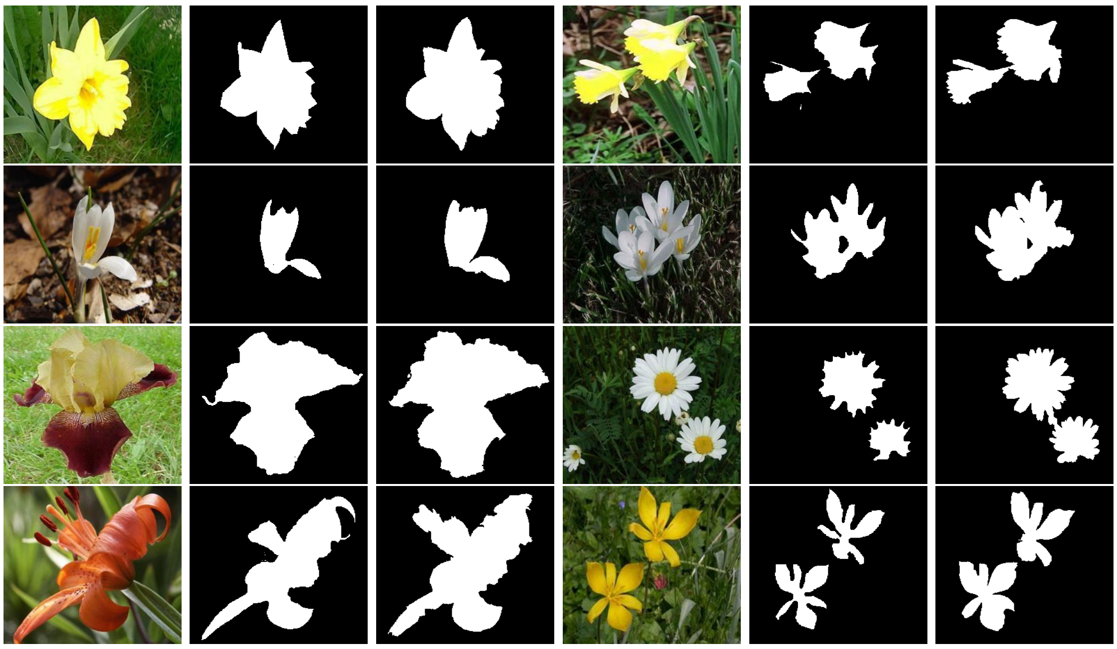
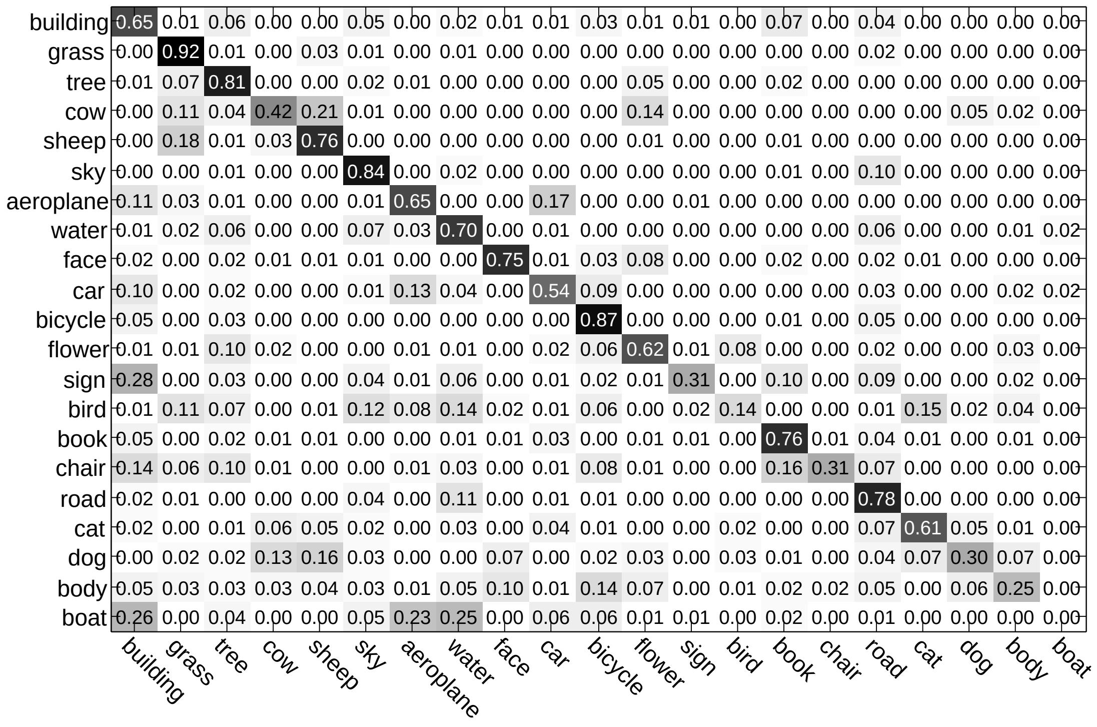
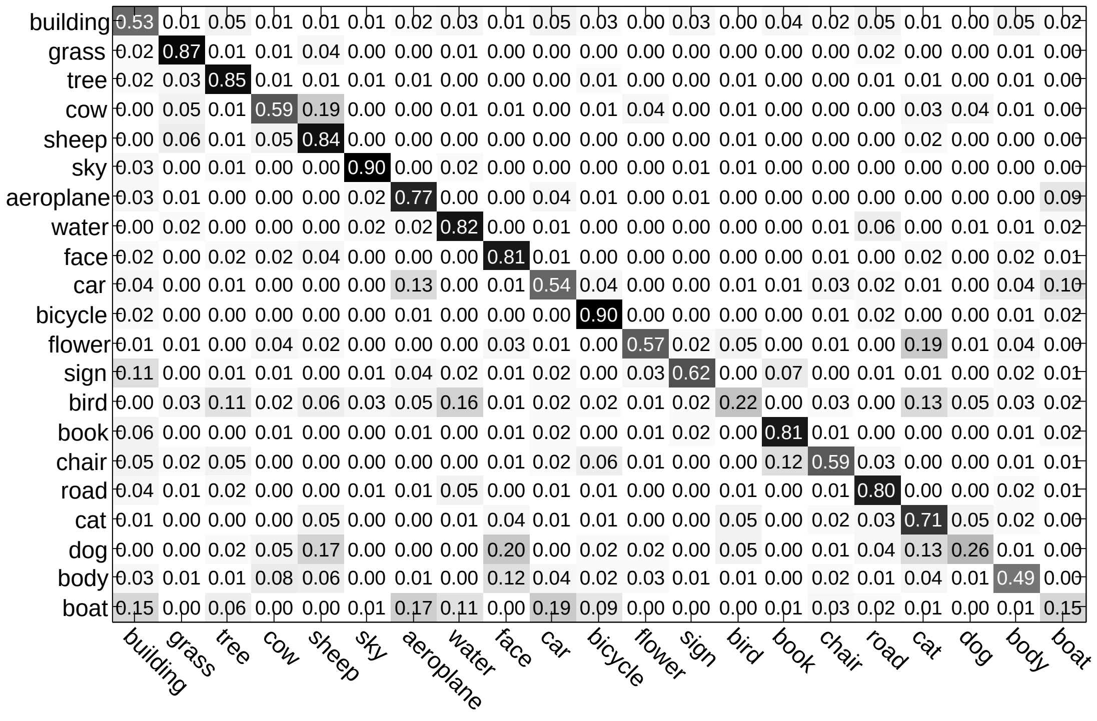


Figure 3.5: Examples of qualitative evaluations on the Oxford flower dataset. 1st and 4th columns: Test images; 2nd and 5th columns: Ground truth; 3rd and 6th columns: Predictions produced by our method CRFTree. Our predictions well preserve the boundaries.



	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat
building	0.65	0.01	0.06	0.00	0.00	0.05	0.00	0.02	0.01	0.01	0.03	0.01	0.01	0.00	0.07	0.00	0.04	0.00	0.00	0.00	0.00
grass	0.00	0.92	0.01	0.00	0.03	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
tree	0.01	0.07	0.81	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00
cow	0.00	0.11	0.04	0.42	0.21	0.01	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.02	0.00
sheep	0.00	0.18	0.01	0.03	0.76	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
sky	0.00	0.00	0.01	0.00	0.00	0.84	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.10	0.00	0.00	0.00	0.00
aeroplane	0.11	0.03	0.01	0.00	0.00	0.01	0.65	0.00	0.00	0.17	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
water	0.01	0.02	0.06	0.00	0.00	0.07	0.03	0.70	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.01	0.02
face	0.02	0.00	0.02	0.01	0.01	0.01	0.00	0.00	0.75	0.01	0.03	0.08	0.00	0.00	0.02	0.00	0.02	0.01	0.00	0.00	0.00
car	0.10	0.00	0.02	0.00	0.00	0.01	0.13	0.04	0.00	0.54	0.09	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.02	0.02
bicycle	0.05	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.87	0.00	0.00	0.00	0.01	0.00	0.05	0.00	0.00	0.00	0.00
flower	0.01	0.01	0.10	0.02	0.00	0.00	0.01	0.01	0.00	0.02	0.06	0.62	0.01	0.08	0.00	0.00	0.02	0.00	0.00	0.03	0.00
sign	0.28	0.00	0.03	0.00	0.00	0.04	0.01	0.06	0.00	0.01	0.02	0.01	0.31	0.00	0.10	0.00	0.09	0.00	0.00	0.02	0.00
bird	0.01	0.11	0.07	0.00	0.01	0.12	0.08	0.14	0.02	0.01	0.06	0.00	0.02	0.14	0.00	0.00	0.01	0.15	0.02	0.04	0.00
book	0.05	0.00	0.02	0.01	0.01	0.00	0.00	0.01	0.01	0.03	0.00	0.01	0.01	0.00	0.76	0.01	0.04	0.01	0.00	0.01	0.00
chair	0.14	0.06	0.10	0.01	0.00	0.00	0.01	0.03	0.00	0.01	0.08	0.01	0.00	0.00	0.16	0.31	0.07	0.00	0.00	0.00	0.00
road	0.02	0.01	0.00	0.00	0.00	0.04	0.00	0.11	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.78	0.00	0.00	0.00	0.00
cat	0.02	0.00	0.01	0.06	0.05	0.02	0.00	0.03	0.00	0.04	0.01	0.00	0.00	0.02	0.00	0.00	0.07	0.61	0.05	0.01	0.00
dog	0.00	0.02	0.02	0.13	0.16	0.03	0.00	0.00	0.07	0.00	0.02	0.03	0.00	0.03	0.01	0.00	0.04	0.07	0.30	0.07	0.00
body	0.05	0.03	0.03	0.03	0.04	0.03	0.01	0.05	0.10	0.01	0.14	0.07	0.00	0.01	0.02	0.02	0.05	0.00	0.06	0.25	0.00
boat	0.26	0.00	0.04	0.00	0.00	0.05	0.23	0.25	0.00	0.06	0.06	0.01	0.01	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00

(a)



	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat
building	0.53	0.01	0.05	0.01	0.01	0.01	0.02	0.03	0.01	0.05	0.03	0.00	0.03	0.00	0.04	0.02	0.05	0.01	0.00	0.05	0.02
grass	0.02	0.87	0.01	0.01	0.04	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.01	0.00
tree	0.02	0.03	0.85	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.01	0.01	0.00	0.01	0.00
cow	0.00	0.05	0.01	0.59	0.19	0.00	0.00	0.01	0.01	0.00	0.01	0.04	0.00	0.01	0.00	0.00	0.00	0.03	0.04	0.01	0.00
sheep	0.00	0.06	0.01	0.05	0.84	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.02	0.00	0.00	0.00
sky	0.03	0.00	0.01	0.00	0.00	0.90	0.00	0.02	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
aeroplane	0.03	0.01	0.00	0.00	0.00	0.02	0.77	0.00	0.00	0.04	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09
water	0.00	0.02	0.00	0.00	0.00	0.02	0.02	0.82	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.06	0.00	0.01	0.01	0.02
face	0.02	0.00	0.02	0.02	0.04	0.00	0.00	0.00	0.81	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.02	0.00	0.02	0.01
car	0.04	0.00	0.01	0.00	0.00	0.00	0.13	0.00	0.01	0.54	0.04	0.00	0.00	0.01	0.01	0.03	0.02	0.01	0.00	0.04	0.10
bicycle	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.01	0.02	0.00	0.00	0.01	0.02
flower	0.01	0.01	0.00	0.04	0.02	0.00	0.00	0.00	0.03	0.01	0.00	0.57	0.02	0.05	0.00	0.01	0.00	0.19	0.01	0.04	0.00
sign	0.11	0.00	0.01	0.01	0.00	0.01	0.04	0.02	0.01	0.02	0.00	0.03	0.62	0.00	0.07	0.00	0.01	0.01	0.00	0.02	0.01
bird	0.00	0.03	0.11	0.02	0.06	0.03	0.05	0.16	0.01	0.02	0.02	0.01	0.02	0.22	0.00	0.03	0.00	0.13	0.05	0.03	0.02
book	0.06	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.01	0.02	0.00	0.01	0.02	0.00	0.81	0.01	0.00	0.00	0.00	0.01	0.02
chair	0.05	0.02	0.05	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.06	0.01	0.00	0.00	0.12	0.59	0.03	0.00	0.00	0.01	0.01
road	0.04	0.01	0.02	0.00	0.00	0.01	0.01	0.05	0.00	0.01	0.01	0.00	0.00	0.01	0.00	0.01	0.80	0.00	0.00	0.02	0.01
cat	0.01	0.00	0.00	0.00	0.05	0.00	0.00	0.01	0.04	0.01	0.01	0.00	0.00	0.05	0.00	0.02	0.03	0.71	0.05	0.02	0.00
dog	0.00	0.00	0.02	0.05	0.17	0.00	0.00	0.00	0.20	0.00	0.02	0.02	0.00	0.05	0.00	0.01	0.04	0.13	0.26	0.01	0.00
body	0.03	0.01	0.01	0.08	0.06	0.00	0.01	0.00	0.12	0.04	0.02	0.03	0.01	0.01	0.00	0.02	0.01	0.04	0.01	0.49	0.00
boat	0.15	0.00	0.06	0.00	0.00	0.01	0.17	0.11	0.00	0.19	0.09	0.00	0.00	0.00	0.01	0.03	0.02	0.01	0.00	0.01	0.15

(b)

Figure 3.6: Confusion matrices of the predictions of different models using bag-of-words feature and color histogram features on the MSRC dataset. (a) SSVM; (b) CRFTree.

Chapter 4

CRF Learning with CNN Potentials for Image Segmentation

In this chapter, we propose to exploit a pre-trained large convolutional neural network (CNN) to construct unary potentials for CRF learning. The deep CNN is trained on the ImageNet dataset and transferred to image segmentations here for constructing potentials of superpixels. Then the CRF parameters are learnt using a structured support vector machine (SSVM). To fully exploit context information in inference, we construct spatially related co-occurrence pairwise potentials and incorporate them into the energy function. This prefers labelling of object pairs that frequently co-occur in a certain spatial layout and at the same time avoids implausible labellings during the inference. Extensive experiments on binary and multi-class segmentation benchmarks demonstrate the promise of the proposed method. We thus provide new baselines for the segmentation performance on the Weizmann horse, Graz-02, MSRC-21, Stanford Background and PASCAL VOC 2011 datasets.

4.1 Introduction

The task of image segmentation is to produce a pixel level labelling of different object categories, with wide variety of applications ranging from image retrieval to object recognition. It is challenging as the objects may appear in various backgrounds and different visual conditions. CRFs [49] model the conditional distribution of labels given observations, representing the state-of-the-art in image/object segmentation [24, 26, 50–52]. In [24], Szummer *et al.* proposed to learn the coefficients of CRF potentials using structured support vector machines (SSVM) and graph cuts. Since then, SSVM has been widely applied for CRF learning in segmentation tasks.

In the pipeline of CRF learning based image segmentation, finding a good feature representation is of great significance, and can have a profound impact on the segmentation accuracy. Most previous studies rely on hand-crafted features, *e.g.*, using color histograms, HOG or SIFT descriptors to construct bag-of-words features [26, 51, 66, 70, 71]. Recently, feature learning and especially deep learning methods have gained great popularity in machine learning and related fields. This type of methods typically take raw images as input and learn a (deep) representation of the images, and have found phenomenal success in various tasks such as speech recognition [72], image classification [4, 73], object detection [40] *etc.*See Bengio *et al.* [74] for a detailed review. Deep learning methods attempt to model high-level abstractions in data at multiple layers, inspired from the cognitive processes of human brains, which generally starts from simpler concepts to more abstract ones. The learning is achieved by using deep architectures, *e.g.*, deep belief networks (DBNs) [72], stacked autoassociator networks [75], deep convolutional neural networks (CNNs) [2, 4, 76], *etc.*Among them, CNNs are high-capacity machine learning models with a very large number of (typically a few million) parameters that are optimized from labelled training examples. The success of CNNs in various vision tasks [2, 4] is mainly due to their ability to learn rich mid-level features that accommodate within-class variance and at the same time possess discriminative information. This is in contrast to low-level hand-crafted features.

On the other hand, prior work [77–79] has demonstrated that holistic reasoning about the occurrences of all classes helps to improve segmentation performance. These are based on the considerations that neighbouring image regions may be occupied by frequently co-occurring objects, and object pairs of mutual exclusion are less likely to appear together. For example, a cow is more likely to show up together with grass rather than a monitor, and grass is less likely to appear above sky. Therefore, we here propose to construct spatially related co-occurrence pairwise potentials to exploit the context information during inference.

In summary, we highlight the main contributions of this work as follows.

- We show that cross-domain image features learned by CNNs with labelled data from ImageNet[1] can be successfully transferred for segmentation purpose. By thoroughly evaluating the performance of the CNN features of different depths and comparing with the traditional bag-of-words and unsupervised feature learning methods, we demonstrate the power of CNN features in image segmentation.

[1]`http://image-net.org`

- We illustrate that SSVM based CRF learning with CNN features yields astounding results and thus provide new baselines for segmentation performance on the Weizmann horse, Graz02, MSRC-21, Stanford Background and PASCAL VOC 2011 datasets.

- We incorporate spatially related co-occurrence pairwise potentials into the inference and gain further performance boost.

4.1.1 Related Work

We briefly review some work that is relevant to ours. The first work on using convolutional networks for scene parsing is [80]. In [80], they train a deep CNN using a supervised greedy learning strategy taking pixels as input to yield a pixel-wise labelling of an image. While somewhat preliminary, they achieved marginal improvement over CRF learning based segmentation methods. We show in this paper that deep CNN features transferred from ImageNet (ImageNet is an image dataset organized according to the WordNet hierarchy, containing millions of labelled images.) combined with SSVM based CRF learning outperforms most state-of-the-art methods. Schulz *et al.* [81] propose to predict the segmentation mask by adding a pairwise class location filter to the conventional CNN architecture of [2]. In the work of [41], the authors use a multiscale convolutional network trained from raw pixels to extract dense feature vectors that encode regions of multiple sizes centered on each pixel and present impressive results on several datasets. Our work differs from [41] in two aspects. First, we transfer a deep CNN trained on the ImageNet [4] dataset to segmentation while [41] trains a 3-stage convolutional network [2] on the current training data of the segmentation dataset, and we demonstrate experimentally that better performance can be achieved by our method. Secondly, our method uses SSVM to learn CRF potentials while no learning is involved in [41]. Figure 4.1 shows a sketch of our segmentation pipeline.

Most recently, Girshick *et al.* [40] demonstrate that a deep CNN trained on ImageNet can be successfully transferred to object detection and great performance boost is achieved on the PASCAL VOC 2012 dataset. As an extension of their statement, they also conduct a scene labelling experiment on the PASCAL VOC segmentation dataset to validate the power of deep CNN features on the segmentation task. Our work is mainly inspired from theirs, but differs in that we combine deep CNN features with SSVM based CRF learning in contrast to their region proposals and support vector regression based method. Furthermore, we thoroughly evaluate the performance of deep CNN features compared to the bag-of-words features and unsupervised learned features, and provides new baselines for labelling performance on various segmentation benchmarks.

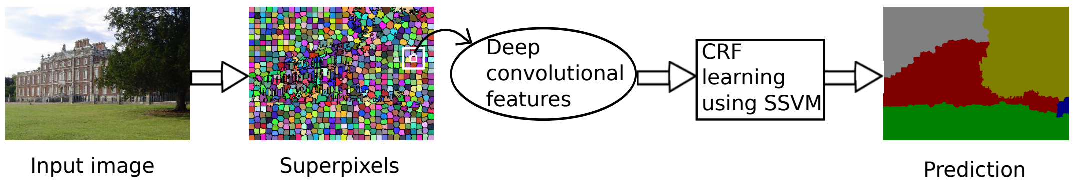


Figure 4.1: An illustration of the proposed segmentation pipeline. We first over-segment the image into superpixels and then compute deep convolutional features of the patch around each superpixel centroid using a pre-trained deep CNN. The learned features are then used to learn a CRF for segmentation.

Co-occurrence statistics have been exploited and demonstrated its strength in the community. In [77], the authors incorporate semantic object context as a post-processing step by considering the co-occurrence counts of label pairs. Ladicky *et al.* [78] explores the inference methods for CRF with co-occurrence statistics by considering a class of global potentials. Different from their methods that ignore spatial relations of the co-occurrences, we propose to construct spatially related co-occurrence pairwise potentials, which favor labellings of object pairs that frequently co-occur in a certain spatial layout while at the same time prevents unreasonable labellings. Our method is inspired from [79] but differs in that they incorporate the mutex information by adding a mutex constraint to the inference problem while we simply construct co-occurrence pairwise potentials, and most importantly, we explore CNN features combined with SSVM based CRF learning.

4.2 Proposed Method

4.2.1 Deep Convolutional Neural Networks

Deep neural networks consist of multiple hidden layers and are typically trained in a supervised fashion. As stated in Section 4.1, there are several architectures currently employed for deep learning. Among them, deep convolutional neural networks (CNN) [2] have shown superior performance on various computer vision problems. A deep CNN is typically composed of multiple convolutional layers, pooling layers and fully-connected layers. Convolutional networks are variants of multi-layer perceptrons (MLP) which are inspired from biological processes. Given an image I, convolutional networks learn multi-layer feature maps. Neurons of each layer are sensitive to a small sub-regions of the input image, which are called receptive fields (RF). A sigmoid function ($f(x) = \tanh(x)$) or rectified linear units (abbreviated as ReLUs: $f(x) = max(0, x)$) are usually applied to each feature map to introduce nonlinearity. In general, a convolutional layer is followed by a subsampling or pooling layer, with each map being max pooled over $p \times p$ continuous regions.

We here introduce in detail a notable deep CNN architecture for image classification on the ImageNet dataset, introduced by Krizhevsky *et al.* [4], which consists of 5 convolutional layers and 2 fully connected layers together with a soft-max layer. Each convolutional layer takes as input the output of its previous layer, and do convolution and optional max pooling. The first convolutional layer takes as input the image of size $m \times m \times r$, where m is the height and width of the image and r is the number of channels, *e.g.*, $r = 3$ for an RGB image. The image is then filtered by k kernels of size $n \times n \times q$ where $n < m$ and $q \leq r$ at a stride of s. Each of the $n \times n$ locally connected regions in the image (RF), are convolved with the kernels to produce k feature maps of size $(\lceil \frac{m-n}{s} \rceil + 1) \times (\lceil \frac{m-n}{s} \rceil + 1)$. ReLUs are applied to each feature map to introduce nonlinearity. For the subsequent convolutional layers, each layer takes the output of its previous layer as input, and do similar operations. An overlapping max-pooling which operates over 3×3 continuous regions at a step size 2 follows the 1st, 2nd and 5th convolutional layer. The final convolutional layers are then followed by two fully connected layers, each with 4096 neurons. The output of the last fully connected layer is fed to a 1000-class soft-max layer which is used as the predictor. During training, the parameters of each layer are initialized and then learned by back propagation using stochastic gradient descent. Figure 4.2 shows an illustration of the network's architecture.

We use the CNN implementation, named Caffe [82], which implements the network of [4]. The network is trained using the LSVRC-2010 dataset, consisting of 1.2 million labelled data with 1000 different classes. It is a subset of the ImageNet dataset. As demonstrated by Girshick *et al.* [40], the pre-trained CNN on ImageNet generalizes well to object detection and semantic segmentation. We then here further explore its power when combined with SSVM based CRF learning.

4.2.2 Segmentation with CRF Models

Before presenting our method, we first revisit how to use the CRF models to perform image segmentation. Given an image instance $\mathbf{x}$ and its corresponding labelling $\mathbf{y}$, CRF [49] models the conditional distribution of the form

$$\Pr(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{1}{\mathrm{Z}(\mathbf{x})} \exp(-E(\mathbf{y},\mathbf{x};\mathbf{w})). \tag{4.1}$$

where $\mathbf{w}$ are parameters and Z the partition function. The energy E of an image $\mathbf{x}$ with segmentation labels $\mathbf{y}$ over the nodes (superpixels) $\mathcal{N}$ and edges $\mathcal{S}$, takes the following

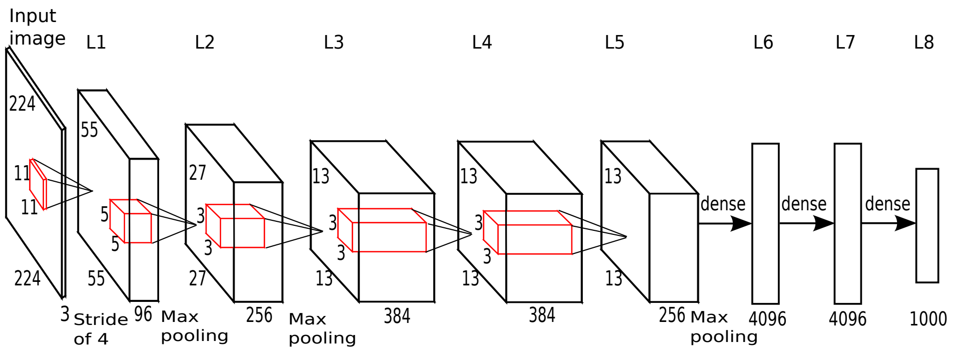


Figure 4.2: An illustration of the deep CNN architecture used for ImageNet classification by Krizhevsky *et al.* [4]. The first convolutional layer filters the input image with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels; the second convolutional layer takes the output of the first layer as input and filters it with 256 kernels of size $5 \times 5 \times 96$; each of the 3rd and 4th layer has 384 kernels of size $3 \times 3 \times 256$ and $3 \times 3 \times 384$ respectively; the 5th convolutional layer has 256 kernels of size $3 \times 3 \times 384$; the fully connected layers have 4096 kernels each and the last soft-max layer has 1000 neurons. A max-pooling layer follows the first, second and fifth layer.

form:

$$E(\mathbf{y}, \mathbf{x}; \mathbf{w}) = \sum_{p \in \mathcal{N}} U(y_p, \mathbf{x}; \mathbf{w}) + \sum_{(p,q) \in \mathcal{S}} V(y_p, y_q, \mathbf{x}; \mathbf{w}). \tag{4.2}$$

Here $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}$; U and V are the unary and pairwise potentials, both of which depend on the observations as well as the parameter $\mathbf{w}$. CRF seeks an optimal labelling that achieves maximum a posterior (MAP), which mainly involves a two-step process [24]: 1) Learning the model parameters from the training data; 2) Inferring a most likely label for the test data given the learned parameters. The segmentation problem thus reduced to minimizing the energy (or cost) over $\mathbf{y}$ by the learned parameters $\mathbf{w}$, which is:

$$\mathbf{y}^* = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}, \mathbf{x}; \mathbf{w}). \tag{4.3}$$

4.2.3 Learning CRF in the Max-Margin Framework

Applying the max-margin based CRF learning is to solve the following optimization:

$$\begin{aligned} &\min_{\mathbf{w}, \boldsymbol{\xi}} \ \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{C}{m} \sum_i \xi_i \\ &\text{s.t.: } \ E(\mathbf{y}, \mathbf{x}_i; \mathbf{w}) - E(\mathbf{y}_i, \mathbf{x}_i; \mathbf{w}) \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \\ &\qquad\qquad \forall i = 1, \dots, m, \ \text{and } \forall \mathbf{y} \in \mathcal{Y}, \\ &\qquad \boldsymbol{\xi} \geq 0, \end{aligned} \tag{4.4}$$

where $\Delta : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is a loss function associated with the prediction and the true label mask. In general, we have $\Delta(\mathbf{y}, \mathbf{y}) = 0$ and $\Delta(\mathbf{y}, \mathbf{y}') > 0$ for any $\mathbf{y}' \neq \mathbf{y}$. Intuitively, the optimization in Eq. (4.4) is to encourage the energy of the ground truth label $E(\mathbf{y}_i, \mathbf{x}_i; \mathbf{w})$ to be lower than any other *incorrect* labels $E(\mathbf{y}, \mathbf{x}_i; \mathbf{w})$ by at least a margin $\Delta(\mathbf{y}_i, \mathbf{y})$. The SSVM solves (4.4) by iteratively finding the most violated constraint for each example i:

$$\mathbf{y}_i^* = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} E(\mathbf{y}, \mathbf{x}; \mathbf{w}) - \Delta(\mathbf{y}_i, \mathbf{y}). \tag{4.5}$$

To learn CRF in the max-margin framework, we consider energy functions that are linear in the parameter $\mathbf{w}$, which indicates that the unary and the pairwise potentials in Eq. (4.2) can be written as:

$$U(y_p, \mathbf{x}; \mathbf{w}) = \left\langle \mathbf{w}^{(1)}, \Psi^{(1)}(y_p, \mathbf{x}) \right\rangle, \tag{4.6}$$

and

$$V(y_p, y_q, \mathbf{x}; \mathbf{w}) = \left\langle \mathbf{w}^{(2)}, \Psi^{(2)}(y_p, y_q, \mathbf{x}) \right\rangle, \tag{4.7}$$

where $\Psi^{(1)}, \Psi^{(2)}$ are the unary and pairwise feature mappings respectively and $\langle \cdot, \cdot \rangle$ denotes inner products. Clearly we have $\mathbf{w} = \mathbf{w}^{(1)} \odot \mathbf{w}^{(2)}$ ($\odot$ stacks two vectors). We will show how to construct unary potentials over the learned deep features in the following.

4.2.3.1 Implementation Details

After obtaining the learned deep features, we define feature mappings upon them to construct the unary potential. Consider the image $\mathbf{x}$ with label $\mathbf{y}$, let $\mathbf{f}^p$ be the feature vector associated with the p-th superpixel, and K is the number of classes (possible labels). Then we define the unary feature mappings as

$$\Psi^{(1)}(y_p, \mathbf{x}) = [\delta(y_p = 1)\mathbf{f}_p^\top, \dots, \delta(y_p = K)\mathbf{f}_p^\top]^\top, \tag{4.8}$$

where $\delta(\cdot)$ is an indicator function which equals 1 if the input is true and 0 otherwise. In the case of multi-class, the dimension of $\Psi^{(1)}(y_p, \mathbf{x})$ can be too large when $\mathbf{f}_p$ is high dimensional. To address this issue, we first train an one-vs-all multi-class linear SVM over the features of superpixels, and then use the output confidence scores of the p-th superpixel as $\mathbf{f}_p$ to construct the unary potential. Similar strategy is used in [26, 51]. Accordingly, the pairwise feature mapping is constructed as

$$\Psi^{(2)}(y_p, y_q, \mathbf{x}) = L_{pq} \cdot \delta(y_p \neq y_q), \tag{4.9}$$

where L_{pq} can be the shared boundary length or inversed color difference between neighbouring superpixels.

The energy function in Eq. (4.2) can then be written as

$$E(\mathbf{y}, \mathbf{x}; \mathbf{w}) = \left\langle \mathbf{w}^{(1)}, \sum_{p \in \mathcal{N}} \Psi^{(1)}(y_p, \mathbf{x}) \right\rangle + \left\langle \mathbf{w}^{(2)}, \sum_{(p,q) \in \mathcal{S}} \Psi^{(2)}(y_p, y_q, \mathbf{x}) \right\rangle . \tag{4.10}$$

To deal with the unbalanced appearance of different categories in the dataset, we define $\Delta(\mathbf{y}_i, \mathbf{y})$ as the weighted Hamming loss, which weighs errors for a given class inversely proportional to the frequency it appears in the training data, similar to [26]. We use the method of [28] to solve the inference in Eq. (4.5).

4.2.4 Inference with Co-Occurrence Pairwise Potentials

To fully exploit context information, we consider the frequency of co-occurred object pairs in different spatial layouts during the inference. On one hand, this prefers labelling of frequently co-occurred label pairs in a certain spatial relation; while on the other hand, it excludes unreasonable labellings of co-occurrences (mutex constraint, similar as [79]), such as grass, water or road appearing above sky. Different from the mutex constraint used in [79], we incorporate the co-occurrence constraint into the pairwise term by devising spatially related co-occurrence pairwise potentials. We consider four spatial relations of the adjacent superpixel pairs: p is above q, p is below q, p is left to q and p is right to q. Then the feature mapping for the pairwise potential in Eq. (4.10) is written as:

$$\begin{aligned}\sum_{(p,q) \in \mathcal{S}} \Psi^{(2)}(y_p, y_q, \mathbf{x}) = & \sum_{(p,q) \in \mathcal{S}_1} \Psi_1^{(2)}(y_p, y_q, \mathbf{x}) + \sum_{(p,q) \in \mathcal{S}_2} \Psi_2^{(2)}(y_p, y_q, \mathbf{x}) \\ & + \sum_{(p,q) \in \mathcal{S}_3} \Psi_3^{(2)}(y_p, y_q, \mathbf{x}) + \sum_{(p,q) \in \mathcal{S}_4} \Psi_4^{(2)}(y_p, y_q, \mathbf{x}). \end{aligned} \tag{4.11}$$

where $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_4$ are the sets of edges where p and q are in the spatial relations "above", "below", "left" and "right" respectively, and $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4$, and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for $i \neq j, i, j = 1, 2, 3, 4$.

To construct the co-occurrence pairwise potentials, we assume that the training data is sufficiently large. The pairwise potentials in Eq. (4.11) can then be written as:

$$\Psi_i^{(2)}(y_p, y_q, \mathbf{x}) = L_{pq} \cdot \delta(y_p \neq y_q) \cdot g_i(y_p, y_q), i = 1, 2, 3, 4. \tag{4.12}$$

where $g_i(y_p, y_q) = \frac{1}{f^i_{co-occur}(y_p,y_q)}$ with $f^i_{co-occur}(y_p, y_q) = \frac{N^i_{pq}}{N_{pq}}$. Here, N_{pq} is the number of training images in which y_p and y_q co-exist, and N^i_{pq} $(i = 1, 2, 3, 4)$ are the numbers of training images in which y_p and y_q appear in the four spatially related neighbouring superpixels respectively. If $N^i_{pq} = 0$, meaning that y_p and y_q never appear in the ith spatial relation, then $g_i(y_p, y_q) = inf$, preventing the inference to yield such pair labellings. Intuitively, this would prefer labellings that frequently co-occurred in certain spatial relations in the training data, and avoid those mutual exclusion labellings, such as grass appear above sky.

Note that the mutex constraint used in [79] can be seen a special case of our co-occurrence pairwise potentials, as it is equivalent to ours when we set $g_i(y_p, y_q) = inf$ for $f^i_{co-occur}(y_p, y_q) = 0$ and $g_i(y_p, y_q) = 1$ for $f^i_{co-occur}(y_p, y_q) \neq 0$. We will provide experimental comparison with this case in Section 4.3.3. After learning the CRF using SSVM, we construct co-occurrence pairwise potentials for prediction. We add a trade-off parameter α multiplied to the pairwise term and tune it from 0.5 to 2 based on validation sets.

4.3 Experiments

To demonstrate the effectiveness of the proposed method, we first compare the CNN features with the traditional bag-of-words feature and an unsupervised feature learning method [59] as well as evaluate the impact of depths to the performance of the CNN features in Sec. 4.3.2. We then compare with state-of-the-art methods on several image segmentation datasets in Sec. 4.3.3.

4.3.1 Experimental Setup

For the CNN features, we use the model trained on ImageNet provided by Caffe [82]. The network follows the famous AlexNet [4], and is composed of 5 convolutional layers and 2 fully connected layers together with a soft-max layer.

We evaluate the performance of the proposed method on Weizmann horse, Graz-02, MSRC-21, Standford Background and PASCAL VOC 2011 segmentation challenge dataset. The Weizmann horse dataset[2] consists of 328 horse images from various backgrounds, with groundtruth masks available for each image. We use the same data split as in [25], [60], and we simply resize the images to 256×256. The Graz-02 dataset[3] contains

[2]http://www.msri.org/people/members/eranb/
[3]http://www.emt.tugraz.at/~pinz/

3 categories (bike, car and people). This dataset is considered challenging as the objects appear at various background and with different poses. We follow the evaluation protocol in [62] to use 150 for training and 150 for testing for each category.

The MSRC-21 dataset [50] is a popular multi-class segmentation benchmark with 591 images containing objects from 21 categories. We follow the standard split to divide the dataset into training/validation/test subsets. The Standford Background dataset [83] is a collection of outdoor scene images from several publicly available datasets, which consists of 715 images coming from 8 categories. Each image is approximately 320×240 pixels and contains at least one foreground object. We use the same evaluation protocol as in [83] to report 5-fold cross validation accuracy (global and per-category). The VOC 2011 dataset consists of images from 20 objects and background. We train on the training set and test on the validation images. The performance are quantified by the standard VOC measure [84].

We start with over-segmenting the images into superpixels using SLIC [63] (~ 700 superpixels per image) and then compute features within regions around each superpixel centroid with different block sizes (36×36, 48×48, 64×64, 72×72). We construct four types of pairwise features also using different block sizes to enforce spatial smoothness, which are color difference in LUV space, color histogram difference, texture difference in terms of LBP operators as well as shared boundary length [51]. Training our model on the MSRC-21 dataset takes around 2 hours. During prediction, the inference is rather efficient (less than 1 sec per image).

4.3.2 Baseline Comparison

To show the superiority of the deep CNN over the unsupervised feature learning, we compare with the traditional bag-of-word (BoW) feature and features learned from a popular unsupervised feature learning method [59]. Specifically, we first extract dense SIFT descriptors within each superpixel block and then quantize them into BoW feature using nearest neighbour search with a codebook size of 400. For the unsupervised feature learning, we first learn a dictionary of size 400 and patch size 6×6 based on the evaluated image dataset using Kmeans, and then use the soft threshold coding [59] to encode patches extracted from each superpixel block. The final feature vectors are obtained by performing a three-level max pooling over the superpixel block.

To investigate the roles of different layers in the proposed segmentation method, we evaluate the performance of features from the last three layers of the CNN model (5th, 6th and 7th layers). The 5th layer (with dimension 9216) is the last convolutional layer of the CNN. The 6th layer (with dimension 4096) is a fully connected layer follows the

Metric	SVM					SSVM				
	BoW	UFL	L5	L6	L7	BoW	UFL	L5	L6	L7
Sa	87.5	89.3	90.1	**92.7**	91.1	92.3	94.6	95.2	**95.7**	95.1
So	58.7	63.6	68.9	**74.6**	72.9	72.5	80.1	82.4	**84.0**	82.3

Table 4.1: Performance of different methods on the Weizmann horse dataset. CNN features perform significantly better than the traditional BoW feature and the unsupervised feature learning method, with features of the 6th layer performing marginally better than other compared layers. SSVM based CRF learning performs far better than SVM.

5th layer and the 7th (with dimension 4096) is the final layer of the feature learning pipeline. Using the two types of learned features, we compare the SSVM based CRF learning with a baseline method, namely, linear SVM, which classifies each superpixel independently without CRF learning. The datasets used in this section are Weizmann horse, Graz-02 and MSRC-21. We use BoW to denote the bag-of-words feature, UFL represent the unsupervised feature learning method, and L5, L6, L7 are CNN features of the 5th, 6th and 7th layer respectively.

4.3.2.1 Weizmann Horse

We first test on the Weizmann horse dataset. The performance are quantified by the global pixel-wise accuracy S_a and the foreground intersection over union score S_o, similar as in [25]. S_a measures the percentage of pixels correctly classified while S_o directly reflects the segmentation quality of the foreground. The compared results are reported in Table 4.1. We can observe that the CNN features perform consistently better than the bag-of-words feature and the unsupervised learned feature in both SVM and SSVM. By enforcing smoothness term, SSVM based CRF learning obtain far better segmentations than simple binary model as SVM. Furthermore, features of different depths exhibit almost similar performance with the 6th layer performs marginally better than the other compared layers in both SVM and SSVM. In Figure 4.3, we show some examples of qualitative evaluation, which yields conclusions that are in accordance with those from Table 4.1.

4.3.2.2 Graz-02

For a comprehensive evaluation, we use two measurements to quantify the performance of our method on the Graz-02 dataset, which are intersection over union score and the pixel accuracy (including foreground and background). We report the results in Table 4.2. It can be observed that feature learning methods generally outperform the traditional bag-of-words feature, with CNN features standing as the best. As for different depths, feature of the 6th layer consistently outperforms all the other compared layers in both

Category		bike	car	people	bike	car	people
		intersection/union (foreground, background) (%)			pixel accuracy (foreground, background) (%)		
	BoW	66.5 (50.4, 82.7)	66.8 (42.2, 91.5)	64.0 (41.9, 86.2)	79.0 (67.9, 90.2)	75.8 (55.2, 96.3)	74.5 (55.4, 93.7)
	UFL	69.7 (55.0, 84.5)	73.1 (52.7, 93.4)	61.4 (37.2, 85.6)	81.7 (72.4, 91.1)	80.9 (64.4, 97.4)	71.2 (48.2, 94.3)
SVM	L5	74.6 (62.4, 86.8)	76.0 (58.4, 93.7)	65.9 (47.0, 84.9)	86.3 (81.2, 91.4)	86.3 (76.2, 96.4)	80.9 (72.4, 89.4)
	L6	**77.7** (66.7, 88.6)	**78.1** (61.8, 94.5)	**68.9** (51.1, 86.6)	**88.4** (84.4, 92.5)	**87.2** (77.3, 97.0)	**83.0** (75.2, 90.8)
	L7	77.1 (66.0, 88.2)	77.6 (60.8, 94.3)	68.4 (50.5, 86.3)	88.2(84.1, 92.2)	86.6 (76.3, 97.0)	82.8 (75.1, 90.5)
	BoW	70.9 (56.6, 85.2)	75.7 (57.2, 94.1)	71.3 (53.5, 89.1)	82.5 (73.5, 91.6)	83.2 (68.9, 97.6)	81.4 (68.2, 94.7)
	UFL	74.2 (61.5, 86.9)	77.9 (60.9, 94.9)	70.9 (53.0, 88.8)	85.4 (78.6, 92.1)	83.8 (69.3, 98.4)	81.5 (68.9, 94.2)
SSVM	L5	81.6 (72.3, 90.8)	84.5 (72.6, 96.4)	75.4 (61.1, 89.7)	91.0 (88.0, 93.9)	90.6 (82.8, 98.3)	88.8 (85.3, 92.3)
	L6	**82.0** (73.1, 91.0)	**85.6** (74.5, 96.6)	**79.6** (67.2, 92.1)	**91.6** (89.5, 93.7)	**91.4** (84.4, 98.4)	**90.0** (85.1, 94.8)
	L7	81.7 (72.6, 90.8)	85.1 (73.7, 96.5)	76.0 (62.0, 90.0)	91.3 (89.0, 93.6)	91.2 (84.0, 98.4)	89.3 (86.1, 92.4)

Table 4.2: Compared results of the average intersection-over-union score and average pixel accuracy on the Graz-02 dataset. We include the foreground and background results in the brackets. CNN features perform significantly better than the traditional BoW feature and the unsupervised feature learning, with features of the 6th layer performing the best among the compared layers in both SVM and SSVM. SSVM based CRF learning performs far better than SVM.

		building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Average**	**Global**
	BoW	61	87	60	29	47	83	56	66	60	54	66	53	68	7	61	33	51	27	35	19	29	50.1	62.7
	UFL	57	95	77	55	59	96	56	70	61	41	67	65	31	17	67	30	75	52	26	32	6	54.1	69.5
SVM	L5	77	91	86	79	83	95	80	85	81	76	84	81	52	55	82	64	83	81	63	68	25	74.8	82.1
	L6	78	95	88	81	87	95	83	88	86	75	86	83	55	58	86	69	85	84	67	72	28	77.6	84.9
	L7	80	98	89	82	91	96	86	87	89	76	86	86	58	59	87	68	87	85	67	74	31	**79.0**	**86.0**
	BoW	65	89	87	64	74	90	58	75	78	56	85	54	55	6	60	14	66	50	35	38	8	57.4	70.7
	UFL	70	97	87	69	77	98	45	75	77	49	86	82	26	12	81	40	79	49	14	47	1	60.1	76.1
SSVM	L5	71	97	92	86	95	98	94	82	93	80	95	92	76	65	94	72	89	87	71	78	51	83.9	86.9
	L6	71	94	93	89	96	96	95	85	92	85	95	90	71	68	94	77	92	93	75	81	54	85.8	87.3
	L7	71	95	92	87	98	97	97	89	95	85	96	94	75	76	89	84	88	97	77	87	52	**86.7**	**88.5**

Table 4.3: Segmentation results on the MSRC-21 dataset. We report the pixel-wise accuracy for each category as well as the average per-category scores and the global pixel-wise accuracy (%). Deep learning performs significantly better than the BoW feature and the unsupervised feature learning, with SSVM based CRF learning using features of the 7th layer of the deep CNN achieving the best results. SSVM based CRF learning performs far better than SVM.

SVM and SSVM, which is in accordance with the conclusion of [40]. We show some segmentation examples in Figure 4.4, from which we can see that SSVM based CRF learning with CNN features produces segmentation similar to ground truth.

4.3.2.3 MSRC-21

The compared results with features of different layers are summarized in Table 4.3. Different from the binary cases as Weizmann horse and Graz-02, features of the 7th layer perform the best, which may results from the fact that MSRC is much more difficult due to the large number of categories. Figure 4.5 shows some qualitative results of SSVM based CRF learning with different features, from which similar conclusions can be drawn.

Method	Sa	So
Levin & Weiss [64]	95.5	-
Cosegmentation [65]	80.1	-
Bertelli *et al.* [25]	94.6	80.1
Kuttel *et al.* [60]	94.7	-
Ours	**95.7**	**84.0**

Table 4.4: State-of-the-art comparison of segmentation performance (%) on the Weizmann horse dataset.

Method	bike	car	people	average
Marszalek & Schimid [62]	61.8	53.8	44.1	53.2
Fulkerson *et al.* [66]	66.4	54.7	51.4	57.5
Aldavert *et al.* [67]	71.9	62.9	58.6	64.5
Kuettel *et al.* [60]	63.2	74.8	66.4	68.1
Ours	**84.5**	**85.4**	**80.4**	**83.4**

Table 4.5: State-of-the-art comparison of segmentation performance (%) on the Graz-02 (right) dataset.

4.3.3 State-of-the-art Comparison

Based on the above evaluation, we choose the best performed 6th layer for the binary (Weizmann horse and Graz-02) and 7th layer features for the multi-class datasets (MSRC-21, Stanford Background and VOC 2011) to learn CRF and compare with state-of-the-art results in this section. For the three multi-class datasets, we add the results of incorporating the mutex and co-occurrence pairwise potentials introduced in Sec. 4.2.4.

4.3.3.1 Binary Datasets

Table 4.4 and Table 4.5 show the compared segmentation results on the Weizmann horse and the Graz-02 datasets. We use a different evaluation metric for comparison on the Graz-02 dataset, which is the F-score ($F = 2pr/(p + r)$, where p is the precision and r is the recall) for each class and the average over classes. In both cases, our method outperforms all the compared methods.

4.3.3.2 Multi-class Datasets

The compared global and average per-category pixel accuracies on the MSRC-21 and the Stanford Background datasets are summarized in Table 4.6 and Table 4.7 respectively. On the MSRC dataset, our method outperforms all the methods except [79]. When incorporated with mutex or co-occurrence pairwise potentials in inference, we obtain further improvements. As expected, the co-occurrence potentials outperform the mutex potentials. [79] performs slightly better than ours in terms of global accuracy (they did not report average per-category accuracy), which may results from the fact that they use a fully connected CRF while ours are not. We show the confusion matrix of the predictions of our method using the co-occurrence pairwise potentials in Fig. 4.11. As we can see, our method achieves high accuracies on all of the classes.

As for the Stanford Background dataset, we can see that our method performs better than [41] and outperforming all the others. The work of [41] trains a 3-stage multiscale convolutional network on the training images while we directly transfer the deep CNN trained on the ImageNet to here sparing the effort of network training. Adding mutex potentials to our method do not bring any performance boost. By further investigations, we found that this is because there is only eight categories (one of which is the ambiguous foreground category) in this dataset, which leads to the fact that the only mutex information obtained is that grass, water and road can not appear above sky. Instead, our co-occurrence potentials perform much better, leading to further performance boost. We show some segmentation examples in Fig. 4.6. In Fig. 4.9, we show the confusion matrix of our predictions for a single run. As we can see, our method yields accurate predictions on most of the categories, with the leading confusing predictions are mountain as tree and mountain as grass. The predictions on the "mountain" category is not so satisfactory. By further investigations, we find out that the reason is due to its highly scarce occurrence in the training data, as shown in Fig. 4.10.

The segmentation results on the PASCAL VOC 2011 validation dataset are reported in Table 4.8 as the standard intersection-over-union score [85]. In [40], Girshick *et al.* achieved an average score of 47.9 by using augmented training data and extra annotation set. Here we did not use any extra dataset but only the VOC training set. By introducing mutex or co-occurrence pairwise potentials, constant improvements are observed on most of the categories. As expected, our co-occurrence potential again outperforms the mutex potential. In Table 4.9, we compare with the recent work of Carreira *et al.* [86], which performed evaluations with the same settings as ours (using the train/val set). Our method achieves the same accuracy as [86]. Note that the dimension of the feature descriptors used in [86] is tens of thousands of (33589) while ours is 4096. In Fig. 4.12, We show the confusion matrix of the predictions produced by our method with co-occurrence pairwise potentials. As we can see, most of the wrong predictions are made towards the background category, with following up confusing categories are predicting dog as cat, cow as horse, bicycle as motorbike and chair as sofa. These are reasonable since the appearances of these category pairs are similar, sometimes even confusing to human observers. We show some qualitative examples and some failure cases in Fig. 4.7 and Fig. 4.8, which validate the conclusions we made from the confusion matrix.

Method	Global (%)	Average (%)
Shotton *et al.* [50]	72	67
Ladicky *et al.* [68]	86	75
Munoz *et al.* [87]	78	71
Gonfaus *et al.* [69]	77	75
Lucchi *et al.* [26]	73	70
Yao *et al.* [71]	86.2	79.3
Lucchi *et al.* [70]	83.7	78.9
Ladicky *et al.* [78]	87	77
Roy *et al.* [79]	**91.5**	-
Ours	88.5	86.7
Ours (mutex)	90.3	89.2
Ours (co-occur)	91.1	**90.5**

Table 4.6: State-of-the-art comparison of global and average per-category pixel accuracy on the MSRC-21 dataset.

Method	Global (%)	Average (%)
Gould *et al.* [83]	76.4	-
Munoz *et al.* [87]	76.9	66.2
Lempitsky *et al.* [88]	81.9	72.4
Farabet *et al.* [41]	81.4	76.0
Roy *et al.* [79]	81.1	-
Ours	82.6	76.2
Ours (mutex)	82.6	76.3
Ours (co-occur)	**83.5**	**76.9**

Table 4.7: State-of-the-art comparison of global and average per-category pixel accuracy on the Stanford Background dataset.

VOC 2011 val	mean	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	house	mbike	person	plant	sheep	sofa	train	tv
Ours	31.9	78.3	43.9	20.4	23.2	22.7	24.6	42.2	41.0	36.1	12.6	24.9	19.8	25.0	23.8	38.6	53.3	20.0	36.6	20.2	38.1	24.6
Ours (mutex)	36.3	79.8	53.1	**23.8**	**26.4**	**28.8**	**28.6**	51.6	48.2	**37.8**	13.1	29.7	22.3	**28.4**	29.6	45.2	52.7	21.0	46.2	20.9	46.2	29.6
Ours (co-occur)	**38.3**	**81.5**	**55.7**	23.6	24.0	27.7	27.3	**52.8**	**54.1**	37.1	**14.9**	**37.1**	**28.6**	22.9	**33.1**	**49.7**	**54.2**	**27.4**	**49.3**	**22.3**	**49.3**	**30.9**

Table 4.8: Results of per-category and mean intersection-over-union score (%) on the PASCAL VOC 2011 validation dataset. Best results are bold faced.

Method	Mean (%)
HOG [86]	14.1
SIFT-PCA-FISHER [86]	31.9
O_2P [86]	**38.3**
Ours (co-occur)	**38.3**

Table 4.9: Comparison of the mean intersection-over-union score (%) on the PASCAL VOC 2011 validation dataset.

4.4 Conclusion

We propose to learn CRF using SSVM based on features learned from a pre-trained deep convolutional neural network for image segmentation. The deep CNN is trained on ImageNet and proved to perform exceptionally well when transferred to object segmentation. We learn the CRF in the max-margin framework by SSVM, and then conduct inference with co-occurrence pairwise potentials incorporated. Extensive experimental evaluations on the Weizmann horse, Graz-02, MSRC-21, Stanford Background and the

PASCAL VOC 2011 dataset demonstrate the advantages of our method and provide new baselines for further research.

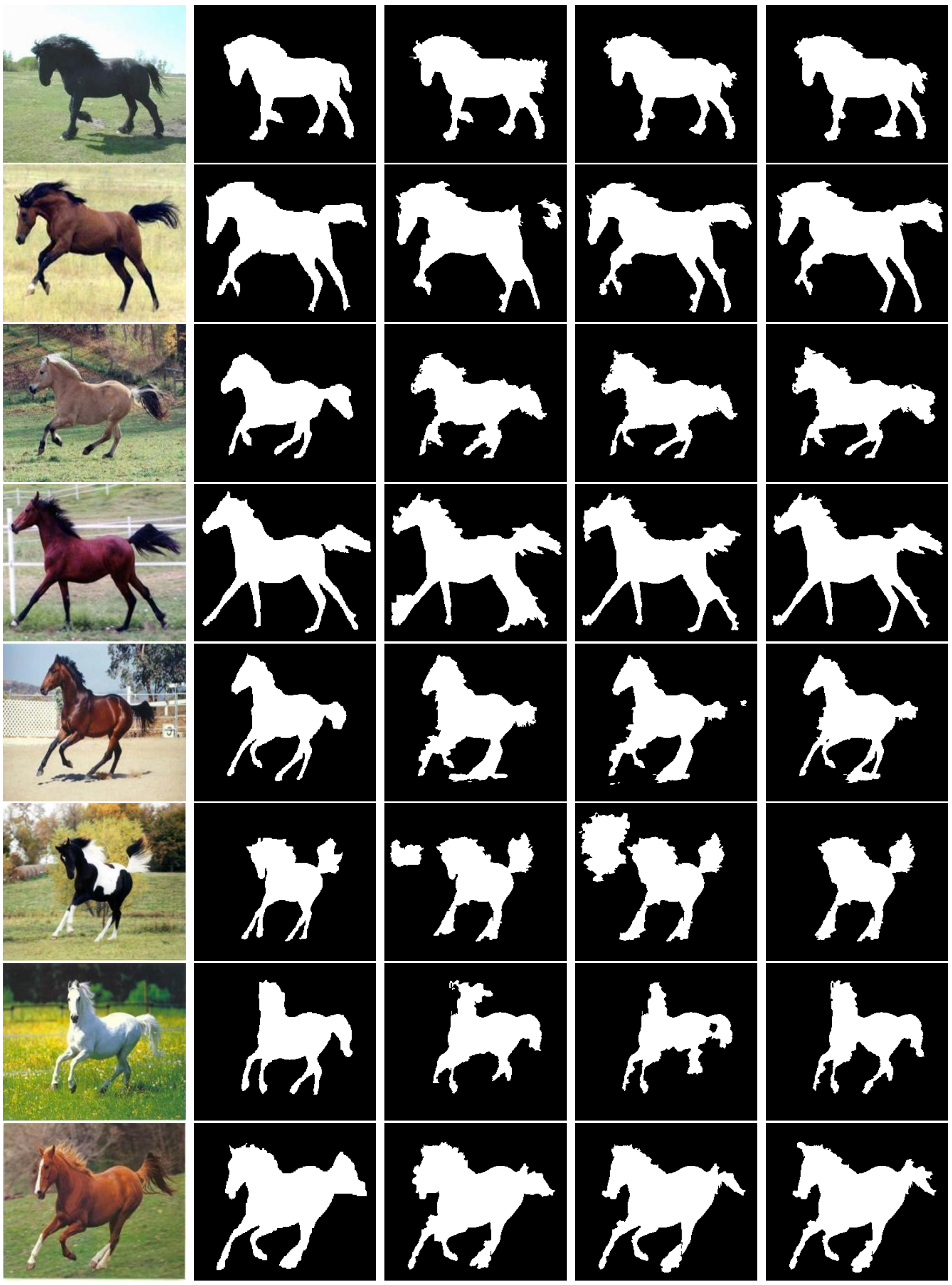


Figure 4.3: Segmentation examples on Weizmann horse. 1st column: Test images; 2nd column: Ground truth; 3rd column: Predictions produced by SSVM based CRF learning with bag-of-words feature; 4th column: Predictions produced by SSVM based CRF learning with unsupervised feature learning; 5th column: Predictions produced by SSVM based CRF learning with the 6th layer CNN features.

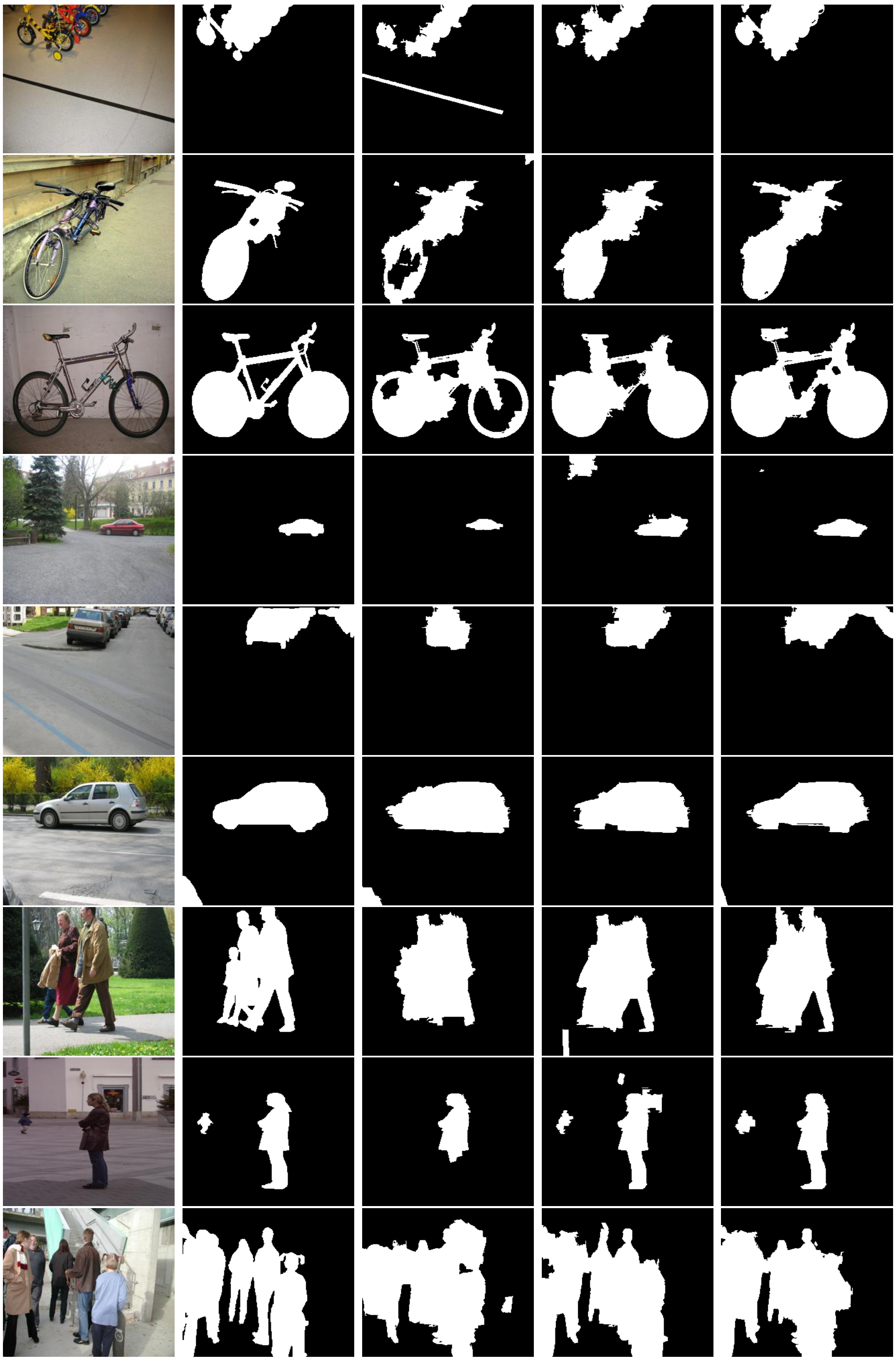


Figure 4.4: Segmentation examples on the Graz-02 dataset. 1st column: Test images; 2nd column: Ground truth; 3rd column: Segmentation results produced by SSVM based CRF learning with bag-of-words feature; 4th column: Segmentation results produced by SSVM based CRF learning with unsupervised feature learning; 5th column: Segmentation results produced by SSVM based CRF learning with the 6th layer CNN features.

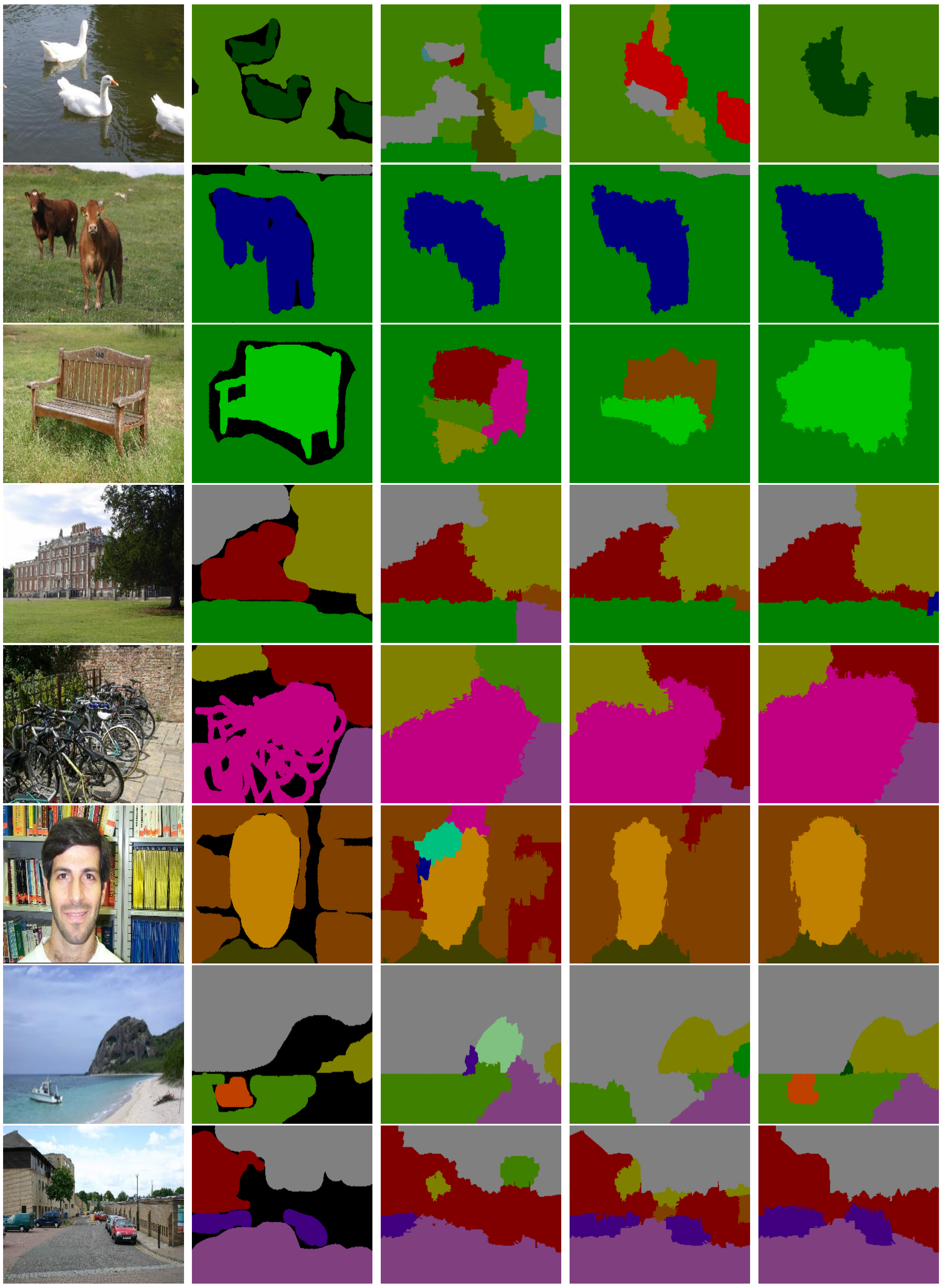


Figure 4.5: Segmentation examples on the MSRC-21 dataset. 1st column: Test images; 2nd column: Ground truth; 3rd column: Predictions produced by SSVM based CRF learning with bag-of-words feature; 4th column: Predictions produced by SSVM based CRF learning with unsupervised feature learning; 5th column: Predictions results produced by our method with co-occurrence pairwise potentials.

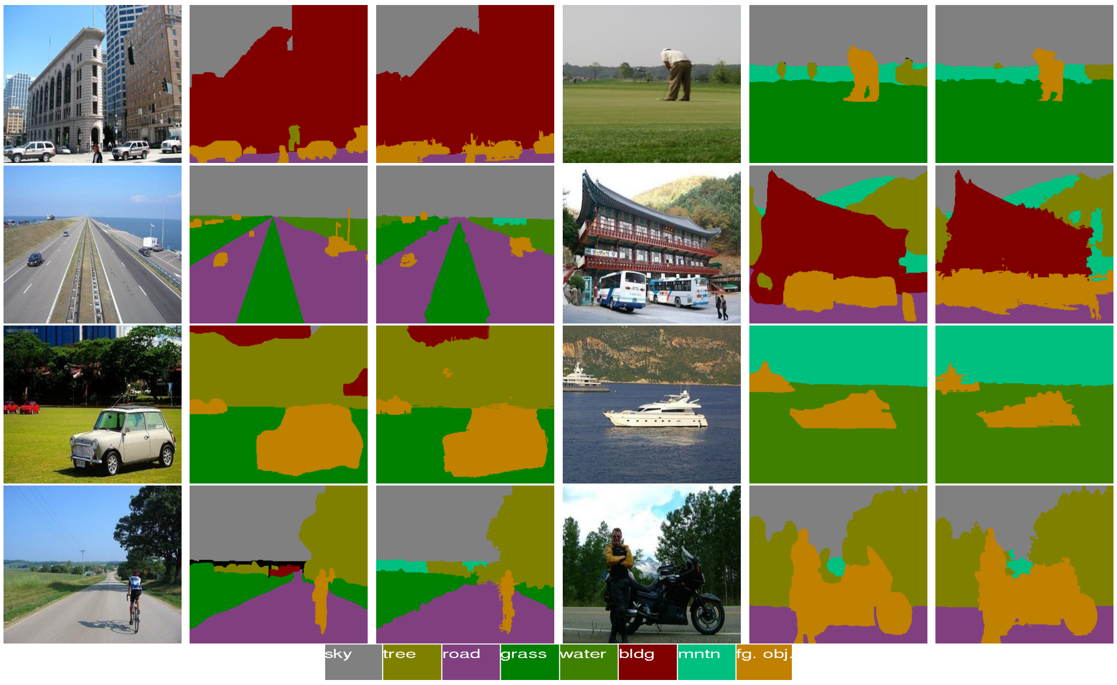


Figure 4.6: Segmentation examples on the Stanford Background dataset. 1st and 4th columns: Test images; 2nd and 5th columns: Ground truth; 3rd and 6th columns: Predictions produced by our method with co-occurrence pairwise potentials.

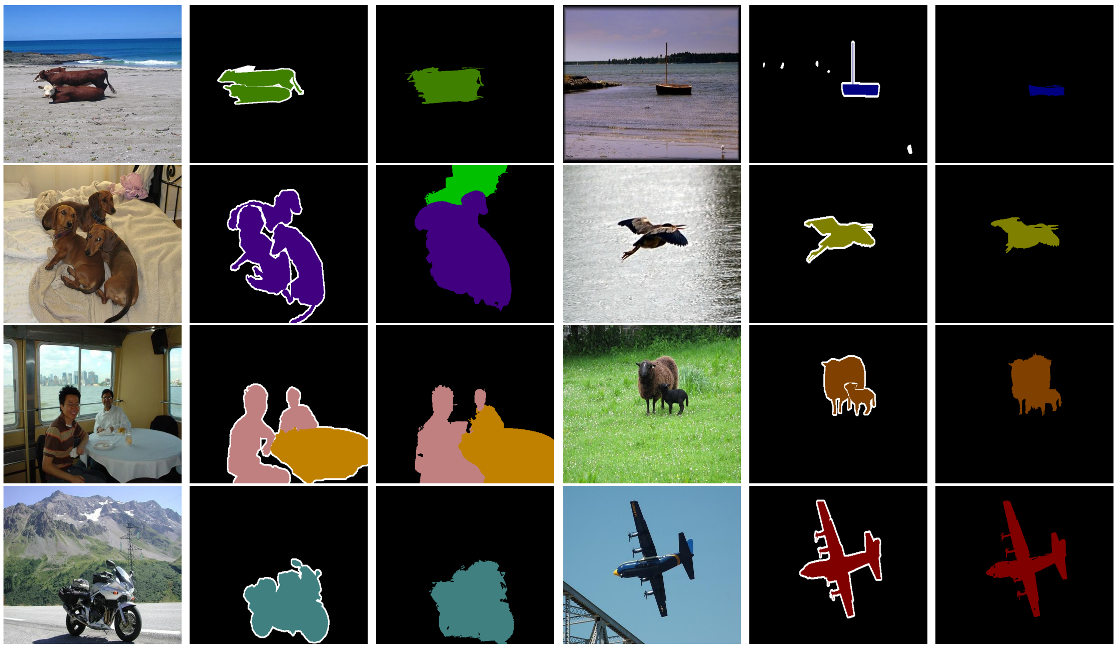


Figure 4.7: Segmentation examples on the PASCAL VOC 2011 dataset. 1st and 4th columns: Test images; 2nd and 5th columns: Ground truth; 3rd and 6th columns: Predictions produced by our method with co-occurrence pairwise potentials.

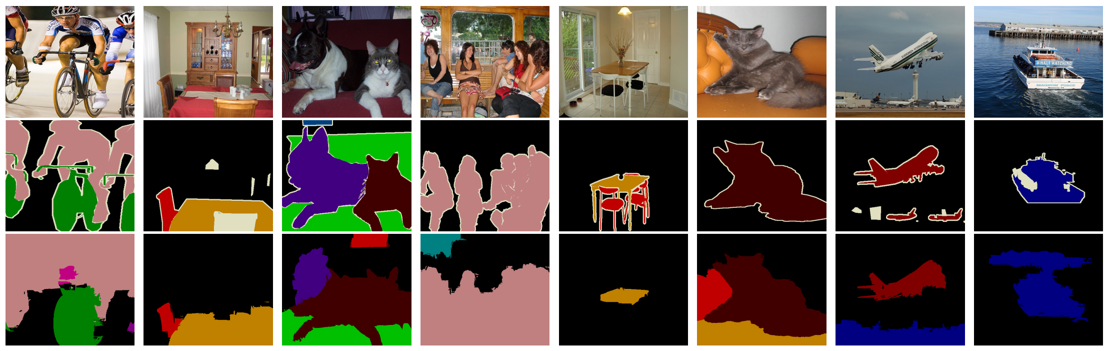
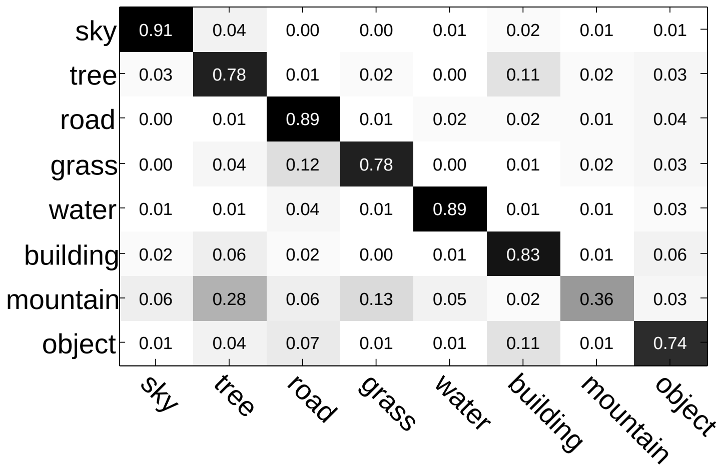


Figure 4.8: Failure examples on the VOC 2011 dataset. 1st row: Test images; 2nd row: Ground truth; 3rd row: Segmentation results produced by our method with co-occurrence pairwise potentials.



	sky	tree	road	grass	water	building	mountain	object
sky	0.91	0.04	0.00	0.00	0.01	0.02	0.01	0.01
tree	0.03	0.78	0.01	0.02	0.00	0.11	0.02	0.03
road	0.00	0.01	0.89	0.01	0.02	0.02	0.01	0.04
grass	0.00	0.04	0.12	0.78	0.00	0.01	0.02	0.03
water	0.01	0.01	0.04	0.01	0.89	0.01	0.01	0.03
building	0.02	0.06	0.02	0.00	0.01	0.83	0.01	0.06
mountain	0.06	0.28	0.06	0.13	0.05	0.02	0.36	0.03
object	0.01	0.04	0.07	0.01	0.01	0.11	0.01	0.74

Figure 4.9: Confusion matrix of the predictions produced by our method for a single run on the StanfordBackground dataset.

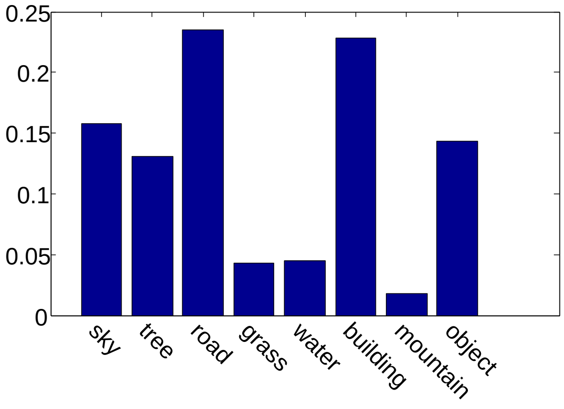
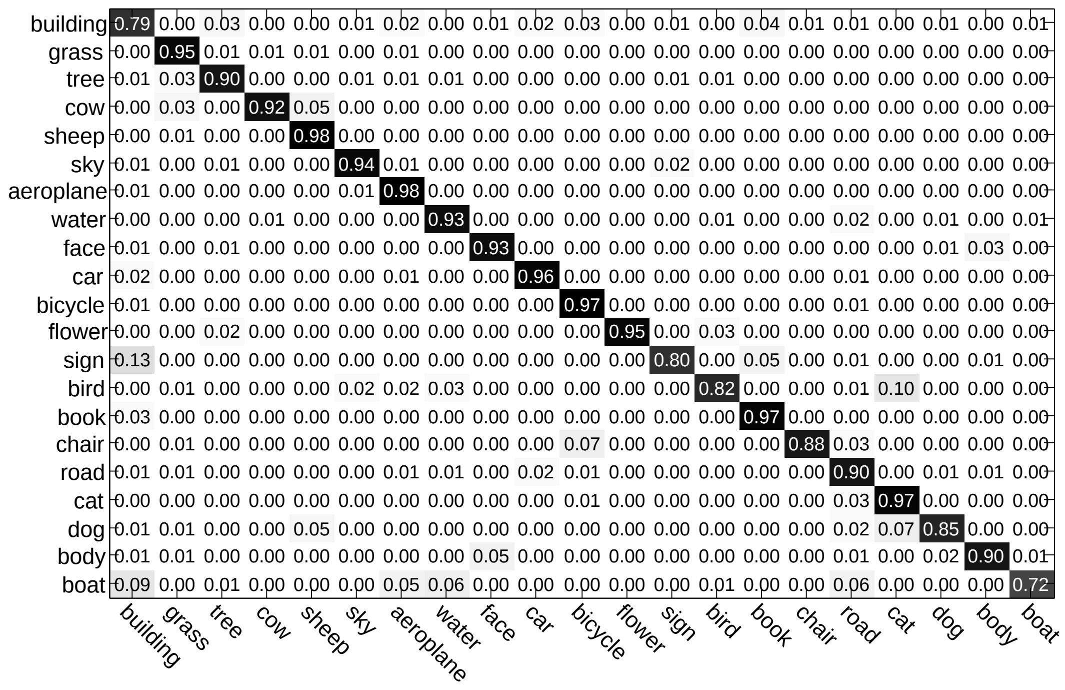
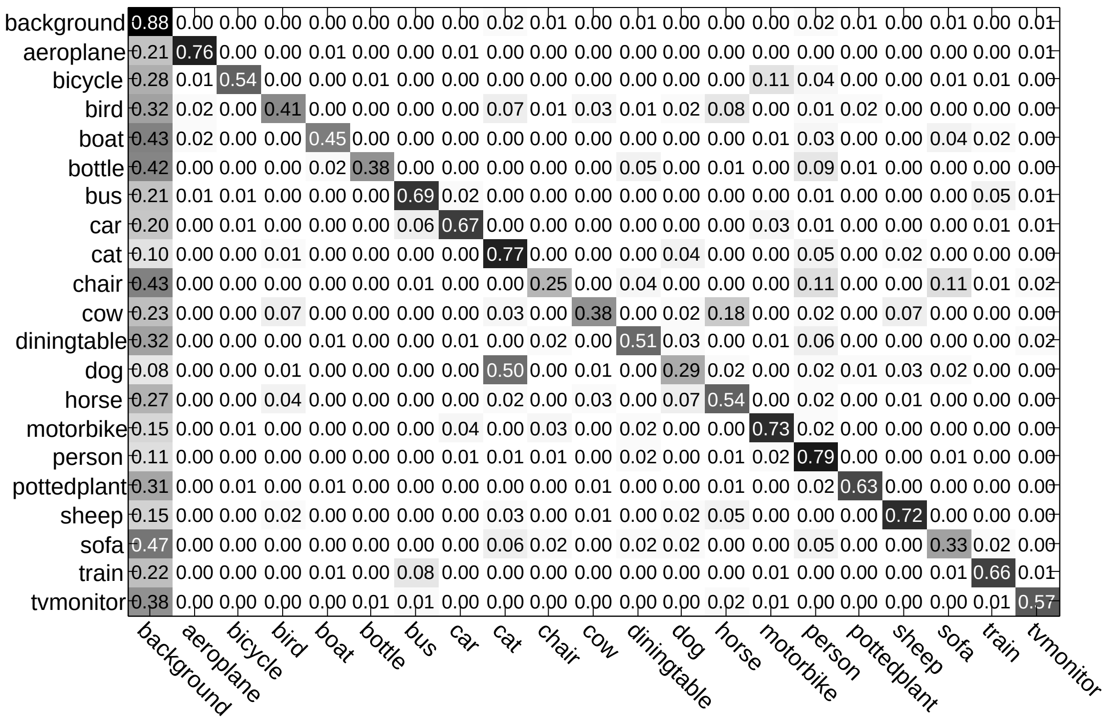


Figure 4.10: Occurrence frequencies of different categories in the training data of the Stanford-Background dataset.



	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat
building	0.79	0.00	0.03	0.00	0.00	0.01	0.02	0.00	0.01	0.02	0.03	0.00	0.01	0.00	0.04	0.01	0.01	0.00	0.01	0.00	0.01
grass	0.00	0.95	0.01	0.01	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tree	0.01	0.03	0.90	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
cow	0.00	0.03	0.00	0.92	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sheep	0.00	0.01	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sky	0.01	0.00	0.01	0.00	0.00	0.94	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
aeroplane	0.01	0.00	0.00	0.00	0.00	0.01	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
water	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.93	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.02	0.00	0.01	0.00	0.01
face	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.03	0.00
car	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
bicycle	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
flower	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sign	0.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.05	0.00	0.01	0.00	0.00	0.01	0.00
bird	0.00	0.01	0.00	0.00	0.00	0.02	0.02	0.03	0.00	0.00	0.00	0.00	0.00	0.82	0.00	0.00	0.01	0.10	0.00	0.00	0.00
book	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00
chair	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.88	0.03	0.00	0.00	0.00	0.00
road	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.01	0.01	0.00
cat	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.03	0.97	0.00	0.00	0.00
dog	0.01	0.01	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.07	0.85	0.00	0.00
body	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.02	0.90	0.01
boat	0.09	0.00	0.01	0.00	0.00	0.00	0.05	0.06	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.06	0.00	0.00	0.00	0.72

Figure 4.11: Confusion matrix of the predictions made by our method on the MSRC dataset.



	background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tvmonitor
background	0.88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.00	0.01	0.00	0.00	0.00	0.02	0.01	0.00	0.01	0.00	0.01
aeroplane	0.21	0.76	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
bicycle	0.28	0.01	0.54	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.04	0.00	0.00	0.01	0.01	0.00
bird	0.32	0.02	0.00	0.41	0.00	0.00	0.00	0.00	0.07	0.01	0.03	0.01	0.02	0.08	0.00	0.01	0.02	0.00	0.00	0.00	0.00
boat	0.43	0.02	0.00	0.00	0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.03	0.00	0.00	0.04	0.02	0.00
bottle	0.42	0.00	0.00	0.00	0.02	0.38	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.01	0.00	0.09	0.01	0.00	0.00	0.00	0.00
bus	0.21	0.01	0.01	0.00	0.00	0.00	0.69	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.05	0.01
car	0.20	0.00	0.01	0.00	0.00	0.00	0.06	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.01	0.00	0.00	0.00	0.01	0.01
cat	0.10	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.77	0.00	0.00	0.00	0.04	0.00	0.00	0.05	0.00	0.02	0.00	0.00	0.00
chair	0.43	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.25	0.00	0.04	0.00	0.00	0.00	0.11	0.00	0.00	0.11	0.01	0.02
cow	0.23	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.03	0.00	0.38	0.00	0.02	0.18	0.00	0.02	0.00	0.07	0.00	0.00	0.00
diningtable	0.32	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.02	0.00	0.51	0.03	0.00	0.01	0.06	0.00	0.00	0.00	0.00	0.02
dog	0.08	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.50	0.00	0.01	0.00	0.29	0.02	0.00	0.02	0.01	0.03	0.02	0.00	0.00
horse	0.27	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.02	0.00	0.03	0.00	0.07	0.54	0.00	0.02	0.00	0.01	0.00	0.00	0.00
motorbike	0.15	0.00	0.01	0.00	0.00	0.00	0.00	0.04	0.00	0.03	0.00	0.02	0.00	0.00	0.73	0.02	0.00	0.00	0.00	0.00	0.00
person	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.02	0.00	0.01	0.02	0.79	0.00	0.00	0.01	0.00	0.00
pottedplant	0.31	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.02	0.63	0.00	0.00	0.00	0.00
sheep	0.15	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.03	0.00	0.01	0.00	0.02	0.05	0.00	0.00	0.00	0.72	0.00	0.00	0.00
sofa	0.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.02	0.00	0.02	0.02	0.00	0.00	0.05	0.00	0.00	0.33	0.02	0.00
train	0.22	0.00	0.00	0.00	0.01	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.66	0.01
tvmonitor	0.38	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.01	0.57

Figure 4.12: Confusion matrix of the predictions produced by our method on the Pascal VOC 2011 dataset.

Chapter 5

Joint Learning of Continuous CRF and CNN for Single Image Depth Estimation

In this chapter, we propose to jointly learn a continuous CRF and CNN for the problem of depth estimation from single monocular images. Compared with depth estimation using multiple images such as stereo depth perception, depth from monocular images is much more challenging. Prior work typically focuses on exploiting geometric priors or additional sources of information, most using hand-crafted features. Recently, there is mounting evidence that features from deep convolutional neural networks (CNN) set new records for various vision applications. On the other hand, considering the continuous characteristic of the depth values, depth estimations can be naturally formulated as a continuous conditional random field (CRF) learning problem. Therefore, here we present a deep convolutional neural field model for estimating depths from single monocular images, aiming to jointly explore the capacity of deep CNN and continuous CRF. In particular, we propose a deep structured learning scheme which learns the unary and pairwise potentials of continuous CRF in a unified deep CNN framework. We then further propose an equally effective model based on fully convolutional networks and a novel superpixel pooling method, which is ~ 10 times faster, to speedup the patch-wise convolutions in the deep model. With this more efficient model, we are able to design deeper networks to pursue better performance.

Our proposed method can be used for depth estimation of general scenes with no geometric priors nor any extra information injected. In our case, the integral of the partition function can be calculated in a closed form such that we can exactly solve the log-likelihood maximization. Moreover, solving the inference problem for predicting depths

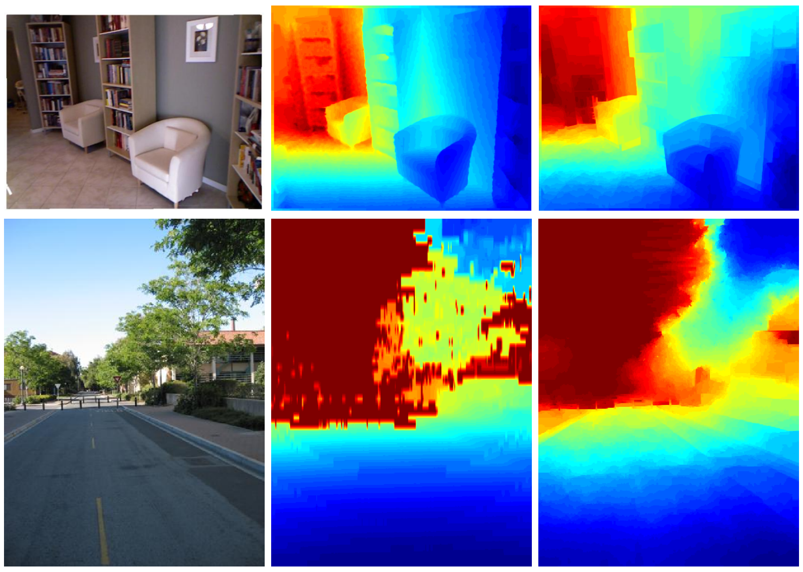


Figure 5.1: Examples of depth estimation results using the proposed deep convolutional neural fields model. First row: NYU v2 dataset; second row: Make3D dataset. From left to right: input image, ground-truth, our prediction.

of a test image is highly efficient as closed-form solutions exist. Experiments on both indoor and outdoor scene datasets demonstrate that the proposed method outperforms state-of-the-art depth estimation approaches.

5.1 Introduction

Estimating depth information from single monocular images depicting general scenes is an important problem in computer vision. Many challenging computer vision problems have proven to benefit from the incorporation of depth information, to name a few, semantic labellings [89], pose estimations [90]. Although the highly developed depth sensors such as Microsoft Kinect nowadays have made the acquisition of RGBD images affordable, most of the vision datasets commonly evaluated among the vision community are still RGB images. Moreover, outdoor applications still rely on LiDAR or other laser sensors due to the fact that strong sunlight can cause infrared interference and make depth information extremely noisy. This has led to wide research interest on the topic of estimating depths from single RGB images. Unfortunately, it is a notoriously ill-posed problem, as one captured image scene may correspond to numerous real world scenarios [9].

Whereas for humans, inferring the underlying 3D structure from a single image is effortless, it remains a challenging task for automated computer vision systems to do so since no reliable cues can be exploited, such as temporal information in videos, stereo correspondences, etc. Previous work mainly focuses on enforcing geometric assumptions, *e.g.*, box models, to infer the spatial layout of a room [5, 91] or outdoor scenes [6]. These models come with innate restrictions, which are limitations to model only particular scene structures and therefore are not applicable for general scene depth estimations. More recently, non-parametric methods [7] are explored, which consists of candidate images retrieval, scene alignment and then depth inference using optimizations with smoothness constraints. This is based on the assumption that scenes with semantically similar appearances should have similar depth distributions when densely aligned. However, this method is prone to propagate errors through the different decoupled stages and relies heavily on building a reasonable sized image database to perform the candidate retrieval. In recent years, efforts have been made towards incorporating additional sources of information, *e.g.*, user annotations [92], semantic labellings [89, 93]. In the recent work of [89], Ladicky *et al.* have shown that jointly performing depth estimation and semantic labelling can benefit each other. Nevertheless, all these methods use hand-crafted features.

In contrast to previous efforts, here we propose to formulate the depth estimation as a deep continuous Conditional Random Fields (CRF) learning problem, without relying on any geometric priors or any extra information. CRF [11] is a popular graphical model for structured output predictions. While extensively studied in classification (discrete) domains, CRF has been less explored for regression (continuous) problems. One of the pioneer work on continuous CRF can be attributed to [33], in which it was proposed for global ranking in document retrieval. Under certain constraints, they can directly solve the maximum likelihood optimization as the partition function can be analytically calculated. Since then, continuous CRF has been successfully applied for solving various structured regression problems, *e.g.*, remote sensing [94, 95], image denoising [95]. Motivated by these successes, we here propose to use it for depth estimation, given the continuous nature of the depth values, and learn the potential functions in a deep convolutional neural network (CNN).

Recent years have witnessed the prosperity of the deep CNN [38] since the breakthrough work of Krizhevsky *et al.* [96]. CNN features have been setting new records for a wide variety of vision applications [42]. Despite all the successes in classification problems, deep CNN has been less explored for structured learning problems, *i.e.*, joint training of a deep CNN and a graphical model, which is a relatively new and not well addressed problem. To our knowledge, no such model has been successfully used for depth estimations. Here, we bridge this gap by jointly exploring CNN and continuous CRF, denoting this new method as a deep convolutional neural field (DCNF).

Fully convolutional networks have recently been studied for dense prediction problems, *e.g.*, semantic labelling [97, 98]. Models based on fully convolutional networks have the advantage of highly efficient training and prediction. We here exploit this advance to speedup the training and prediction of our DCNF model. However, the feature maps produced by the fully convolutional models are typically much smaller than the input image size. This can cause problems for both training and prediction. During training, one needs to downsample the ground-truth maps, which may lead to information loss since small objects might disappear. In prediction, the upsampling operation will probably bring in degraded performance at the object boundaries. We therefore propose a novel superpixel pooling method to address this issue. It jointly exploits the strengths of highly efficient fully convolutional networks and the benefits of superpixels at preserving object boundaries.

To sum up, we highlight the main contributions of this work as follows.

- We propose a deep convolutional neural field (DCNF) model for depth estimations by exploring CNN and continuous CRF. Given the continuous nature of the depth values, the partition function in the probability density function can be analytically calculated, therefore we can directly solve the log-likelihood optimization without any approximations. The gradients can be exactly calculated in the back propagation training. Moreover, solving the MAP problem for predicting the depth of a new image is highly efficient since closed form solutions exist.
- We jointly learn the unary and pairwise potentials of the CRF in a unified deep CNN framework, which is trained using back propagation.
- We propose a faster model based on fully convolutional networks and a novel superpixel pooling method, which results in ~ 10 times speedup while producing similar prediction accuracy. With this more efficient model, which we refer as DCNF-FCSP, we are able to design very deep networks for better performance.
- We demonstrate that the proposed method outperforms state-of-the-art results of depth estimation on both indoor and outdoor scene datasets.

5.2 Related Work

Our method exploits the recent advances of deep nets in image classification [96, 99], object detection [100] and semantic segmentation [97, 98], for single view image depth estimations. In the following, we give a detailed literature review of the work that closely related to ours.

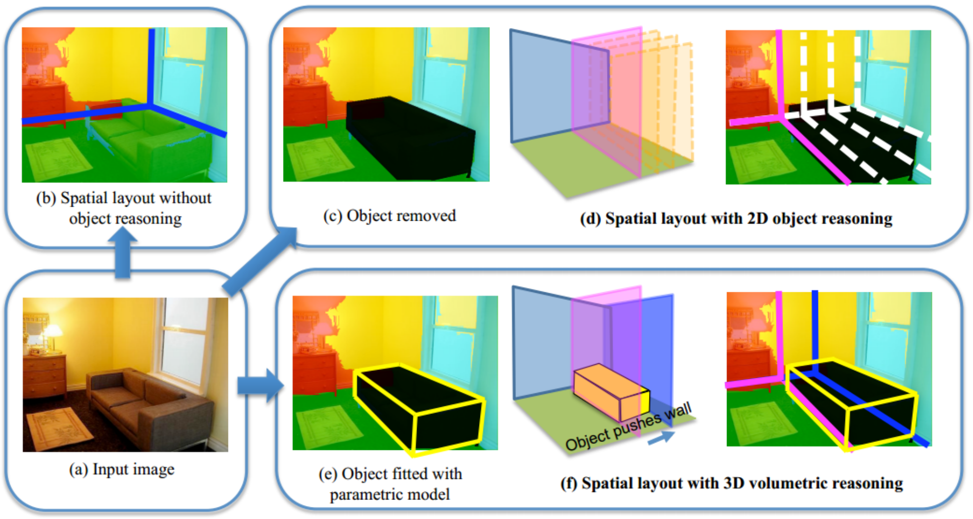

(b) Spatial layout without object reasoning
(c) Object removed
(d) Spatial layout with 2D object reasoning
(a) Input image
(e) Object fitted with parametric model
Object pushes wall
(f) Spatial layout with 3D volumetric reasoning

Figure 5.2: An illustration of the box model based methods for room layout estimation. Figure reproduced from [5].

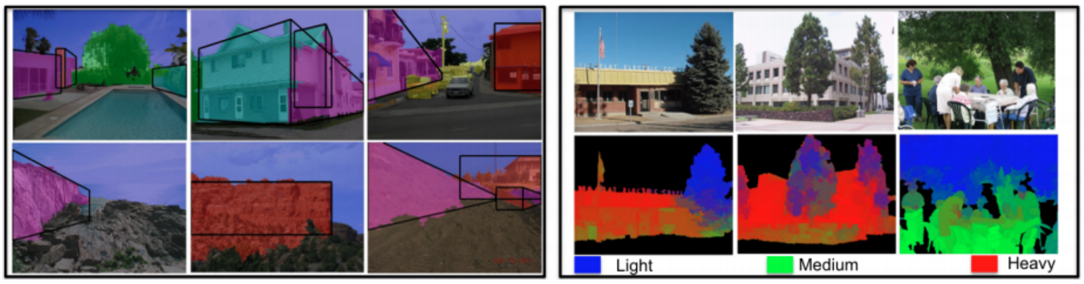


Figure 5.3: An illustration of the block model based methods for outdoor 3D scene understanding. Left: examples of extracted blocks; Right: examples of super-pixel based density estimation. Figure reproduced from [6].

5.2.1 Depth Perception in Vision

Traditional methods for recovering pixel-wise depths from RGB images generally focus on binocular vision, *i.e.*, stereopsis. In the work of [101], Scharstein *et al.* provide a comprehensive survey of popular dense two-frame stereo algorithms. Another vein of efforts rely on multiple images, including structure-from-motion [102] and depth from defocus [103]. These kinds of methods generally work by making use of the geometric or triangulation differences. When such cues are not available, *i.e.*, in the single image scenario, estimating depths based solely on monocular cues becomes more challenging. Next we focus on reviewing methods for estimating depths from single monocular images.

5.2.1.1 Depth Estimation from Single Monocular Images

In the literature, there exits a range of methods for the single image depth estimation problem, which we here roughly group into three categories and detail in the following.

Geometric methods Traditional methods generally rely on geometry transformations by considering texture or luminance variations. Examples include shape from texture [104, 105], shape from shading [106]. However, these methods are restricted to those scenes with relatively uniform texture or luminance variations, and not applicable to general scene images. Other efforts rely on known and fixed objects [107], *e.g.*, human faces, cars, for depth inference. This strategy requires parsing the image into constituent elements, which remains challenging and unreliable. Another type of geometric methods work by enforcing ad hoc assumptions, *e.g.*, parametric modelling such as box models, on the scene structure [5, 6, 91]. In [91] and [5], the authors employ 3D oriented box models to obtain estimates of the room layout. The work of [91] incorporates prior constraints, *e.g.*, objects are supported by floor and can not stick through the walls. While Lee *et al.* [5] incorporate volumetric constraints, *e.g.*, finite volume, spatial exclusion, containment, *etc.*. Fig. 5.2 shows a sketch of their method [5]. In the work of [6], Gupta *et al.* utilize a similar block model to facilitate the outdoor 3D scene understanding. Specifically, they incorporate global geometric constraints such as support force constraints, volume constraints and depth ordering constraints. An illustration of this method [6] is shown in Fig. 5.3. Likewise, these kinds of methods impose great restrictions on the scenarios to which they can be applied.

Non-parametric methods Non-parametric methods [7, 108] avoid explicitly defining a parametric model and require fewer assumptions as in traditional methods. The motivation behind these methods is that images with similar semantic scenes should have similar depth distributions when densely aligned. Therefore, the procedures of these approaches typically consist of building an image database for candidate retrieval, scene alignment and then depth transfer. Specifically, for a given RGBD image, they first find similar images from a pre-constructed image database, and then apply a warping algorithm, *i.e.*, SIFT flow, to align the input image with its retrieved candidates. Finally, an optimization procedure is used to interpolate and smooth the warped candidate depth values to obtain the inferred depth maps. Fig. 5.4 provides an illustration of the non-parametric method proposed in [7] for single image depth estimation. Benefits of these methods are that they scale well with respect to the training data size, and can be applied to depth estimation of general scene images. However, the three decoupled stages are prone to propagate errors. Furthermore, they rely on building a proper-sized image database and effective image retrieval algorithms.

Probabilistic methods Recently, supervised learning methods based on probabilistic graphical models have been investigated for tackling this problem, *e.g.*, [8, 89, 93, 109]. This kind of methods assume that an image is composed of homogeneous regions, *i.e.*, superpixels, with each region being a plane having similar properties. Depths of neighbouring regions should be closely related. Therefore probabilistic graphical models can

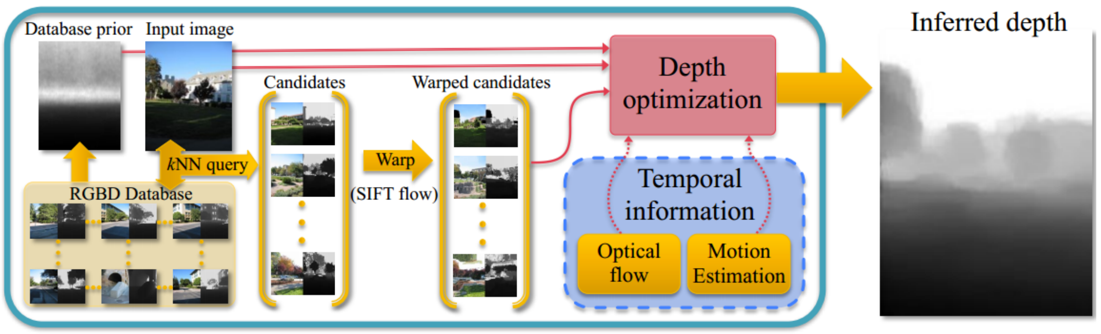


Figure 5.4: An illustration of the non-parametric methods for depth estimation. Figure reproduced from [7].

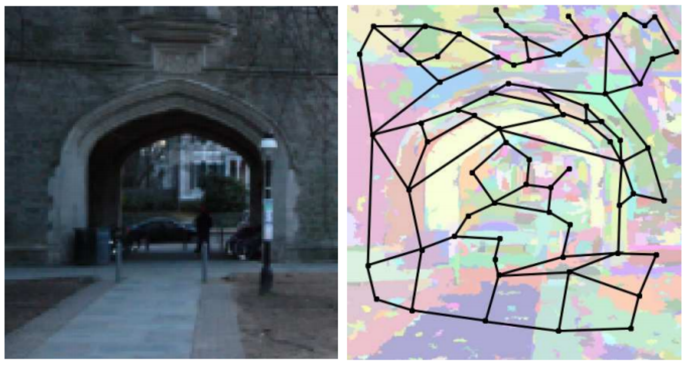


Figure 5.5: An illustration of the probabilistic model based methods for depth estimation. Left: input image; Right: superpixels overlaid with an MRF. Figure reproduced from [8].

be applied to capture these relationships. Specifically, most of these methods formulate the depth estimation as a Markov Random Field (MRF) learning problem. As exact MRF learning and inference are intractable in general, these approaches typically employ approximation methods, *e.g.*, multi-conditional learning (MCL) or particle belief propagation (PBP) for depth inference. Predicting the depths of a new image is inefficient, which takes around 4-5s in [8] and even longer (30s) in [93]. To make things worse, these methods suffer from lacking of flexibility in that [8, 109] rely on horizontal alignment of images. More recently, Liu *et al.* [10] propose a discrete-continuous CRF model to take into consideration the relations between adjacent superpixels, *e.g.*, occlusions. They also need to use approximation methods for learning and the maximum a posteriori (MAP) inference. Besides, their method relies on an image retrieval procedure to obtain a reasonable initialization. In contrast, here we present a deep continuous CRF model in which we can directly solve the log-likelihood optimization without any approximations, since the partition function can be analytically calculated. Predicting the depth of a new image is highly efficient since a closed form solution exists. Moreover, we do not inject any geometric priors nor any extra information.

Note that the above categorized methods are not completely isolated. Instead, they can be combined to yield a more robust depth estimation model. For instance, the geometric method in [91] and the non-parametric method in [7] also rely on probabilistic models for depth optimization; the MRF based probabilistic model presented in [8] also incorporates geometry constraints, *e.g.*, co-planarity and co-linearity.

Besides these methods, recent efforts have been focusing on incorporating additional information to assist the task of depth estimation, *e.g.*, user annotations [92], semantic labellings [89, 93]. In [89], Ladicky *et al.* have shown that jointly performing depth estimation and semantic labelling can benefit each other. Nevertheless, they all require additional efforts. To summarize, all of the previous methods [7, 8, 10, 89, 93] use hand-crafted features in their work, *e.g.*, texton, GIST, SIFT, PHOG, object bank, *etc.*. In contrast, we learn deep CNN for constructing the unary and pairwise potentials of a continuous CRF.

Most recently, Eigen *et al.* [9] proposed a multi-scale CNN approach for depth estimation, which bears similarity to our work here. However, our method differs critically from theirs: they use the CNN as a black-box by directly regressing the depth map from an input image through convolutions; in contrast we use a continuous CRF to explicitly model the relations of neighboring superpixels, and learn the potentials (both unary and binary) in a unified CNN framework.

5.2.2 Combining CNN and CRF

In [41], Farabet *et al.* propose a multi-scale CNN framework for scene labelling, which uses CRF as a post-processing step for local refinement. In the most recent work of [44], Tompson *et al.* present a hybrid architecture for jointly training a deep CNN and an MRF for human pose estimation. They first train a unary term and a spatial model separately, then jointly learn them as a fine tuning step. During fine tuning of the whole model, they simply remove the partition function in the likelihood to have a loose approximation. In contrast, our model performs continuous variable prediction. We can directly solve the log-likelihood optimization without using approximations as the partition function is integrable and can be analytically calculated. Moreover, during prediction, we have closed-form solutions to the MAP inference problem. Although no convolutional layers are involved, the work of [110] shares similarity with ours in that both continuous CRF's use neural networks to model the potentials. Note that the model in [110] is not deep and only one hidden is used, while ours is much deeper. It is unclear how the method of [110] performs on the challenging depth estimation problem that we consider here, which usually needs many convolutional layers to accommodate the complexity.

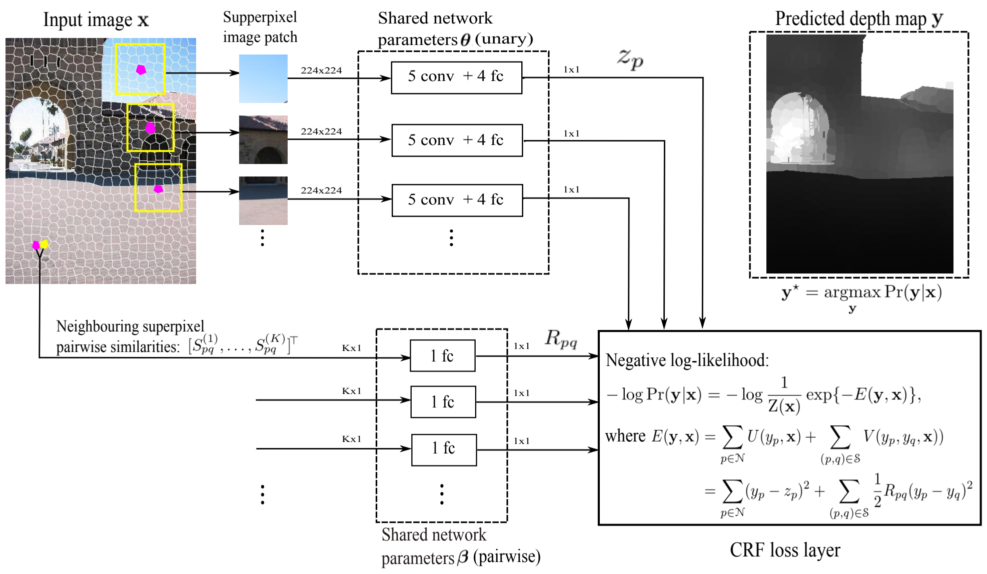

Input image $\mathbf{x}$
Supperpixel image patch
Shared network parameters $\boldsymbol{\theta}$ (unary)
224x224
5 conv + 4 fc
1x1
z_p
Predicted depth map $\mathbf{y}$
$\mathbf{y}^\star = \operatorname{argmax}_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x})$
Neighbouring superpixel pairwise similarities: $[S_{pq}^{(1)}, \ldots, S_{pq}^{(K)}]^\top$
Kx1
1 fc
R_{pq}
Shared network parameters $\boldsymbol{\beta}$ (pairwise)
Negative log-likelihood:
$-\log \Pr(\mathbf{y}|\mathbf{x}) = -\log \frac{1}{Z(\mathbf{x})} \exp\{-E(\mathbf{y},\mathbf{x})\}$,
where $E(\mathbf{y},\mathbf{x}) = \sum_{p \in \mathcal{N}} U(y_p,\mathbf{x}) + \sum_{(p,q) \in \mathcal{S}} V(y_p,y_q,\mathbf{x}))$
$= \sum_{p \in \mathcal{N}} (y_p - z_p)^2 + \sum_{(p,q) \in \mathcal{S}} \frac{1}{2} R_{pq}(y_p - y_q)^2$
CRF loss layer

Figure 5.6: An illustration of our DCNF model for depth estimation. The input image is first over-segmented into superpixels. In the unary part, for a superpixel p, we crop the image patch centred around its centroid, then resize and feed it to a CNN which is composed of 5 convolutional and 4 fully-connected layers (details refer to Fig. 5.7). In the pairwise part, for a pair of neighboring superpixels (p, q), we consider K types of similarities, and feed them into a fully-connected layer. The outputs of unary part and the pairwise part are then fed to the CRF structured loss layer, which minimizes the negative log-likelihood. Predicting the depths of a new image $\mathbf{x}$ is to maximize the conditional probability $\Pr(\mathbf{y}|\mathbf{x})$, which has closed-form solutions (see Sec. 5.3.3 for details).

5.2.3 Fully Convolutional Networks

Fully convolutional networks are recently been actively studied for dense prediction problems, *e.g.*, semantic segmentation [97, 98], image restoration [111], image super-resolution [112], depth estimations [9]. To deal with the downsampled output issue, interpolations are generally applied [9, 98]. In [113], Sermanet *et al.* propose an input shifting and output interlacing trick to produce dense predictions from coarse outputs without interpolations. Later on, Long *et al.* [97] present a deconvolution approach to put the upsampling into the training regime instead of applying it as a post-processing step. The CNN model presented in Eigen *et al.* [9] for depth estimations also suffers from this upsampling problem, *i.e.*, the predicted depth maps of [9] is 1/4-resolution of the original input image with some border areas lost. They simply use bilinear interpolations to upsample the predictions to the input image size. Unlike these existing methods, we propose a novel superpixel pooling method to address this issue. It jointly exploits the strengths of highly efficient fully convolutional networks and the benefits of superpixels at preserving object boundaries.

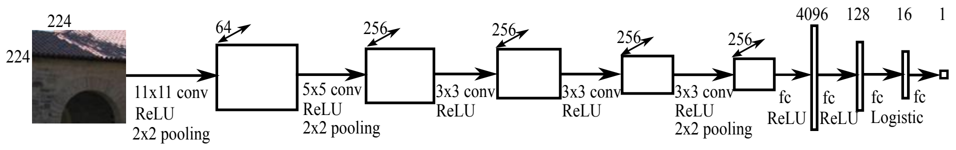


Figure 5.7: Detailed network architecture of the unary part in Fig. 5.6.

5.3 Deep Convolutional Neural Fields

We present the details of our deep convolutional neural field (DCNF) model for depth estimation in this section. Unless otherwise stated, we use boldfaced uppercase and lowercase letters to denote matrices and column vectors respectively.

5.3.1 Overview

The goal here is to infer the depth of each pixel in a single image depicting a general scene. Following the work of [8, 10, 93], we make the common assumption that an image is composed of small homogeneous regions (superpixels) and consider the graphical model composed of nodes defined on superpixels. Each superpixel is portrayed by the depth of its centroid. Let $\mathbf{x}$ be an image and $\mathbf{y} = [y_1, \dots, y_n]^\top \in \mathbb{R}^n$ be a vector of continuous depth values corresponding to all n superpixels in $\mathbf{x}$. Similar to conventional CRFs, we model the conditional probability distribution of the data with the following density function:

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{\mathrm{Z}(\mathbf{x})} \exp(-E(\mathbf{y}, \mathbf{x})), \tag{5.1}$$

where E is the energy function; Z is the partition function defined as:

$$\mathrm{Z}(\mathbf{x}) = \int_{\mathbf{y}} \exp\{-E(\mathbf{y}, \mathbf{x})\} \mathrm{d}\mathbf{y}. \tag{5.2}$$

Here, because $\mathbf{y}$ is continuous, the integral in Eq. (5.1) can be analytically calculated under certain circumstances, which we will show in Sec. 5.3.3. This is different from the discrete case, in which approximation methods need to be applied. To predict the depth of a new image, we solve the following MAP inference problem:

$$\mathbf{y}^\star = \operatorname*{argmax}_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}). \tag{5.3}$$

We formulate the energy function as a typical combination of unary potentials U and pairwise potentials V over the nodes (superpixels) $\mathcal{N}$ and edges $\mathcal{S}$ of the image $\mathbf{x}$:

$$E(\mathbf{y}, \mathbf{x}) = \sum_{p \in \mathcal{N}} U(y_p, \mathbf{x}) + \sum_{(p,q) \in \mathcal{S}} V(y_p, y_q, \mathbf{x}). \tag{5.4}$$

The unary term U aims to regress the depth value for a single superpixel. The pairwise term V encourages neighboring superpixels with similar appearances to take similar depths. We aim to jointly learn U and V in a unified CNN framework.

Fig. 5.6 sketches our deep convolutional neural field model for depth estimation. The whole network comprises a unary part, a pairwise part and a continuous CRF loss layer. For an input image, which has been over-segmented into n superpixels, we consider image patches centred around each superpixel centroid. The unary part then takes an image patch as input and feeds it to a CNN whose outputs is a single number, the regressed depth value of the superpixel.

The network for the unary part is composed of 5 convolutional and 4 fully-connected layers with details in Fig. 5.7. Note that the CNN parameters are shared across all the superpixels. The pairwise part takes similarity vectors (each with K components) of all neighboring superpixel pairs as input and feeds each of them to a fully-connected layer (parameters are shared among different pairs), then outputs a vector containing all the 1-dimensional similarities for each of the neighboring superpixel pairs. The continuous CRF loss layer takes the outputs from the unary and the pairwise terms to minimize the negative log-likelihood. Compared to the direct regression method in [9], our model possesses two potential advantages:

- We achieve translation invariance as we construct unary potentials irrespective of the superpixel's coordinate (shown in Sec. 5.3.2);
- We explicitly model the relations of neighboring superpixels through pairwise potentials.

In the following, we describe the details of potential functions involved in the energy function in Eq. (5.4).

5.3.2 Potential Functions

We present the details of the potential functions used in our model, which consists of a unary potential and a pairwise potential.

5.3.2.1 Unary potential

The unary potential is constructed from the output of a CNN by considering the least square loss:

$$U(y_p, \mathbf{x}; \boldsymbol{\theta}) = (y_p - z_p(\boldsymbol{\theta}))^2, \ \ \forall p = 1, ..., n. \tag{5.5}$$

Here z_p is the regressed depth of the superpixel p parametrized by the CNN parameters $\boldsymbol{\theta}$.

The network architecture for the unary part is depicted in Fig. 5.7. Our CNN model in Fig. 5.7 is mainly inspired by the well-known network architecture of Krizhevsky *et al.* [96] with modifications. It is composed of 5 convolutional layers and 4 fully-connected layers. The input image is first over-segmented into superpixels, then for each superpixel, we consider the image patch centred around its centroid. Each of the image patches is resized to 224×224 pixels (other resolutions also work) and then fed to the convolutional neural network. Note that the convolutional and the fully-connected layers are shared across all the image patches of different superpixels. Rectified linear units (ReLU) are used as activation functions for the five convolutional layers and the first two fully connected layers. For the third fully-connected layer, we use the logistic function $f(x) = (1 + e^{-x})^{-1}$ as the activation function. The last fully-connected layer plays the role of model ensemble with no activation function followed. The output is an 1-dimensional real-valued depth for a single superpixel.

5.3.2.2 Pairwise Potential

We construct the pairwise potential from K types of similarity observations, each of which enforces smoothness by exploiting consistency information of neighboring superpixels:

$$V(y_p, y_q, \mathbf{x}; \boldsymbol{\beta}) = \frac{1}{2} R_{pq}(y_p - y_q)^2, \; \forall p, q = 1, ..., n. \tag{5.6}$$

Here R_{pq} is the output of the network in the pairwise part (see Fig. 5.6) from a neighboring superpixel pair (p, q). We use a fully-connected layer here:

$$R_{pq} = \boldsymbol{\beta}^{\top} [S_{pq}^{(1)}, \ldots, S_{pq}^{(K)}]^{\top} = \sum_{k=1}^{K} \beta_k S_{pq}^{(k)}, \tag{5.7}$$

where $\mathbf{S}^{(k)}$ is the k-th similarity matrix whose elements are $S_{pq}^{(k)}$ ($\mathbf{S}^{(k)}$ is symmetric); $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_k]^{\top}$ are the network parameters. From Eq. (5.7), we can see that we do not use any activation function. However, as our framework is general, more complicated networks may be seamlessly incorporated for the pairwise part. In Sec. 5.3.3, we will show that we can derive a general form for calculating the gradients with respect to $\boldsymbol{\beta}$ (see Eq. (5.18)). To ensure that $\mathrm{Z}(x)$ in Eq. (5.2) is integrable, we require $\beta_k \geq 0$ as in [33]. Note that this is a sufficient but not necessary condition.

Here we consider 3 types of pairwise similarities, measured by the colour difference, colour histogram difference and texture disparity in terms of local binary patterns (LBP) [114], which take the conventional form:

$$S_{pq}^{(k)} = e^{-\gamma \left\| s_p^{(k)} - s_q^{(k)} \right\|}, \; k = 1, 2, 3;$$

where $s_p^{(k)}$, $s_q^{(k)}$ are the observation values of the superpixel p, q calculated from colour, colour histogram and LBP within the bounding box of the superpixel; $\|\cdot\|$ denotes the ℓ_2 norm of a vector and γ is a constant. It may be possible to learn features for the pairwise term too. For example, the pairwise term can be a deep CNN with raw pixels as the input. A more sophisticated pairwise energy may further improve the estimation, especially for complex discrete labelling problems, with the price of increased computation complexity. For depth estimation, we find that our current pairwise energy already works very well.

5.3.3 Learning

With the unary and the pairwise potentials defined in Eq. (5.5), (5.6), we can now write the energy function as:

$$E(\mathbf{y}, \mathbf{x}) = \sum_{p \in \mathcal{N}} (y_p - z_p)^2 + \sum_{(p,q) \in \mathcal{S}} \frac{1}{2} R_{pq} (y_p - y_q)^2. \tag{5.8}$$

For ease of expression, we introduce the following notation:

$$\mathbf{A} = \mathbf{I} + \mathbf{D} - \mathbf{R}, \tag{5.9}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix; $\mathbf{R}$ is the affinity matrix composed of R_{pq}; $\mathbf{D}$ is a diagonal matrix with $D_{pp} = \sum_q R_{pq}$. We see that $\mathbf{D} - \mathbf{R}$ is the graph Laplacian matrix. Thus $\mathbf{A}$ is the regularized Laplacian matrix. Expanding Eq. (5.8), we have:

$$\begin{aligned} E(\mathbf{y}, \mathbf{x}) &= \sum_p y_p^2 - 2 \sum_p y_p z_p + \sum_p z_p^2 + \frac{1}{2} \sum_{pq} R_{pq} y_p^2 - \sum_{pq} R_{pq} y_p y_q + \frac{1}{2} \sum_{pq} R_{pq} y_q^2 \\ &= \mathbf{y}^\top \mathbf{y} - 2\mathbf{z}^\top \mathbf{y} + \mathbf{z}^\top \mathbf{z} + \mathbf{y}^\top \mathbf{D} \mathbf{y} - \mathbf{y}^\top \mathbf{R} \mathbf{y} \\ &= \mathbf{y}^\top (\mathbf{I} + \mathbf{D} - \mathbf{R}) \mathbf{y} - 2\mathbf{z}^\top \mathbf{y} + \mathbf{z}^\top \mathbf{z} \\ &= \mathbf{y}^\top \mathbf{A} \mathbf{y} - 2\mathbf{z}^\top \mathbf{y} + \mathbf{z}^\top \mathbf{z}. \end{aligned} \tag{5.10}$$

Due to the quadratic terms of $\mathbf{y}$ in the energy function in Eq. (5.10) and the positive definiteness of $\mathbf{A}$ (with all positive edges of the graph, the Laplacian matrix must be positive semidefinite and therefore $\mathbf{A}$ must be positive definite), we can analytically calculate the integral in the partition function (Eq. (5.2)) as:

$$\begin{aligned}
\mathrm{Z}(\mathbf{x}) &= \int_{\mathbf{y}} \exp\Big\{ - E(\mathbf{y}, \mathbf{x}) \Big\} \mathrm{d}\mathbf{y} \\
&= \int_{\mathbf{y}} \exp\Big\{ - \mathbf{y}^\top \mathbf{A} \mathbf{y} + 2\mathbf{z}^\top \mathbf{y} - \mathbf{z}^\top \mathbf{z} \Big\} \mathrm{d}\mathbf{y} \\
&= \exp\{-\mathbf{z}^\top \mathbf{z}\} \int_{\mathbf{y}} \exp\Big\{ - \mathbf{y}^\top \mathbf{A} \mathbf{y} + 2\mathbf{z}^\top \mathbf{y} \Big\} \mathrm{d}\mathbf{y} \\
&= \exp\{-\mathbf{z}^\top \mathbf{z}\} \sqrt{\frac{(2\pi)^n}{|2\mathbf{A}|}} \exp\{\mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z}\} \\
&= \frac{(\pi)^{\frac{n}{2}}}{|\mathbf{A}|^{\frac{1}{2}}} \exp\{\mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z} - \mathbf{z}^\top \mathbf{z}\}, \qquad (5.11)
\end{aligned}$$

From Eqs. (5.1), (5.10), (5.11), we can now write the probability distribution function as:

$$\begin{aligned}
\Pr(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{y}, \mathbf{x})) \\
&= \frac{\exp\Big\{ - \mathbf{y}^\top \mathbf{A} \mathbf{y} + 2\mathbf{z}^\top \mathbf{y} - \mathbf{z}^\top \mathbf{z} \Big\}}{\frac{(\pi)^{\frac{n}{2}}}{|\mathbf{A}|^{\frac{1}{2}}} \exp\{\mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z} - \mathbf{z}^\top \mathbf{z}\}} \\
&= \frac{|\mathbf{A}|^{\frac{1}{2}}}{\pi^{\frac{n}{2}}} \exp\Big\{ - \mathbf{y}^\top \mathbf{A} \mathbf{y} + 2\mathbf{z}^\top \mathbf{y} - \mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z} \Big\}, \qquad (5.12)
\end{aligned}$$

where $\mathbf{z} = [z_1, \dots, z_n]^\top$; $| \cdot |$ denotes the determinant of a matrix, and $\mathbf{A}^{-1}$ the inverse of $\mathbf{A}$. Then the negative log-likelihood can be written as:

$$\begin{aligned}
- \log \Pr(\mathbf{y}|\mathbf{x}) = \mathbf{y}^\top \mathbf{A} \mathbf{y} - 2\mathbf{z}^\top \mathbf{y} + \mathbf{z}^\top \mathbf{A}^{-1} \mathbf{z} \qquad (5.13) \\
- \frac{1}{2} \log(|\mathbf{A}|) + \frac{n}{2} \log(\pi).
\end{aligned}$$

During learning, we minimize the negative conditional log-likelihood of the training data. Adding regularization to $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, we then arrive at the final optimization:

$$\begin{aligned}
\min_{\boldsymbol{\theta}, \boldsymbol{\beta} \geq \mathbf{0}} \frac{\lambda_1}{2} \|\boldsymbol{\theta}\|_2 + \frac{\lambda_2}{2} \|\boldsymbol{\beta}\|_2 \qquad (5.14) \\
- \sum_{i=1}^{N} \log \Pr(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}; \boldsymbol{\theta}, \boldsymbol{\beta}),
\end{aligned}$$

where $\mathbf{x}^{(i)}$, $\mathbf{y}^{(i)}$ denote the i-th training image and the corresponding depth map; N is the number of training images; λ_1 and λ_2 are weight decay parameters.

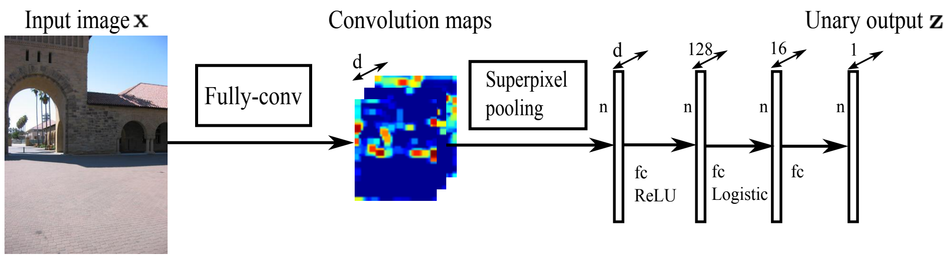


Figure 5.8: An overview of the unary part of the DCNF-FCSP model. For the unary part, the input image is fed into a fully-convolutional network to produce convolution maps (d is the number of filters of the last fully-convolutional layer). The obtained convolution maps, together with the superpixel segmentation over the original input image, are fed to a superpixel pooling layer. The outputs are $n \times 1$ d dimensional feature vectors for each of the n superpixels, which are then followed by 3 fully-connected layers to produce the unary output $\mathbf{z}$. The pairwise part are omitted here since we use the same network architecture as in the DCNF model (Fig. 5.6). The unary output $\mathbf{z}$ and the pairwise output $\mathbf{R}$ are used as input to the CRF loss layer, which minimizes the negative log-likelihood (See Sec. 5.3.4 for details) .

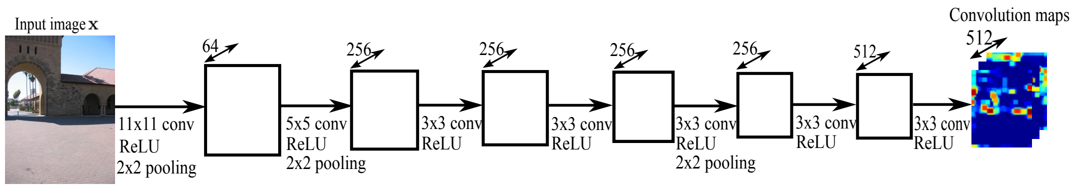


Figure 5.9: The fully convolutional network architecture used in Fig. 5.8. The network takes input images of arbitrary size and output convolution maps.

5.3.3.1 Optimization

We use stochastic gradient descent (SGD) based back propagation to solve the optimization problem in Eq. (5.14) for learning all parameters of the whole network. We project the solutions to the feasible set when the bounded constraints $\beta_k \geq 0$ is violated. In the following, we calculate the partial derivatives of $-\log \Pr(\mathbf{y}|\mathbf{x})$ with respect to the network parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$.

For the unary part, here we calculate the partial derivatives of $-\log \Pr(\mathbf{y}|\mathbf{x})$ with respect to θ_l (one element of $\boldsymbol{\theta}$). Recall that $\mathbf{A} = \mathbf{I}+\mathbf{D}-\mathbf{R}$ (Eq. (5.9)); $\mathbf{A}^\top = \mathbf{A}$; $(\mathbf{A}^{-1})^\top = \mathbf{A}^{-1}$;

$|\mathbf{A}^{-1}| = \frac{1}{|\mathbf{A}|}$, we have:

$$\begin{aligned}
&\frac{\partial\{-\log \Pr(\mathbf{y}|\mathbf{x})\}}{\partial \theta_l} \\
&= \frac{\partial\{-2\mathbf{z}^\top \mathbf{y} + \mathbf{z}^\top \mathbf{A}^{-1}\mathbf{z}\}}{\partial \theta_l} \\
&= \frac{\partial\{-2\mathbf{z}^\top \mathbf{y}\}}{\partial \theta_l} + \frac{\partial\{\mathbf{z}^\top \mathbf{A}^{-1}\mathbf{z}\}}{\partial \theta_l} \\
&= -2\frac{\partial\{\sum_p z_p y_p\}}{\partial \theta_l} + \frac{\partial\{\sum_{pq} z_p z_q A^{-1}_{pq}\}}{\partial \theta_l} \\
&= -2\sum_p \left(y_p \frac{\partial z_p}{\partial \theta_l}\right) + \sum_{pq}\left(z_p \frac{\partial z_q}{\partial \theta_l} + z_q \frac{\partial z_p}{\partial \theta_l}\right) A^{-1}_{pq} \\
&= -2\mathbf{y}^\top \frac{\partial \mathbf{z}}{\partial \theta_l} + 2\mathbf{z}^\top \mathbf{A}^{-1}\frac{\partial \mathbf{z}}{\partial \theta_l} \\
&= 2(\mathbf{A}^{-1}\mathbf{z} - \mathbf{y})^\top \frac{\partial \mathbf{z}}{\partial \theta_l}.
\end{aligned} \tag{5.15}$$

For the pairwise part, we calculate the partial derivatives of $-\log \Pr(\mathbf{y}|\mathbf{x})$ with respect to β_k as:

$$\begin{aligned}
&\frac{\partial\{-\log \Pr(\mathbf{y}|\mathbf{x})\}}{\partial \beta_k} \\
&= \frac{\partial\{\mathbf{y}^\top \mathbf{A}\mathbf{y} + \mathbf{z}^\top \mathbf{A}^{-1}\mathbf{z} - \frac{1}{2}\log(|\mathbf{A}|)\}}{\partial \beta_k} \\
&= \frac{\partial\{\mathbf{y}^\top \mathbf{A}\mathbf{y}\}}{\partial \beta_k} + \frac{\partial\{\mathbf{z}^\top \mathbf{A}^{-1}\mathbf{z}\}}{\partial \beta_k} - \frac{1}{2}\frac{\partial \log(|\mathbf{A}|)}{\partial \beta_k}, \\
&= \mathbf{y}^\top \frac{\partial \mathbf{A}}{\partial \beta_k}\mathbf{y} - \mathbf{z}^\top \mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \beta_k}\mathbf{A}^{-1}\mathbf{z} - \frac{1}{2}\frac{1}{|\mathbf{A}|}\frac{\partial\{|\mathbf{A}|\}}{\partial \beta_k}, \\
&= \mathbf{y}^\top \frac{\partial \mathbf{A}}{\partial \beta_k}\mathbf{y} - \mathbf{z}^\top \mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \beta_k}\mathbf{A}^{-1}\mathbf{z} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \beta_k}\right).
\end{aligned} \tag{5.16}$$

where $\operatorname{Tr}(\cdot)$ denotes the trace of a matrix. We here introduce matrix $\mathbf{J}$ to denote $\frac{\partial \mathbf{A}}{\partial \beta_k}$. Each element of $\mathbf{J}$ is:

$$\begin{aligned}
J_{pq} &= \frac{\partial A_{pq}}{\partial \beta_k} \\
&= \frac{\partial\{D_{pq} - R_{pq}\}}{\partial \beta_k} \\
&= \frac{\partial D_{pq}}{\partial \beta_k} - \frac{\partial R_{pq}}{\partial \beta_k} \\
&= -\frac{\partial R_{pq}}{\partial \beta_k} + \delta(p = q)\sum_q \frac{\partial R_{pq}}{\partial \beta_k},
\end{aligned} \tag{5.17}$$

where $\delta(\cdot)$ is the indicator function, which equals 1 if $p = q$ is true and 0 otherwise. According to Eq. (5.16) and the definition of $\mathbf{J}$ in (5.17), we can now write the partial derivative of $-\log \Pr(\mathbf{y}|\mathbf{x})$ with respect to β_k as:

$$\frac{\partial\{-\log \Pr(\mathbf{y}|\mathbf{x})\}}{\partial \beta_k} = \mathbf{y}^\top \mathbf{J}\mathbf{y} - \mathbf{z}^\top \mathbf{A}^{-1}\mathbf{J}\mathbf{A}^{-1}\mathbf{z} - \frac{1}{2}\operatorname{Tr}\left(\mathbf{A}^{-1}\mathbf{J}\right). \tag{5.18}$$

From Eqs. (5.18), (5.17), we can see that our framework is general and more complicated networks for the pairwise part can be seamlessly incorporated. Here, in our case, with the definition of R_{pq} in Eq. (5.7), we have $\frac{\partial R_{pq}}{\partial \beta_k} = S_{pq}^{(k)}$.

5.3.3.2 Depth Prediction

Predicting the depths of a new image is to solve the MAP inference in Eq. (5.3), which writes as:

$$\begin{aligned} \mathbf{y}^\star &= \operatorname*{argmax}_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) \\ &= \operatorname*{argmax}_{\mathbf{y}} \log \Pr(\mathbf{y}|\mathbf{x}) \\ &= \operatorname*{argmax}_{\mathbf{y}} -\mathbf{y}^\top \mathbf{A}\mathbf{y} + 2\mathbf{z}^\top \mathbf{y}. \end{aligned} \tag{5.19}$$

With the definition of $\mathbf{A}$ in Eq. (5.9), $\mathbf{A}$ is symmetric. Then by setting the partial derivative of $-\mathbf{y}^\top \mathbf{A}\mathbf{y} + 2\mathbf{z}^\top \mathbf{y}$ with respect to $\mathbf{y}$ to $\mathbf{0}$, we have

$$\begin{aligned} &\frac{\partial\{-\mathbf{y}^\top \mathbf{A}\mathbf{y} + 2\mathbf{z}^\top \mathbf{y}\}}{\partial \mathbf{y}} = \mathbf{0} \\ \Rightarrow\ & -(\mathbf{A} + \mathbf{A}^\top)\mathbf{y} + 2\mathbf{z} = \mathbf{0} \\ \Rightarrow\ & -2\mathbf{A}\mathbf{y} + 2\mathbf{z} = \mathbf{0} \\ \Rightarrow\ & \mathbf{y} = \mathbf{A}^{-1}\mathbf{z}. \end{aligned} \tag{5.20}$$

It shows that closed-form solutions exist for the MAP inference in Eq. (5.19):

$$\mathbf{y}^\star = \mathbf{A}^{-1}\mathbf{z} \tag{5.21}$$

If we discard the pairwise terms, namely $R_{pq} = 0$, then Eq. (5.21) degenerates to $\mathbf{y}^\star = \mathbf{z}$, which is a plain CNN regression model (we will report the results of this method as a baseline in the experiment). Note that we do not need to explicitly compute the matrix inverse $\mathbf{A}^{-1}$ which can be expensive (cubic in the number of nodes). Instead we can obtain the value of $\mathbf{A}^{-1}\mathbf{z}$ by solving a linear system.

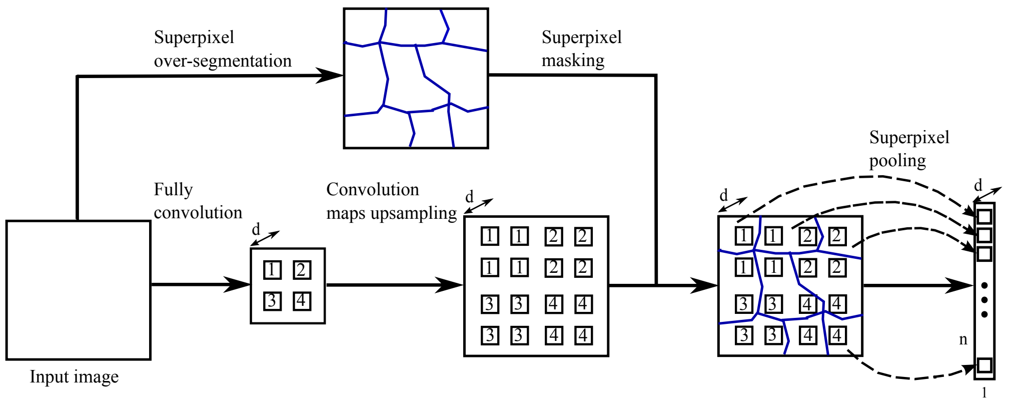


Figure 5.10: An illustration of the superpixel pooling method, which mainly consists of convolution maps upsampling and superpixel pooling. The convolution maps are upsampled to the original image size by nearest neighbor interpolations, over which the superpixel masking is applied. Then average pooling is performed within each superpixel region, to produce the n convolution features. n is the number of superpixels in the image. d is the number of channels of the convolution maps.

5.3.4 Speeding up Training Using Fully Convolutional Networks and Superpixel Pooling

Thus far, we have presented our DCNF model for depth estimations based on image superpixels. From Fig. 5.6, we can see that for constructing the unary potentials, we are essentially performing patchwise convolutions (similar operations are performed in the R-CNN [100]). A major concern of the proposed method is its computational efficiency and memory consumption, since we need to perform convolutions over hundreds or even thousands (number of superpixels) of image patches for a single input image. Many of those convolutions are redundant due to significant image patch overlaps.

Naturally, a promising direction for reducing the computation burden is to perform convolutions over the entire image once, and then obtain convolutional features for each superpixel. However, to find the convolutional features of the image superpixels from the obtained convolution maps, one needs to establish associations between these two. Therefore, we here propose an improved model, which we term as DCNF-FCSP, based on fully convolutional networks and a novel superpixel pooling method, to address this issue. As we will show in Sec. 5.4.2, this new model significantly speeds up the training and prediction while producing almost the same prediction accuracy. Most importantly, with this more efficient model, we are able to design deeper networks to achieve better performance.

5.3.4.1 DCNF-FCSP Overview

In comparison to the DCNF model, our new DCNF-FCSP model mainly improves the unary part while keeping the same pairwise network architecture as in Fig. 5.6. We show the model architecture of the unary part in Fig. 5.8. Specifically, the input image is fed to a fully convolutional network (introduced in the sequel). The outputs are convolutional feature maps of size $h \times w \times d$ (d is the dimension of the obtained convolutional feature vector, *i.e.*, the number of channels of the last convolutional layer). The size of the convolution maps $h \times w$ are typically smaller than the input image size. Each convolutional feature vector in the output convolution maps corresponds to a patch in the input image. We propose a novel superpixel pooling method (described in the following) to associate these outputs back to the superpixels in the input image. Namely, the convolutional feature maps are used as inputs to a superpixel pooling layer to obtain n superpixel feature vectors with d dimensions (n is the number of superpixels). The n superpixel feature vectors are then fed to 3 fully-connected layers to produce the unary output $\mathbf{z}$. We use the same pairwise network architecture as depicted in Fig. 5.6, which we do not show here. With the unary output $\mathbf{z}$ and the pairwise output $\mathbf{R}$, we construct potential functions according to Eqs. (5.5) and (5.6) and optimize the negative log-likelihood.

Compared to the original proposed DCNF method, this improved DCNF-FCSP model only needs to perform convolutions over the entire image once, rather than hundreds of superpixel image patches. This significantly reduces the computation and GPU memory burden, bringing around 10 times training speedup while producing almost the same prediction accuracy, as we demonstrate later in Sec. 5.4.2. We next introduce the fully convolutional networks and the superpixel pooling method in detail.

5.3.4.2 Fully Convolutional Networks

Typical CNN models, including AlexNet [96], vggNet [99, 115], *etc.*, are composed of convolution layers and fully connected layers. They usually take standard sized images as inputs, *e.g.*, 224×224 pixels, to produce nonspatial outputs. In contrast, a fully convolutional network can take as inputs arbitrarily sized images, and outputs convolutional spatial maps. It has therefore been actively studied for dense prediction problems [9, 97, 98, 112] in very recently. We here exploit this new development trend in CNN to speedup the patch-wise convolutions in the DCNF model. We illustrate the fully convolutional network architecture that we use in Fig. 5.9. As shown, the network is composed of 7 convolution layers, with the first 5 layers transferred from the AlexNet [96]. We then add 2 more convolution layers with 3×3 filter size and 512 channels each.

The network takes as input images of arbitrary size and outputs convolution maps of channel $d = 512$. Note that we can design deeper networks here for pursuing better performance. We will demonstrate in Sec. 5.4.3 the benefits of using deeper models.

5.3.4.3 Superpixel Pooling

After the input images go through the fully convolutional networks, we acquire convolution maps. To obtain superpixel features, we need to associate these convolutional feature maps back to the image superpixels. Thus we here propose a novel superpixel pooling method. An illustration of this method is shown in Fig. 5.10, which mainly consists of the convolution maps upsampling and superpixel average pooling. In general, this superpixel pooling layer takes the convolution maps as input and output superpixel features. Specifically, the obtained convolution maps are first upsampled to the original image size by nearest neighbor interpolation. Note that other interpolation methods such as linear interpolation may be applicable too. However, as discussed below, nearest neighbor interpolation makes the implementation much easier and computation faster.

Then the superpixel masking is applied and average pooling is performed within each superpixel region. The outputs are pooled superpixel features, which are used for constructing the unary potentials. In the sequel, we describe this method in detail.

In practice there is no need to explicitly upsample the convolution maps. Instead, we count the frequencies of the convolutional feature vectors that fall into each superpixel region. We denote the convolution maps as $\mathbf{C} \in \mathbb{R}^{h \times w \times d}$, with each element being C_{ijk} $(i = 1, \ldots, h; j = 1, \ldots, w; k = 1, \ldots, d)$. We represent the t-th superpixel feature as a d-dimensional column vector $\mathbf{h}_t$ $(t = 1, \ldots, n)$, with elements h_{tk} $(k = 1, \ldots, d)$. $\mathbf{W}_t \in \mathbb{R}^{h \times w}$ is a frequency weighting matrix associated to the t-th superpixel, with elements being W_{ijt}. W_{ijt} represents the weight of (i, j)-th feature vector in the convolution maps that associated to the t-th superpixel. To calculate W_{ijt}, we simply count the occurrences of the (i, j)-th convolutional feature vector that appear in the t-th superpixel region, and do L_1 normalization for each $\mathbf{W}_t$. By constructing this frequency matrix $\mathbf{W}_t$, we avoid the explicit upsampling operation. Then the superpixel pooling can be represented as:

$$h_{tk} = \sum_{(i,j) \in \mathcal{R}_t} W_{ijt} \cdot C_{ijk}, \tag{5.22}$$

where $(i, j) \in \mathcal{R}_t$ denotes the (i, j)-th convolutional feature vector in the convolution maps being associated to the t-th superpixel.

During the network forward pass, the superpixel pooling layer performs a linear transformation in Eq. 5.22 to output $\mathbf{h}_t$ from the input $\mathbf{C}$. For the network backward, the gradients can be easily calculated since we have:

$$\frac{\partial h_{tk}}{\partial C_{ijk}} = \begin{cases} W_{ijt} & \text{if } (i,j) \in \mathcal{R}_t \\ 0 & \text{otherwise.} \end{cases} \tag{5.23}$$

Thus far, we have successfully established the associations between the convolutional feature maps and the image superpixels. It should be noted that although simple as it is, the proposed superpixel pooling method jointly exploits the benefits of fully convolutional networks and superpixels. It provides an efficient yet equally effective approach to the patchwise convolutions used in the DCNF model, as we demonstrate in Sec. 5.4.2.

5.3.5 Implementation Details

We implement the network training based on the CNN toolbox: VLFeat MatConvNet[1] with our own modifications. Training is done on a standard desktop with an NVIDIA GTX 780 GPU with 6GB memory. We present the implementation details of the two proposed models in the following.

DCNF We initialize the first 6 layers of the unary part in Fig. 5.7 using a CNN model trained on the ImageNet from [115]. First, we do not back propagate through the previous 6 layers by keeping them fixed and train the rest of the network (we refer this process as pre-train) with the following settings: momentum is set to 0.9, and weight decay parameters λ_1, λ_2 are set to 0.0005. During pre-train, the learning rate is initialized at 0.0001, and decreased by 40% every 20 epoches. We then run 60 epoches to report the results of pre-train (with learning rate decreased twice). During pre-training, it takes less than 0.1s for one network forward pass to do depth predictions. Then we train the whole network with the same momentum and weight decay. We apply dropout with ratio 0.5 in the first two fully-connected layers of Fig. 5.7. Training the whole network takes around 16.5 hours on the Make3D dataset, and around 33 hours on the NYU v2 dataset.

DCNF-FCSP We initialize the first 5 layers in Fig. 5.9 with the same model trained on the ImageNet from [115]. The momentum and weight decay parameters are set the same as in the DCNF model. We also use the same training protocol as in the DCNF model, *i.e.*, first pre-train and then fine-tune the whole model.

[1]VLFeat MatConvNet: http://www.vlfeat.org/matconvnet/

Table 5.1: Baseline comparisons on the NYU v2 dataset. Our method with the whole network training performs the best.

Method	Error (lower is better)			Accuracy (higher is better)		
	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
SVR	0.313	0.128	1.068	0.490	0.787	0.921
SVR (smooth)	0.290	0.116	0.993	0.514	0.821	0.943
Unary only	0.295	0.117	0.985	0.516	0.815	0.938
Unary only (smooth)	0.287	0.112	0.956	0.535	0.828	0.943
DCNF (pre-train)	0.257	0.101	0.843	0.588	0.868	0.961
DCNF (fine-tune)	**0.230**	**0.095**	**0.824**	**0.614**	**0.883**	**0.971**

Table 5.2: Baseline comparisons on the Make3D dataset. Our method with the whole network training performs the best.

Method	Error (C1) (lower is better)			Error (C2) (lower is better)		
	rel	log10	rms	rel	log10	rms
SVR	0.433	0.158	8.93	0.429	0.170	15.29
SVR (smooth)	0.380	0.140	**8.12**	0.384	0.155	15.10
Unary only	0.366	0.137	8.63	0.363	0.148	14.41
Unary only (smooth)	0.341	0.131	8.49	0.349	0.144	14.37
DCNF (pre-train)	0.331	0.127	8.82	0.324	0.134	13.29
DCNF (fine-tune)	**0.314**	**0.119**	8.60	**0.307**	**0.125**	**12.89**

Table 5.3: Performance comparisons of DCNF and DCNF-FCSP on the NYU v2 dataset. The two models show comparable performance.

Method	Error (lower is better)			Accuracy (higher is better)		
	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
DCNF (pre-train)	0.257	0.101	0.843	0.588	0.868	0.961
DCNF (fine-tune)	0.230	0.095	0.824	**0.614**	0.883	**0.971**
DCNF-FCSP (pre-train)	0.261	0.100	0.842	0.583	0.869	0.964
DCNF-FCSP (fine-tune)	0.237	**0.082**	**0.822**	0.608	**0.889**	0.969

Table 5.4: Performance comparisons of DCNF and DCNF-FCSP on the Make3D dataset. The two models perform on par in general.

Method	Error (C1) (lower is better)			Error (C2) (lower is better)		
	rel	log10	rms	rel	log10	rms
DCNF (pre-train)	0.331	0.127	8.82	0.324	0.134	13.29
DCNF (fine-tune)	0.314	0.119	**8.60**	0.307	0.125	**12.89**
DCNF-FCSP (pre-train)	0.323	0.127	9.01	0.318	0.136	13.89
DCNF-FCSP (fine-tune)	**0.312**	**0.113**	9.10	**0.305**	**0.120**	13.24

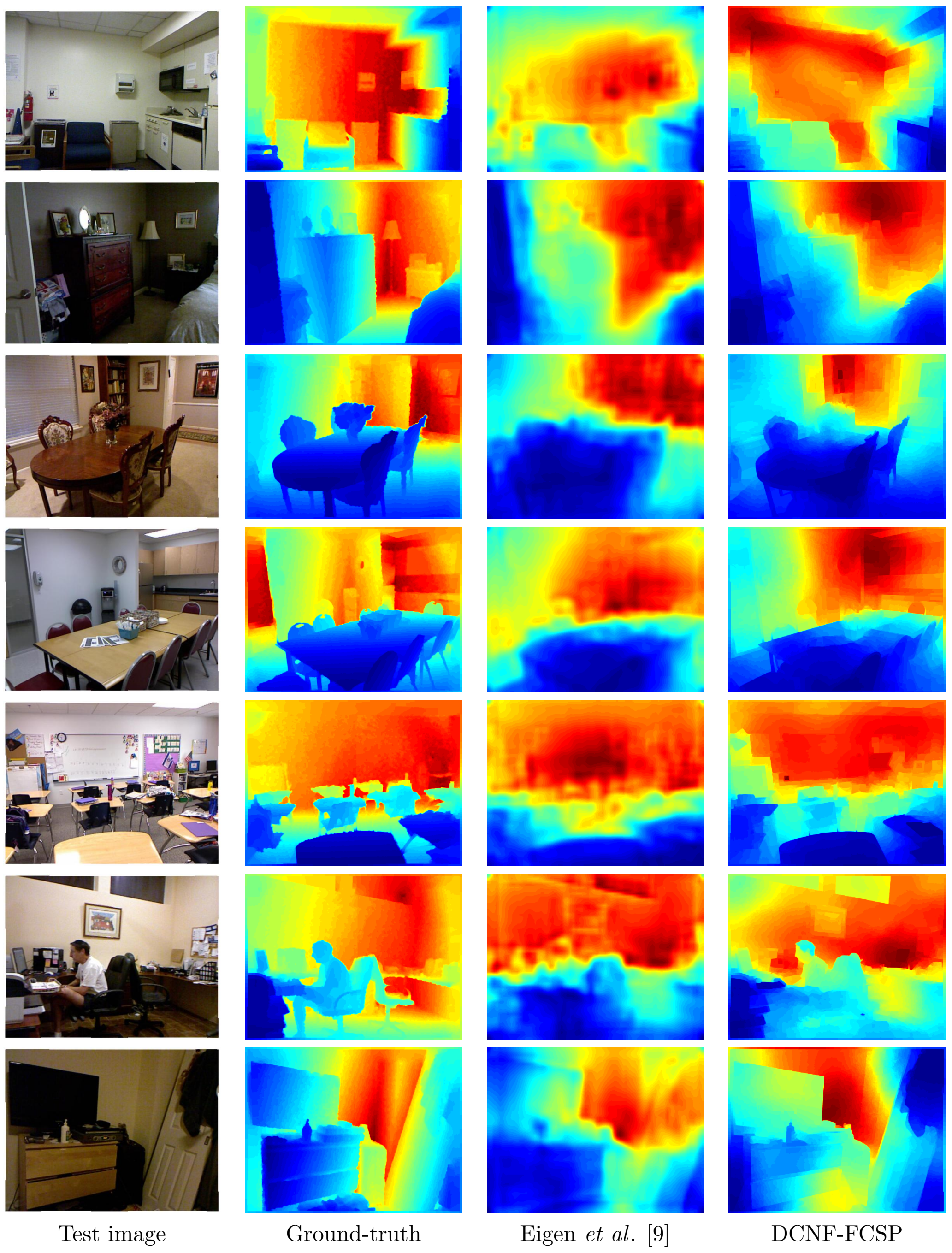


Figure 5.11: Examples of qualitative comparisons on the NYUD2 dataset (Best viewed on screen). Color indicates depths (red is far, blue is close). Our method yields visually better predictions with sharper transitions, aligning to local details.

5.4 Experiments

We organize our experiments into the following three parts: 1) We compare our DCNF model with several baseline methods to show the benefits of jointly learning CNN and CRF; 2) We perform comparisons between the two proposed models, *i.e.*, DCNF and DCNF-FCSP, to show that the DCNF-FCSP model is equally effective while generally

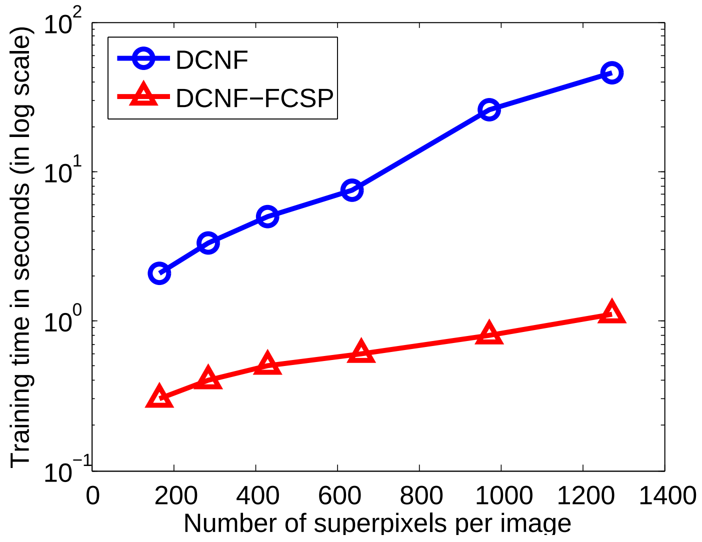


Figure 5.12: Comparison of the whole model training time (network forward + backward) in seconds (in *log* scale) for one image on the NYU v2 dataset with respect to different numbers of superpixels per image. The DCNF-FCSP model is orders of magnitude faster than the DCNF model.

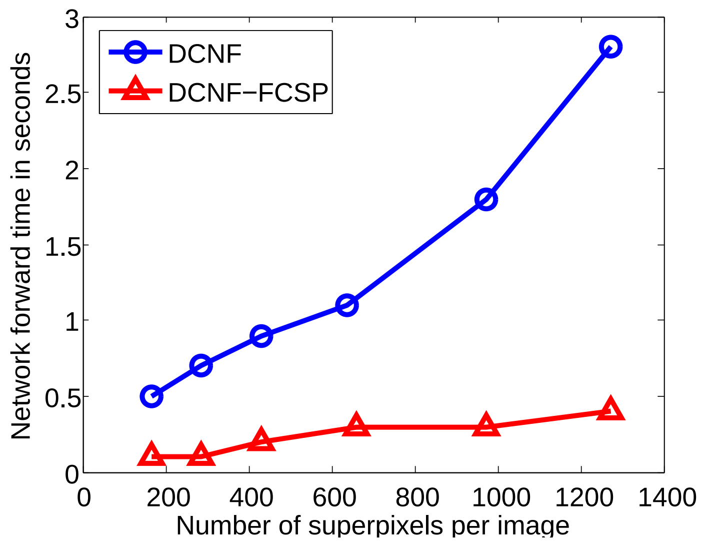


Figure 5.13: Comparison of the network forward time of the whole model during depth prediction (in seconds) for one image on the NYU v2 dataset with respect to different numbers of superpixels per image. The DCNF-FCSP model is significantly faster than the DCNF model.

being $\sim$ 10 times faster; 3) We compare our DCNF-FCSP model using deeper network design with state-of-the-art methods to show that our model performs significantly better. We evaluate on two popular datasets which are available online: the indoor NYU v2 Kinect dataset [116] and the outdoor Make3D range image dataset [8]. Several measures commonly used in prior works are applied here for quantitative evaluations:

- average relative error (rel): $\frac{1}{T}\sum_p \frac{|d_p^{gt} - d_p|}{d_p^{gt}}$;
- root mean squared error (rms): $\sqrt{\frac{1}{T}\sum_p (d_p^{gt} - d_p)^2}$;

Table 5.5: State-of-the-art comparisons on the NYU v2 dataset. Our method performs the best in most cases. Note that the results of Eigen *et al.* [9] are obtained by using extra training data (in the millions in total) while ours are obtained using the standard training set.

Method	Error (lower is better)			Accuracy (higher is better)		
	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Make3d [8]	0.349	-	1.214	0.447	0.745	0.897
DepthTransfer [7]	0.35	0.131	1.2	-	-	-
Discrete-continuous CRF [10]	0.335	0.127	1.06	-	-	-
Ladicky *et al.* [89]	-	-	-	0.542	0.829	0.941
Eigen *et al.* [9]	0.215	-	0.907	0.611	0.887	0.971
DCNF-FCSP (pre-train)	0.234	0.095	0.842	0.604	0.885	0.973
DCNF-FCSP (fine-tune)	**0.213**	**0.087**	**0.759**	**0.650**	**0.906**	**0.976**

Table 5.6: State-of-the-art comparisons on the Make3D dataset. Our method performs the best. Note that the C2 errors of the Discrete-continuous CRF [10] are reported with an ad-hoc post-processing step (train a classifier to label sky pixels and set the corresponding regions to the maximum depth).

Method	Error (C1) (lower is better)			Error (C2) (lower is better)		
	rel	log10	rms	rel	log10	rms
Make3d [8]	-	-	-	0.370	0.187	-
Semantic Labelling [93]	-	-	-	0.379	0.148	-
DepthTransfer [7]	0.355	0.127	9.20	0.361	0.148	15.10
Discrete-continuous CRF [10]	0.335	0.137	9.49	0.338	0.134	**12.60**
DCNF-FCSP (pre-train)	0.331	0.119	7.77	0.330	0.133	14.46
DCNF-FCSP (fine-tune)	**0.287**	**0.109**	**7.36**	**0.287**	**0.122**	14.09

Table 5.7: State-of-the-art comparisons on the KITTI dataset. Our method achieves the best RMS error. Note that the results of Eigen *et al.* [9] are obtained by using extra training data (in the millions in total) while ours are obtained using 700 training images. The results of Saxena *et al.* [8] are reproduced from [9]

Method	Error (lower is better)			Accuracy (higher is better)		
	rel	log10	rms	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Saxena *et al.* [8]	0.280	-	8.734	0.601	0.820	0.926
Eigen *et al.* [9]	**0.190**	-	7.156	**0.692**	**0.899**	**0.967**
DCNF-FCSP (pre-train)	0.236	0.101	7.421	0.613	0.858	0.949
DCNF-FCSP (fine-tune)	0.217	**0.092**	**7.046**	0.656	0.881	0.958

- average $\log_{10}$ error (log10):
  $\frac{1}{T}\sum_p |\log_{10} d_p^{gt} - \log_{10} d_p|$;
- accuracy with threshold *thr*:
  percentage (%) of d_p s.t.: $\max(\frac{d_p^{gt}}{d_p}, \frac{d_p}{d_p^{gt}}) = \delta < thr$;

where d_p^{gt} and d_p are the ground-truth and predicted depths respectively at pixel indexed by p, and T is the total number of pixels in all the evaluated images.

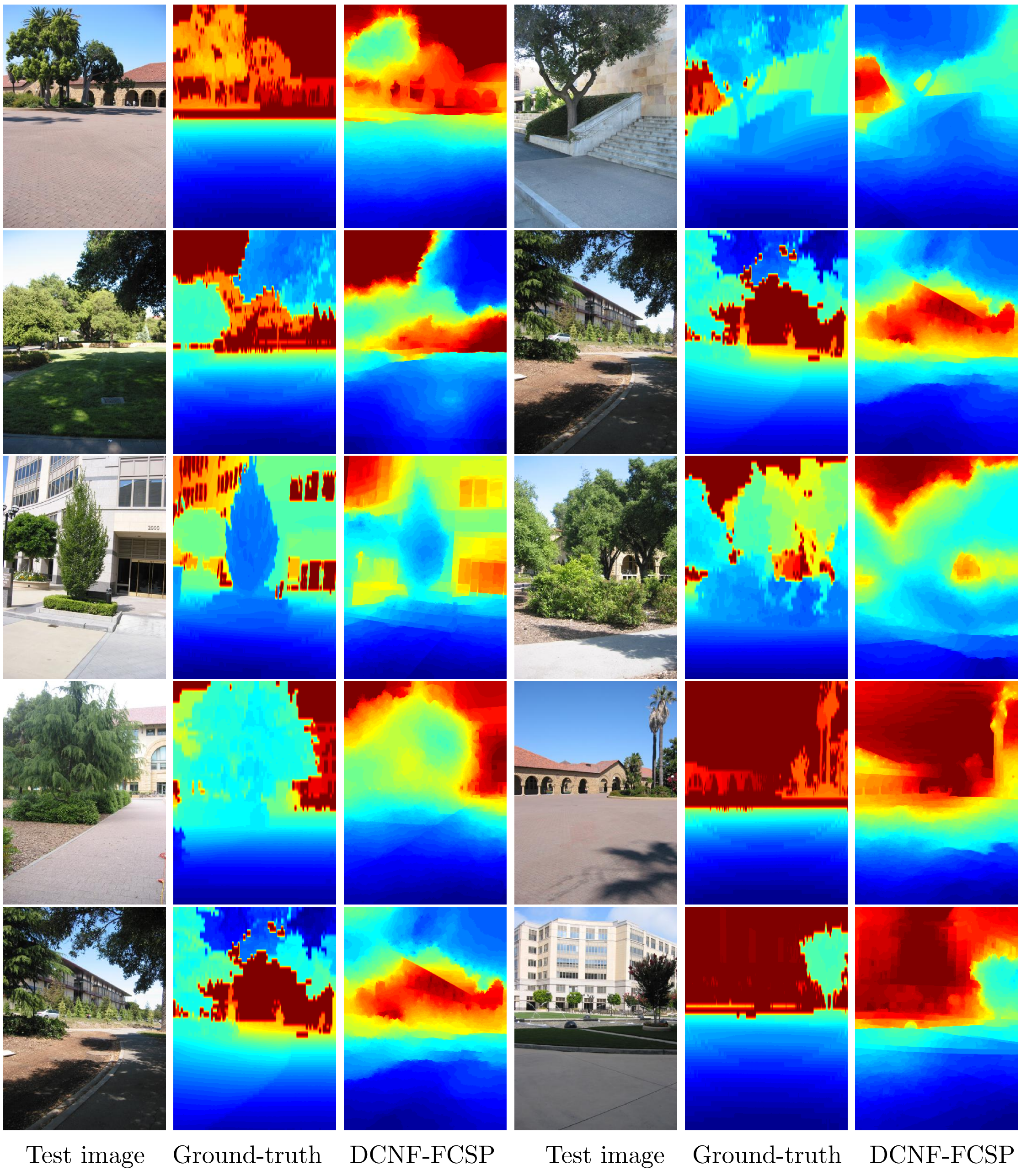


Figure 5.14: Examples of depth predictions on the Make3D dataset (Best viewed on screen). Depths are shown in log scale and in color (red is far, blue is close).

We use SLIC [117] to segment the images into a set of non-overlapping superpixels. For the DCNF model, we consider the image patch within a rectangular box centred on the centroid of each of the superpixels, which contains a large portion of its background surroundings. More specifically, we use a box size of 168×168 pixels for the NYU v2 and 120×120 pixels for the Make3D dataset. Following [8, 9, 93], we transform the depths into log-scale before training. For better visualizations, we apply a cross-bilateral filter [118] for inpainting using the provided toolbox [116] after obtaining the superpixel depth predictions. Our experiments empirically show that this post-processing has negligibly

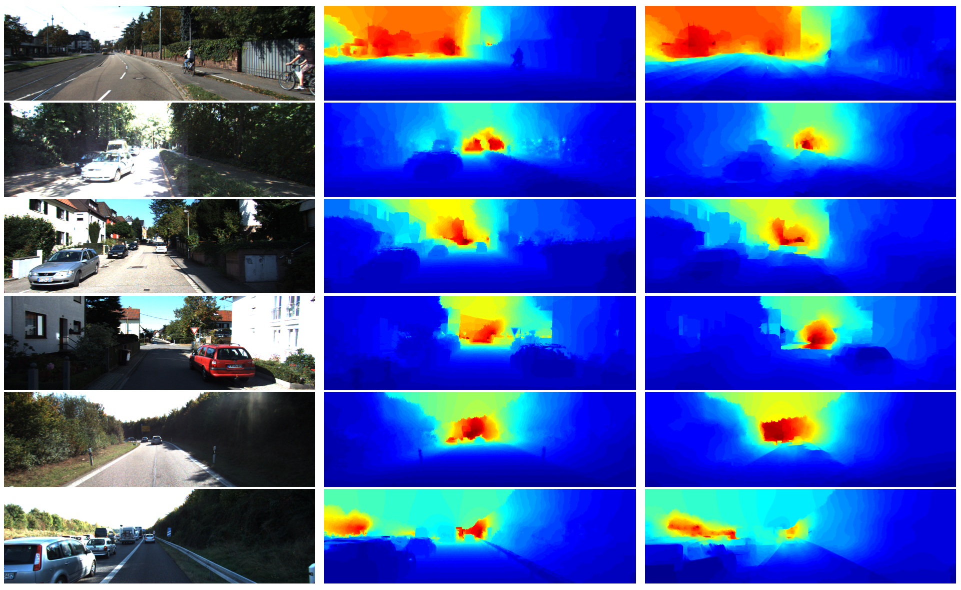


Figure 5.15: Examples of depth predictions on the KITTI dataset (Best viewed on screen). Depths are shown in log scale and in color (red is far, blue is close).

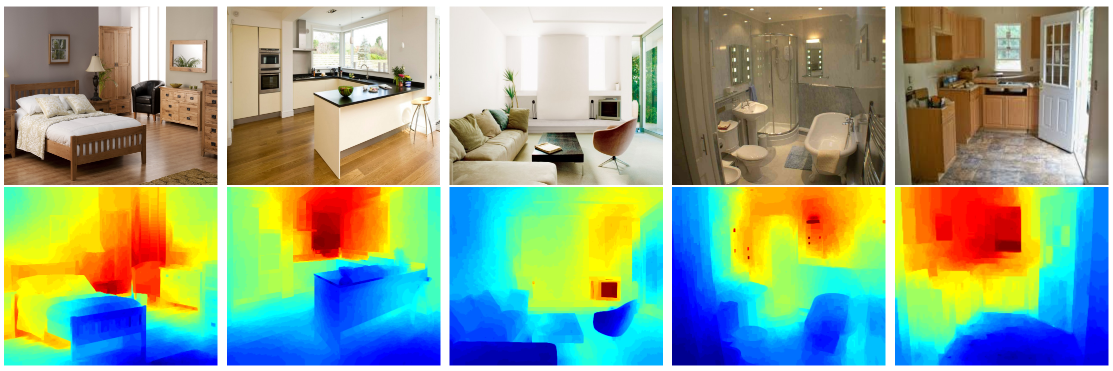


Figure 5.16: Examples of depth predictions on general indoor scene images obtained from the Internet (First row: test images; second row: our depth predictions. Best viewed on screen). Depths are shown in log scale and in color (red indicates far and blue indicates close).

impact on the evaluation performance.

5.4.1 Baseline Comparisons

To demonstrate the effectiveness of the proposed method, we first conduct experimental comparisons against several baseline methods:

- SVR: We train a support vector regressor using the CNN representations from the first 6 layers of Fig. 5.7;

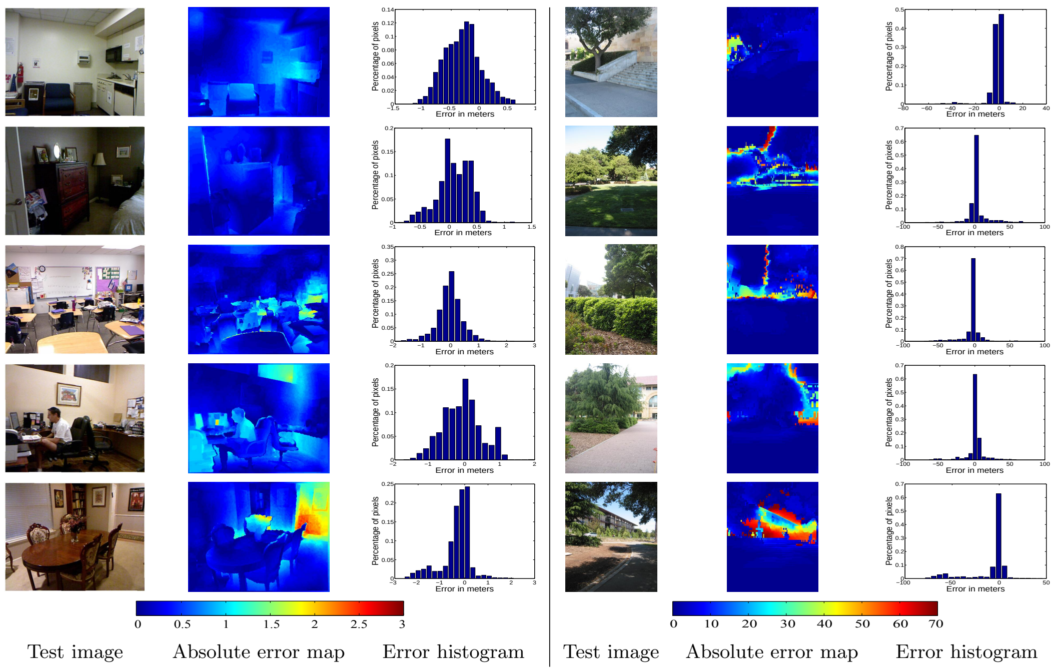


Figure 5.17: An illustration of the absolute error maps and the pixel-wise error histograms of our predictions (Left: NYU v2; Right: Make3D). The absolute error maps are shown in meters, with the color bar shown in the last row. For the error histogram plot, the horizontal axis shows the prediction error in meters (quantized into 20 bins), and the vertical axis shows the percentage of pixels in each bin.

- SVR (smooth): We add a smoothness term to the trained SVR during prediction by solving the inference problem in Eq. (5.21). As tuning multiple pairwise parameters is not straightforward, we only use color difference as the pairwise potential and choose the parameter β by hand-tuning on a validation set;
- Unary only: We replace the CRF loss layer in Fig. 5.6 with a least-square regression layer (by setting the pairwise outputs $R_{pq} = 0$, $p, q = 1, ..., n$), which degenerates to a deep regression model trained by SGD.
- Unary only (smooth): We add similar smoothness term to our unary only model, as did in the SVR (smooth) case.

5.4.1.1 NYU v2 Dataset

The NYU v2 dataset consists of 1449 RGBD images of indoor scenes, among which 795 are used for training and 654 for test (we use the standard training/test split provided with the dataset). We report the baseline comparisons in Table 5.1. From the table, several conclusions can be made: 1) When trained with only unary term, deeper network is beneficial for better performance, which is demonstrated by the fact that our unary only model outperforms the SVR model; 2) Adding smoothness term to the SVR or our

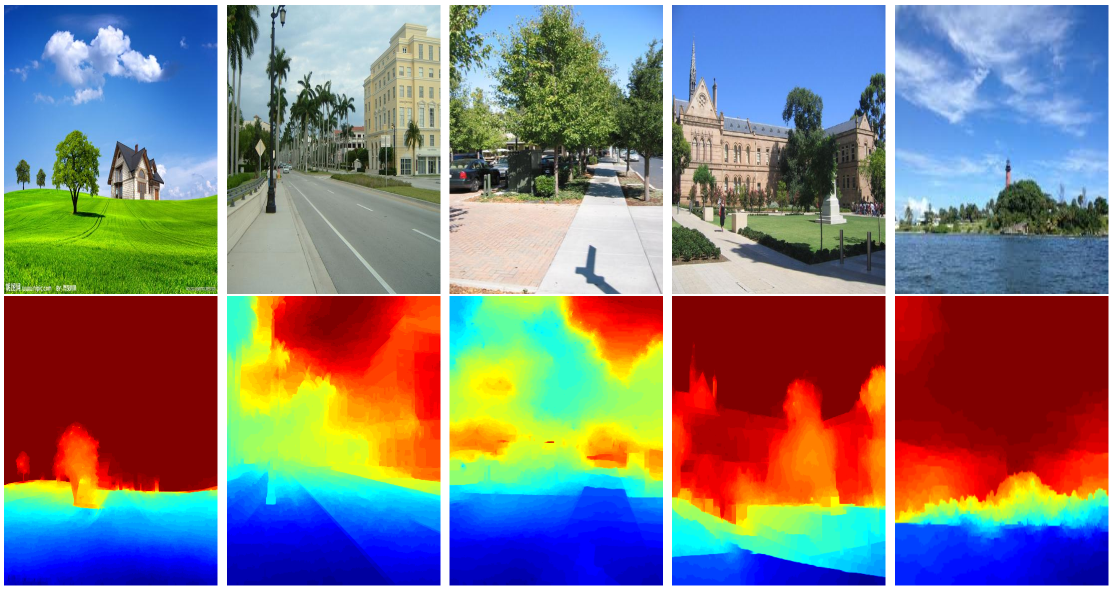


Figure 5.18: Examples of depth predictions on general outdoor scene images obtained from the Internet (First row: test images; second row: our depth predictions. Best viewed on screen). Depths are shown in log scale and in color (red indicates far and blue indicates close).

unary only model helps improve the prediction accuracy; 3) Our DCNF model achieves the best performance by jointly learning the unary and the pairwise parameters in a unified deep CNN framework. Moreover, fine-tuning the whole network yields further performance gain. These well demonstrate the efficacy of our model.

5.4.1.2 Make3D Dataset

The Make3D dataset contains 534 images depicting outdoor scenes, with 400 for training and 134 images for test. As pointed out in [8, 10], this dataset is with limitations: the maximum value of depths is 81m with far-away objects are all mapped to the one distance of 81 meters. As a remedy, two criteria are used in [10] to report the prediction errors: (C_1) Errors are calculated only in the regions with the ground-truth depth less than 70 meters; (C_2) Errors are calculated over the entire image. We follow this protocol to report the evaluation results in Table 5.2. As we can see, our full DCNF model with the whole network training performs the best among all the compared baseline methods. Using deeper networks and adding smoothness terms generally help improve the performance.

5.4.2 DCNF vs. DCNF-FCSP

In this section, we compare the performance of the proposed DCNF and DCNF-FCSP in terms of both prediction accuracy and computational efficiency. The compared prediction performance are reported in Table 5.3 and Table 5.4. We can see that the proposed DCNF-FCSP model performs very close to the DCNF model. Next, we compare the computational efficiency of these two models. Specifically, we report the training time (network forward + backward) of one image for both whole models in terms of different numbers of superpixels we use per image. The comparison is conducted on the NYU v2 dataset and is shown in Fig. 5.12. As demonstrated, the DCNF-FCSP model is generally orders of magnitude faster than the DCNF model. Moreover, the speedup becomes more significant with the increase of superpixels. We also compare the network forward time of whole models during depth predictions, and plot the results in Fig. 5.13. The shown time is for processing one image. We can see that the DCNF-FCSP model is much faster as well as more scalable than the DCNF model. Most importantly, with this more efficient DCNF-FCSP model, we can design deeper network for better performance, as we will show in the sequel in Sec. 5.4.3.

5.4.3 State-of-the-art Comparisons

Recent studies have shown that very deep networks can significantly improve the image classifications performance [99, 119]. Thanks to the speedup brought by the superpixel pooling method, we are now able to design deeper networks in our framework. We transfer the popular VGG-16 net trained on the ImageNet from [99]. Specifically, we replace the AlexNet part (the first 5 convolutional layers in Fig. 5.9) with all the convolutional layers (including the 5-th pooling layer) in VGG-16. These layers are followed by the 2 newly added convolutional layers with 512 channels each, and then the 3 fully connected layer to construct the unary potentials. We follow the same training protocol, *i.e.*, first pre-train the remaining layers by fixing the transferred layers (the VGG-16 net part) and then fine-tune the whole network.

5.4.3.1 NYU v2 Dataset

In Table 5.5, we report the results compared to several popular state-of-the-art methods on the NYU v2 dataset. As can be observed, our method outperforms classic methods like Make3d [8], DepthTransfer [7] with large margins. Most notably, our results are significantly better than that of [89], which jointly exploits depth estimation and semantic labelling. Compared to the recent work of Eigen *et al.* [9], our method also exhibits better performance in terms of all metrics. Note that, to overcome overfit, they [9] have to collect millions of additional labelled images to train their model. In contrast, we only use the standard training sets (795) without any extra data, yet we achieve better performance. Fig. 5.11 illustrates some qualitative evaluations of our method compared against Eigen *et al.* [9] (We download the predictions of [9] from the authors' website.). Compared to the coarse predictions of [9], our method yields better visualizations with sharper transitions, aligning to local details. To better illustrate how our predictions deviate from the ground-truth depths, we plot the absolute depth error maps and the pixel-wise error histograms in Fig. 5.17. Specifically, the absolute error maps are shown in meters, with the color bar shown in the last row. For the error histogram plot, the horizontal axis shows the prediction error in meters (quantized into 20 bins), and the vertical axis shows the percentage of pixels in each bin. As we can see, our predictions are mostly well aligned to the ground truth depth maps.

5.4.3.2 Make3D Dataset

We show the compared results on the Make3D dataset in Table 5.6. We can see that our DCNF-FCSP model with the whole network training ranks the first in overall performance, outperforming the compared methods by large margins. Note that the C2 errors of [10] are reported with an ad-hoc post-processing step, which trains a classifier to label sky pixels and set the corresponding regions to the maximum depth. In contrast, we do not employ any of those heuristics to refine our results, yet we achieve better results in terms of relative error. Compared to the results of the DCNF-FCSP model using smaller net in Table 5.4, we get better $C1$ error but degraded $C2$ error. This can be explained from the limitations of this dataset that the depths of all far away objects are all set to one maximum depth value. Some examples of qualitative evaluations are shown in Fig. 5.14. By jointly learning the unary and pairwise potentials, our DCNF-FCSP model produce predictions that well capture local details. The absolute depth error maps and the pixel-wise error histograms are shown in the right part of Fig. 5.17. As can be observed, our predictions are mostly well aligned to the ground truth depth maps, with most of the prediction errors are on the boundary regions which show extremely large depth jumps. In these cases, our predictions exhibit relatively mild depth transitions.

5.4.3.3 KITTI dataset

We further perform depth estimation on the KITTI dataset [120], which consists of videos taken from a driving vehicle with depths captured by a LiDaR sensor. We use the same test set, *i.e.*, 697 images from 28 scenes, as provided by Eigen [9]. As for the training set, we use the same 700 images that Eigen *et al.* [9] used to train the method of Saxena *et al.* [8]. Since the ground-truth depths of the KITTI dataset are scattered at irregularly spaced points, which only consists of $\sim 5\%$ pixels of each image, we extract the ground-truth depth closest to each superpixel centroid as the superpixel depth label. We then construct our CRF graph only on those superpixels that have ground-truth labels. The compared results are presented in Table 5.7. In summary, Eigen *et al.* [9] have achieved better performance on this dataset by leveraging large amounts of training data (in the millions). This can be explained by the fact that the highly sparse ground-truth depth maps lessened the benefits of the pairwise term in our model. Fig. 5.15 shows some prediction examples.

5.4.4 Generalization to Depth Estimations of General Scene Images

As stated, our model aims to predict depths from single monocular images depicting general scenes. We thus demonstrate the generalization ability of our trained models (indoor and outdoor) on random images that irrelevant to the training data. Fig. 5.16 shows some prediction examples of our trained indoor model on general indoor scene images obtained from the Internet. Similarly, Fig. 5.18 shows some prediction examples of our trained outdoor model on general outdoor scene images obtained from the Internet. As we can see, the predictions are generally reasonable, well preserving the space hierarchy. It is worth noting that this generalization works well at capturing relative depths, but not absolute depths, since our trained model relies solely on appearance information. Despite this drawback, our depth predictions can still benefit other vision tasks, *e.g.*, semantic labelling, scene recognition *etc.*. In [121], the authors incorporated the predicted depths of our model for place recognition and demonstrate its benefits.

5.5 Conclusion

We have presented a deep convolutional neural field model for depth estimation from a single image. The proposed method combines the strength of deep CNN and continuous CRF in a unified CNN framework. We show that the log-likelihood optimization in our method can be directly solved using back propagation without any approximations required. Predicting the depths of a new image by solving the MAP inference can be efficiently performed as closed-form solutions exist. We further propose an improved model that based on fully convolutional networks and a novel superpixel pooling method. We experimentally demonstrate that it is equally effective while brings orders of magnitude faster training speedup, which enables the use of deeper networks for better performance. Given the general learning framework of our method, it is also possible to be applied to other vision applications with minimum modification, *e.g.*, image denoising, and deblurring. Experimental results demonstrate that the proposed method outperforms state-of-the-art methods on both indoor and outdoor scene datasets.

Chapter 6

Conclusion

This thesis makes practical contributions in learning structured prediction models for computer vision applications. Specifically, our efforts are towards nonlinear CRFs learning, which typically yields better performance than linear CRFs models. We experimentally demonstrate that our proposed methods achieve state-of-the-art results on the image segmentation and single image depth estimation tasks.

In Chapter 3, we exploit the power of decision trees for constructing potentials in CRFs models for image segmentation. Specifically, we model the unary and the pairwise potentials of CRFs as two groups of ensembled decision trees. We then jointly learn the decision trees and the ensemble parameters in a unified optimization framework. We therefore achieve nonlinear learning of both the unary potential and the pairwise potential, which distinguishes our method from most of the existing methods. Furthermore, our method enables learning class-specific decision trees for each category of objects that appear in the image, which we refer as object-aware CRFs learning. This can bring additional performance gain. We show that the resulted optimization problem can be efficiently solved by combining a modified column generation technique and a cutting-plane algorithm.

In Chapter 4, we leverage a CNN model pre-trained on a large image dataset, *i.e.*, ImageNet, for CRFs learning for the task of image segmentation. Namely, we transfer a CNN model trained for image classification to semantic segmentation. The trained CNN model is used for generating deep features for constructing unary potentials of superpixels. For the pairwise potential, we incorporate co-occurrence information by constructing spatial related concurrence potential functions. It injects prior knowledge into learning and brings the benefit of encouraging reasonable labellings while preventing implausible labellings. With the constructed unary and pairwise potentials, we then learn the CRFs parameters within the max-margin framework. We conduct extensive experiments on binary as well as multi-class segmentation datasets and demonstrate state-of-the-art performance. Additional contributions include that we perform considerable comparisons among traditional hand-crafted features, unsupervised learned features and CNN features, and make valuable conclusions.

In Chapter 5, we propose a joint CNN and continuous CRFs learning framework, *i.e.*, deep convolutional neural fields (DCNF), for depth estimation from single images. In more detail, we propose to model the depth estimation task as continuous CRFs learning problem. We then design CNN networks for constructing the unary and the pairwise potentials of the CRFs model. Due to the continuous property of the output depth values, the partition function in the CRFs model can be explicitly calculated without approximations. We therefore can directly solve the log-likelihood maximization through back-propagations. Predicting the depth of a new image is efficient since we have closed-form solution for the MAP inference problem. Furthermore, we propose an improved model based on fully convolutional networks and a novel superpixel pooling method, termed as DCNF-FCSP, to speedup the patch-wise convolutions in the originally proposed model. We experimentally demonstrate that this new model is equally effective while being orders of magnitude faster. Comprehensive evaluations on both indoor and outdoor datasets show that our method significantly outperform state-of-the-art methods. We also show that our trained models can be used for depth estimation of general scene images, which indicates potentials for benefiting other vision tasks.

6.1 Future Work

We have been focusing on the specific problem of structured learning for computer vision applications in this thesis. Next, we discussion several potential directions for future work.

6.1.1 Deep Structured Learning

In the most recent years, deep learning methods have been setting new records in various aspects of machine learning related applications, *e.g.*, computer vision, natural language processing, bioinformatics, *etc.*. While most of these successes have been focusing on relatively simple tasks, *e.g.*, classification, regression, less progress has been made in more complicated applications. Hence learning deep models for structured prediction problems becomes a promising working direction.

Deep structured learning puts the hierarchical representation learning and modelling the dependencies among the variables of interest in a unified framework, which is potentially promising to benefiting both. However, training deep models typically requires huge amounts of data, which poses challenges for structured learning in this scenario. This is because current structured learning methods are generally not able to scale to such large datasets. A simple solution is to rely on a two-step process, which first trains a unary model using deep features and then incorporate spatial terms as a post-processing step, as did in [43, 44]. This learning scheme is suboptimal since the two decoupled steps are agnostic about each other. In contrast, joint learning can be expected to achieve better performance, since all the parameters are learned within a unified framework to optimize a single objective.

Moreover, learning deep structured models calls for more efficient inference methods, which can also be a potential direction for future work.

6.1.2 Semi-supervised Structured Learning

One of the practical obstacles in applying structured learning is that training requires obtaining a considerable number of labelled data. Manually annotating large amounts of data can be expensive while plentiful of unlabelled data are readily available. Furthermore, training structured models on large scale fully annotated datasets is generally inefficient. This makes semi-supervised structured learning an appealing direction.

By leveraging large amounts of unlabelled data, semi-supervised learning enables structured training on a limited number of annotated data. In the case of generative models, unlablled data can be naturally incorporated through the expectation maximization [122]. However, it remains an open problem how to make use of the large quantities of unlabelled data in the discriminative models. This can be explored in the future work.

Bibliography

[1] B. Taskar. Learning structured prediction models: A large margin approach. `http://www.seas.upenn.edu/~taskar/pubs/thesis.pdf`, 2004. PhD thesis, Stanford University.

[2] Yann LeCun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[3] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proceedings of European Conference on Computer Vision*, 2014.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2012.

[5] David C. Lee, Abhinav Gupta, Martial Hebert, and Takeo Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *Proceedings of Advances in Neural Information Processing Systems*, 2010.

[6] Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Proceedings of European Conference on Computer Vision*, 2010.

[7] Kevin Karsch, Ce Liu, and Sing Bing Kang. Depthtransfer: Depth extraction from video using non-parametric sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[8] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3D: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[9] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Proceedings of Advances in Neural Information Processing Systems*, 2014.

[10] Miaomiao Liu, Mathieu Salzmann, and Xuming He. Discrete-continuous depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[11] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

[12] John Lafferty, Xiaojin Zhu, and Yan Liu. Kernel conditional random fields: Representation and clique selection. In *Proceedings of the International Conference on Machine Learning*, 2004.

[13] Fayao Liu, Guosheng Lin, and Chunhua Shen. CRF learning with CNN features for image segmentation. *Pattern Recognition*, 2015.

[14] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[15] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian D. Reid. Learning depth from single monocular images using deep convolutional neural fields. *CoRR*, abs/1502.07411, 2015.

[16] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

[17] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 1995.

[18] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 1994.

[19] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector mchines. *The Journal of Machine Learning Research*, 2001.

[20] Ross Kindermann, J. Laurie James Laurie Snell, and American mathematical society. *Markov random fields and their applications.* Contemporary mathematics. Providence, R.I. American Mathematical Society, 1980.

[21] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning*, 2004.

[22] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2004.

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

[24] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *Proceedings of European Conference on Computer Vision*, 2008.

[25] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. Kernelized structural SVM learning for supervised object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[26] A. Lucchi, Y. Li, K. Smith, and P. Fua. Structured image segmentation using kernelized features. In *Proceedings of European Conference on Computer Vision*, 2012.

[27] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 2011.

[28] Zhen Zhang, Qinfeng Shi, Yanning Zhang, Chunhua Shen, and Anton van den Hengel. Constraint reduction using marginal polytope diagrams for map lp relaxations. *CoRR*, 2013. URL `http://arxiv.org/abs/1312.4637`.

[29] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *The Journal of Machine Learning Research*, 2005.

[30] J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 1960.

[31] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[32] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 2012.

[33] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, De-Sheng Wang, and Hang Li. Global ranking using continuous conditional random fields. In *Proceedings of Advances in Neural Information Processing Systems*, 2008.

[34] Thomas G. Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems*, 2000.

[35] *Ensemble Methods for Structured Prediction*, 2014.

[36] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 2002.

[37] C. Shen and Z. Hao. A direct formulation for totally-corrective multi-class boosting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[38] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

[39] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014.

[40] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[41] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[42] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2014.

[43] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, 2014. URL `http://arxiv.org/abs/1412.7062`.

[44] Jonathan Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Proceedings of Advances in Neural Information Processing Systems*, 2014.

[45] Liang-Chieh Chen, Alexander G. Schwing, Alan L. Yuille, and Raquel Urtasun. Learning deep structured models. In *Proceedings of the International Conference on Machine Learning*, 2015.

[46] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. *CoRR*, abs/1502.03240, 2015.

[47] Guosheng Lin, Chunhua Shen, Ian D. Reid, and Anton van den Hengel. Efficient piecewise training of deep structured models for semantic segmentation. *CoRR*, abs/1504.01013, 2015.

[48] Guosheng Lin, Chunhua Shen, Ian D. Reid, and Anton van den Hengel. Deeply learning the messages in message passing inference. *CoRR*, abs/1506.02108, 2015.

[49] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

[50] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[51] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Proceedings of the International Conference on Computer Vision*, 2009.

[52] S. Nowozin, P. Gehler, and C. H. Lampert. On parameter learning in CRF-based approaches to object class image segmentation. In *Proceedings of European Conference on Computer Vision*, 2010.

[53] C.-N. Yu and T. Joachims. Training structural SVMs with kernels using sampled cuts. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2008.

[54] A. Severyn and A. Moschitti. Fast support vector machines for structural kernels. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2011.

[55] I. Steinwart. Sparseness of support vector machines. *The Journal of Machine Learning Research*, 2003.

[56] Chunhua Shen, Guosheng Lin, and Anton van den Hengel. Structboost: Boosting methods for predicting structured output variables. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[57] Sebastian Nowozin, Carsten Rother, Shai Bagon, Toby Sharp, Bangpeng Yao, and Pushmeet Kohli. Decision tree fields. In *Proceedings of the International Conference on Computer Vision*, 2011.

[58] Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2006.

[59] Adam Coates and Andrew Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the International Conference on Machine Learning*, 2011.

[60] Daniel Kuettel and Vittorio Ferrari. Figure-ground segmentation by transferring window masks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[61] Maria-Elena Nilsback and Andrew Zisserman. Delving deeper into the whorl of flower segmentation. *Image and Vision Computing*, 2010.

[62] Marcin Marszalek and Cordelia Schmid. Accurate object localization with shape masks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[63] Radhakrishna Achanta, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Ssstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[64] Anat Levin and Yair Weiss. Learning to combine bottom-up and top-down segmentation. In *Proceedings of European Conference on Computer Vision*, 2006.

[65] Armand Joulin, Francis R. Bach, and Jean Ponce. Discriminative clustering for image co-segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[66] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Localizing objects with smart dictionaries. In *Proceedings of European Conference on Computer Vision*, 2008.

[67] David Aldavert, Arnau Ramisa, Ramon Lpez de Mntaras, and Ricardo Toledo. Fast and robust object segmentation with the integral linear classifier. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[68] Lubor Ladicky, Christopher Russell, Pushmeet Kohli, and Philip H. S. Torr. Associative hierarchical crfs for object class image segmentation. In *Proceedings of the International Conference on Computer Vision*, 2009.

[69] Josep M. Gonfaus, Xavier Boix Bosch, Joost van de Weijer, Andrew D. Bagdanov, Joan Serrat Gual, and Jordi Gonzàlez Sabaté. Harmony potentials for joint classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[70] Aurelien Lucchi, Yunpeng Li, and Pascal Fua. Learning for structured prediction using approximate subgradient descent with working sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[71] Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[72] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006.

[73] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the International Conference on Machine Learning*, 2014.

[74] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[75] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2007.

[76] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[77] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *Proceedings of the International Conference on Computer Vision*, 2007.

[78] Lubor Ladicky, Christopher Russell, Pushmeet Kohli, and Philip H. S. Torr. Inference methods for crfs with co-occurrence statistics. *International Journal of Computer Vision*, 2013.

[79] Anirban Roy and Sinisa Todorovic. Scene labeling using beam search under mutex constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[80] David Grangier, Lon Bottou, and Ronan Collobert. Deep convolutional networks for scene parsing. In *Proceedings of the International Conference on Machine Learning Deep Learning Workshop*, 2009.

[81] Hannes Schulz and Sven Behnke. Learning object-class segmentation with convolutional neural networks. In *Proceedings of the European Symposium on Artificial Neural Networks*, 2012.

[82] Yangqing Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

[83] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *Proceedings of the International Conference on Computer Vision*, 2009.

[84] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 2010.

[85] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 2010.

[86] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Free-form region description with second-order pooling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[87] Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. Stacked hierarchical labeling. In *Proceedings of European Conference on Computer Vision*, 2010.

[88] Victor S. Lempitsky, Andrea Vedaldi, and Andrew Zisserman. Pylon model for semantic segmentation. In *Proceedings of Advances in Neural Information Processing Systems*, 2011.

[89] Lubor Ladick, Jianbo Shi, and Marc Pollefeys. Pulling things out of perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[90] Jamie Shotton, Ross B. Girshick, Andrew W. Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[91] Varsha Hedau, Derek Hoiem, and David A. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *Proceedings of European Conference on Computer Vision*, 2010.

[92] Bryan C. Russell and Antonio Torralba. Building a database of 3d scenes from user annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[93] Beyang Liu, Stephen Gould, and Daphne Koller. Single image depth estimation from predicted semantic labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[94] Vladan Radosavljevic, Slobodan Vucetic, and Zoran Obradovic. Continuous conditional random fields for regression in remote sensing. In *Proceedings of the Biennial European Conference on Artificial Intelligence*, 2010.

[95] Kosta Ristovski, Vladan Radosavljevic, Slobodan Vucetic, and Zoran Obradovic. Continuous conditional random fields for efficient regression in large fully connected graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2013.

[96] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2012.

[97] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[98] Michael Cogswell, Xiao Lin, Senthil Purushwalkam, and Dhruv Batra. Combining the best of graphical models and convnets for semantic segmentation. *CoRR*, 2014. URL `http://arxiv.org/abs/1412.4313`.

[99] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*, 2015.

[100] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[101] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 2002.

[102] David A. Forsyth and Jean Ponce. *Computer Vision A Modern Approach.* Prentice Hall, 2003.

[103] Subhodev Das and Narendra Ahuja. Performance analysis of stereo, vergence, and focus as depth cues for active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[104] Tony Lindeberg and Jonas Garding. Shape from texture from a multi-scale perspective. In *Proceedings of the International Conference on Computer Vision*, 1993.

[105] Jitendra Malik and Ruth Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, 1997.

[106] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999.

[107] Takayuki Nagai, Masaaki Ikehara, and Akira Kurematsu. Hmm-based surface reconstruction from single images. *Proceedings of the International Conference on Image Processing*, 2002.

[108] Janusz Konrad, Meng Wang, and Prakash Ishwar. 2d-to-3d image conversion by learning depth from examples. In *CVPR Workshops*, 2012.

[109] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. In *Proceedings of Advances in Neural Information Processing Systems*, 2005.

[110] Tadas Baltrušaitis, Louis-Philippe Morency, and Peter Robinson. Continuous conditional neural fields for structured regression. In *Proceedings of European Conference on Computer Vision*, 2014.

[111] David Eigen, Dilip Krishnan, and Rob Fergus. Restoring an image taken through a window covered with dirt or rain. In *Proceedings of the International Conference on Computer Vision*, 2013.

[112] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Proceedings of European Conference on Computer Vision*, 2014.

[113] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. In *Proceedings of the International Conference on Learning Representations*, 2014.

[114] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of the International Conference on Pattern Recognition*, 1994.

[115] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference*, 2014.

[116] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *Proceedings of European Conference on Computer Vision*, 2012.

[117] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[118] Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 2002.

[119] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[120] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Robotics Res.*, 2013.

[121] M. Milford, C. Shen, S. Lowry, N. Suenderhauf, S. Shirazi, G. Lin, F. Liu, E. Pepperell, C. Lerma, B. Upcroft, and I. Reid. Sequence searching with deep-learnt depth for condition- and viewpoint-invariant route-based place recognition. In *6th International Workshop on Computer Vision in Vehicle Technology, in conjunction with IEEE Conference on Computer Vision and Pattern Recognition (CVVT'15)*, 2015.

[122] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 2000.