université
PARIS-SACLAY

PHD THESIS

to obtain the title of

Doctor of the Université Paris-Saclay

Doctoral School STIC (580)
Sciences et Technologies de l'Information et de la Communication

Speciality : Mathematiques & Informatique

Defended by

Siddhartha CHANDRA

Efficient and Exact Deep Structured Prediction for Dense Labeling Tasks in Computer Vision

Thesis Advisor: Iasonas KOKKINOS

prepared at the Center for Visual Computing of
CentraleSupelec / INRIA Saclay (GALEN Team)

Jury :

Reviewers :	Nikos KOMODAKIS	-	Ecole des Ponts ParisTech
	Victor LEMPITSKY	-	Skolkovo Inst. of Science and Technology
Advisor :	Iasonas KOKKINOS	-	University College London & Facebook Artificial Intelligence Research
Chair :	Nikos PARAGIOS	-	CentraleSupelec (CVN) / INRIA
Examinators :	Lourdes AGAPITO	-	University College London
	Camille COUPRIE	-	Facebook Artificial Intelligence Research
	Ivan LAPTEV	-	Ecole Normale Superieure / INRIA

Efficient Deep Structured Prediction for Dense Labeling Tasks in Computer Vision

Abstract

In this thesis we propose a structured prediction technique that combines the virtues of Gaussian Conditional Random Fields (G-CRFs) with Convolutional Neural Networks (CNNs).

The starting point of this thesis is the observation that while being of a limited form G-CRFs allow us to perform exact Maximum-A-Posteriori (MAP) inference efficiently. We prefer exactness and simplicity over generality and advocate G-CRF based structured prediction in deep learning pipelines. Our proposed structured prediction methods accomodate (i) exact inference, (ii) both short- and long- term pairwise interactions, (iii) rich CNN-based expressions for the pairwise terms, and (iv) end-to-end training alongside CNNs. We devise novel implementation strategies which allow us to overcome memory and computational challenges when dealing with fully-connected graphical models. We perform extensive experimental studies and demonstrate the utility of our methods by showing empirical improvements over strong baselines on a variety of computer vision benchmarks.

We first propose a structured prediction model for capturing short-range interactions via a sparsely-connected graphical model in Chap. 2. For this model, we introduce multi-resolution architectures to couple information across scales in a joint optimization framework, yielding systematic improvements. We extend this model to capture Potts-type pairwise terms, reducing the memory and computational complexity of our method. We demonstrate the utility of our approach on the challenging VOC PASCAL 2012 image segmentation benchmark, showing substantial improvements over strong baselines.

In Chap. 3, we endow the Deep G-CRF model with a densely connected graph structure. To cope with the prohibitive memory and computational demands of a fully-connected graphical model we express the pairwise interactions as inner products of low-dimensional, learnable embeddings. The G-CRF system matrix is therefore low-rank, allowing us to solve the resulting system very efficiently on the GPU by using the conjugate gradient algorithm. In this context, we adapt the conjugate gradient algorithm to accommodate this low-rank structure. By virtue of these customizations, we demonstrate that the complexity of allowing fully-connected graphical structure comes at negligible computational overhead compared to our sparse model in Chap. 2. We also develop even faster, Potts-type variants of our embeddings. We show that the learned embeddings capture pixel to-pixel affinities in a task-specific manner, while our approach achieves state of the art results on three challenging benchmarks, namely semantic segmentation, human part segmentation, and saliency estimation.

Finally in Chap. 4, we introduce a time- and memory-efficient method for structured prediction that couples neuron decisions across both space at time. We show that we are able to perform exact and efficient inference on a densely-connected spatio-temporal graph by customizing the standard conjugate gradient algorithm to the particular spatio-temporal structure we use. We experiment with multiple connectivity patterns in the temporal domain, and present empirical improvements over strong baselines on the tasks of instance tracking, semantic and instance segmentation for videos.

Contents

List of Figures

Chapter 1

Introduction

The goal of this thesis is to develop efficient and exact strategies for structured prediction in Convolutional Neural Networks (CNNs) for dense labeling tasks in computer vision. Our primary contributions in this thesis are the following:

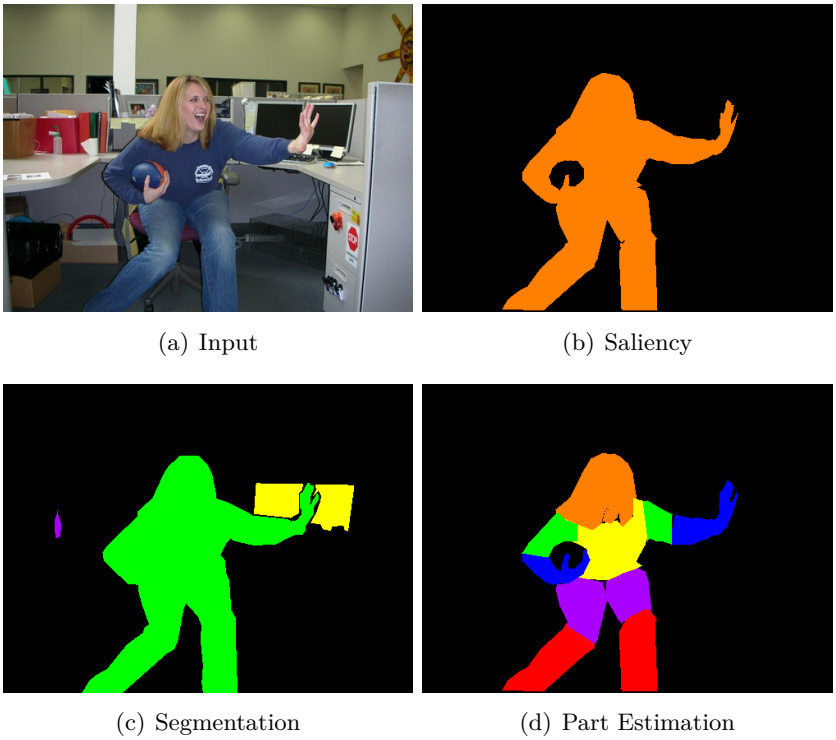
1. A structured prediction approach for CNNs with sparsely connected graphical models.
2. Extending our approach to fully connected graphical models while keeping the computational and memory demands controllable.
3. Spatio-temporal structured prediction for video understanding.

Our proposed strategies are efficient in time and memory, allow exact inference and are end-to-end trainable via back-propagation. In order to position our contributions with respect to the broader structured prediction context, in Sec. 1.1 we begin by first defining dense labeling tasks in the context of this thesis, followed by a review of the relevant deep learning literature. We then give a brief overview of structured prediction in Sec. 1.2, followed by a review of the relevant structured-prediction literature in the context of deep learning in Sec. 1.3. Finally in Sec. 1.4, we discuss the contributions of this thesis.

1.1 Dense Labeling Tasks

In the context of this thesis, we use the term *dense labeling tasks* to describe computer vision problems where the objective is to assign one label to each pixel in the image. Semantic segmentation is one example of a dense labeling problem, where each pixel is assigned a label which corresponds to the object/background class. Localization of human parts is another example, where each pixel is assigned a label corresponding to the body part or background. Saliency estimation is yet another example, where each pixel is assigned a label indicating whether the pixel is interesting (salient) or not. These problems are shown in Fig. 1.1. Furthermore, dense labeling of pixels in a video with multiple image frames also falls in the category of dense labeling tasks, for a spatio-temporal input signal.

So far, the labels we have considered are symbolic, and represent concepts (the class/type of pixel) and not quantities. There are other problems where the labels represent quantities, such as depth and surface normal. We refer to these problems as dense regression problems, where the goal is to estimate real values at each pixel. This is shown in Fig. 1.2. Even though in this thesis we focus on dense labeling



(a) Input (b) Saliency

(c) Segmentation (d) Part Estimation

Figure 1.1: Examples of Dense Labeling Tasks: Saliency Estimation, Semantic Segmentation, and Human Part Estimation.

problems all our methods described in Chap. 2, 3 and 4 are also applicable to dense regression problems.

One way to solve dense labeling problems is to use a 'sliding window' classifier, which consists of the following steps: (i) we choose fixed sized patches centered at each pixel in the image, (ii) we extract features for each pixel by computing features on the corresponding patch, and (iii) we label each pixel by using a model which classifies the features computed at that pixel.

Typically such an image labeling pipeline involves two components: (i) the feature extractor, which extracts meaningful features from the each patch, and (ii) the classifier which learns to make predictions by using the statistics of the features. A significant part of the last three decades of research in computer vision was dedicated towards designing sophisticated hand-crafted feature extractors [Lowe 2004, Dalal 2005, Shotton 2009, Perronnin 2010] and classifiers [Cortes 1995, Breiman 2001, Jancsary 2012] (detailed comparisons and analysis of hand-crafted features and models based on them are available in [Chatfield 2011]). More recently, deep learning methods have returned to the fore [Krizhevsky 2012, Simonyan 2015, He 2016, Chatfield 2014] with the advancements in computer hardware and the availability of very large publicly available datasets [Len 2014, Deng 2009] for vision tasks. Deep learning methods [Bengio 2013, Schmidhuber 2015] combine

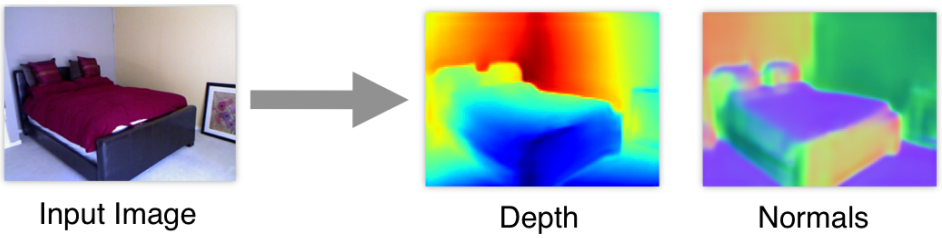


Figure 1.2: Examples of Dense Regression Tasks: Depth and Surface Normal Estimation.

(a) rich feature extractors consisting of hierarchical networks composed of simple units performing linear and non-linear computations on the data, and (b) relatively simple decision layers trained with standard l-2 or cross-entropy losses for regression and classification respectively. We now discuss some of the relevant deep learning literature which addresses dense labeling tasks, before turning to structured prediction and its interplay with deep learning in Sec. 1.2 and 1.3.

1.1.1 Deep Learning for Dense Labeling Tasks

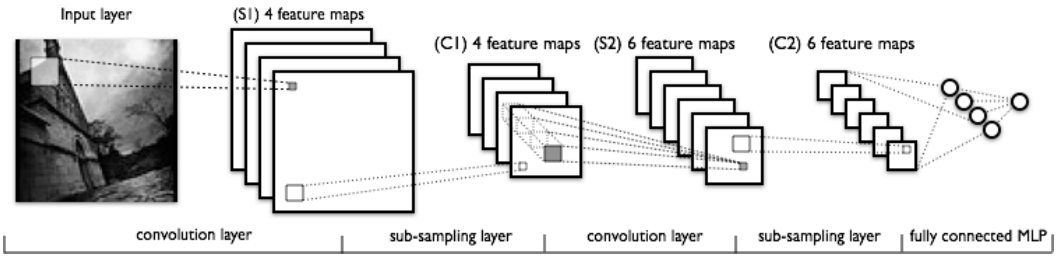


Figure 1.3: Convolutional Neural Network: A schematic representation of the LeNet-5 CNN from [Lecun 1998], showing the common deep learning modules, namely convolution and sub-sampling. The convolutional backbone of the network performs local, patch-based feature extraction. This network also employs fully-connected layers to obtain a global image-representation from these local features.

Our discussion on deep learning for dense labeling tasks begins with a brief overview of Convolutional Neural Networks (CNNs) [Lecun 1998] which were designed to recognize visual patterns directly from pixel images with minimal preprocessing. CNNs are layered networks primarily composed of three kinds of operations (i) local, spatially invariant linear operations (convolutions), (ii) non-linear activation functions, and (iii) sub-sampling operations. The convolution operator scans the units from the previous layer in a sliding window fashion and linearly transforms the inputs onto a feature space dictated by its internal parameters. This is followed by a non-linear activation function such as sigmoid or ReLU [Hahnloser 2000]. Spatial sub-sampling via averaging, max-pooling or random-sampling is also employed

at various stages of the network to reduce the sensitivity of the output to small shifts and variations in the input. CNNs also often employ fully-connected layers to summarize local features from different image regions into global image-level representations (Fig. 1.3). Since these modules are fairly common by now, we refer to [Lecun 1998, Hahnloser 2000] for details and focus below on deep learning literature related to dense labeling tasks.

Initial Attempts. Early methods addressing dense labeling tasks exploited CNNs as feature extractors and relied on probabilistic graphical model based inference techniques to obtain pixel-level labels [Farabet 2013, Gupta 2014]. Some of these approaches trained CNNs with patches of images, and classified each image patch to get a dense labeling over the image [Ganin 2014, Pinheiro 2014, Girshick 2014]. The authors in [Farabet 2013, Mostajabi 2015] used CNNs to extract deep features for super-pixels and used these features in combination with traditional scene labeling strategies such as non-linear classifiers or inference on segmentation trees. While these approaches harnessed the expressive power of deep networks, their performance was dependent on the performance of the super-pixel extraction methods they employed. Hariharan *et al.* [Hariharan 2015] combined convolutional features from different stages of the network via upsampling and element-wise addition and referred to the resulting feature representation as hypercolumns. They used these features to train pixel-level classifiers. The idea of combining convolutional features from different stages of the network was also explored previously in [Farabet 2012, Farabet 2013, Sermanet 2013].

Fully Convolutional Networks. While these initial attempts gave promising results, they suffered from the limitation that they used features from a network trained for a different task. Due to this discrepancy, the next natural step was to train CNNs that produced dense, i.e. one per pixel, predictions. This gave rise to "fully-convolutional networks" (FCNs) shown in [Long 2015], which were first introduced in [Lecun 1998] as space displacement networks. As the name suggests, FCNs are CNNs without fully connected layers, and produce outputs which spatially correspond to patches in the input image. In other terms, if we remove the fully connected layers from the network from [Lecun 1998] in Fig. 1.3, the resulting network will be a fully-connected network. Long and Shelhamer [Long 2015] showed that networks pretrained for other vision tasks such as classification could be adapted for pixel-wise labeling by replacing the fully connected layers with convolutional layers, and finetuning them with pixel-level annotations using the softmax cross-entropy loss. FCNs were previously also used by [Wolf 1994, Matan 1992] for other domains. Today FCNs are used ubiquitously for dense labeling tasks as in [Xie 2015, Liu 2016].

Atrous Convolution and Residual Networks. A major challenge that presented itself in the use of FCNs for dense labeling tasks was the downsampling factor, also referred to as the network stride. The output scores predicted by a CNN

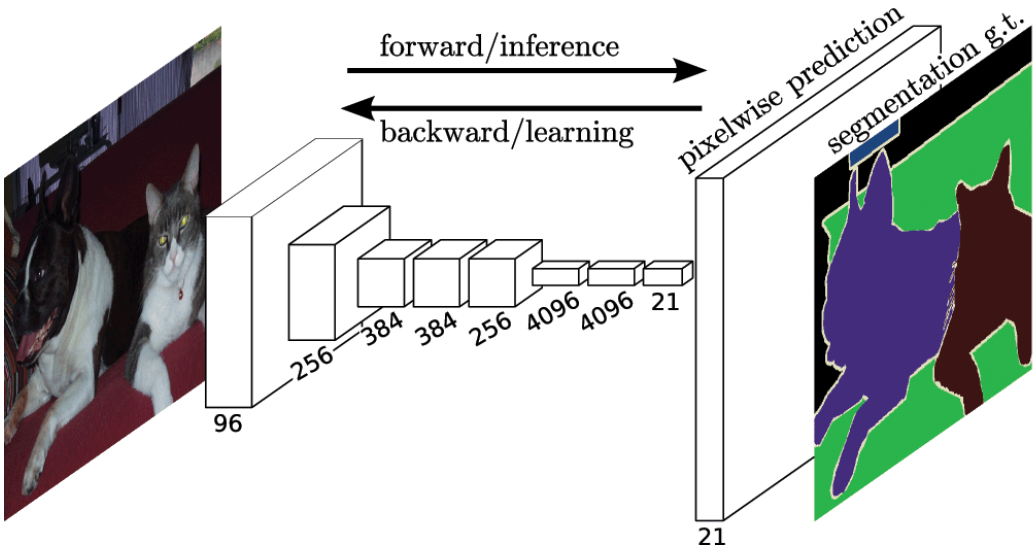
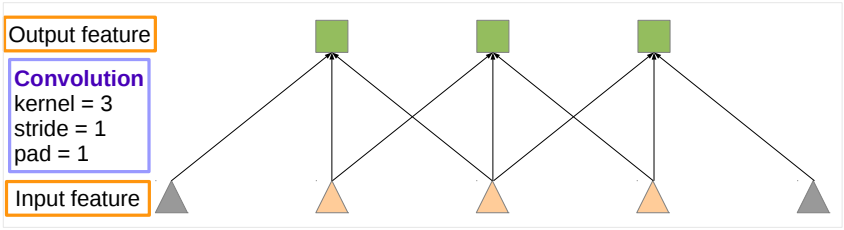


Figure 1.4: Fully Convolutional Networks (FCNs): FCNs were used in [Long 2015] for semantic segmentation. FCNs are CNNs that do not use any fully connected layers (in contrast with Fig. 1.3). This allows them to produce outputs that spatially correspond to patches in the input image.

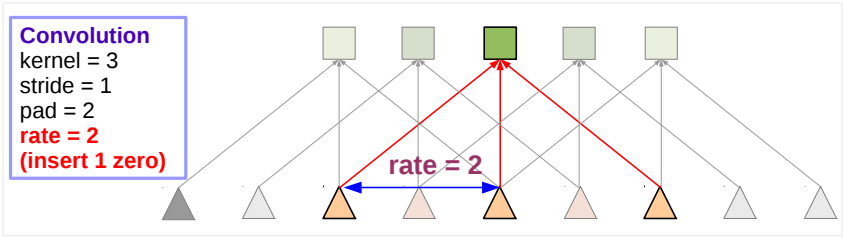
are smaller in spatial size than the input image due to repeated max-pooling and convolutional striding operations. Thus, obtaining a labeling that is the same size as the input image requires upsampling of the output scores via interpolation. This results in quantization and approximation errors. The downsampling factor of the FCN component of classification networks such as [Krizhevsky 2012,Simonyan 2015] is 32. This means each output unit corresponds to a 32×32 patch in the input image. Long and Shelhamer [Long 2015] proposed the use of a deconvolution filter which is a backwards convolution operation to upsample the output, thereby reducing the downsampling factor to 16. However this results in an increased number of parameters and longer training time.

Chen *et al.* [Chen 2014a] remedied this problem by using the *atrous* algorithm to introduce holes in the convolution kernel, thereby reducing the downsampling factor to 8. This is illustrated in Fig. 1.5: the use of atrous convolutions allows obtaining outputs at a desired receptive field of view (by controlling the kernel size) without a loss in spatial resolution or increase in the number of parameters. Further, the number of convolution parameters does not increase. The authors in [Yu 2016] employed the same operation to reduce the downsampling factor and capture contextual information, rebranding it as 'dilated convolutions'.

More recently deep residual networks [He 2016] were introduced which ease the training of very deep networks (with 101 or 151 convolutional layers) by adding 'residual' connections to the network architecture. The building block of a residual network, referred to as a residual block is illustrated in Fig. 1.6. He *et al.* [He 2016] hypothesize that optimizing the residual mapping is easier than optimizing the



(a) Standard Convolution



(b) Atrous Convolution

Figure 1.5: Comparison of standard convolution with atrous convolution [Chen 2014a] on a 1−D feature image. (a) shows the standard convolution with a 3×1 kernel where the input features (yellow triangles) are padded by 1 pixel (gray triangles) on either side to obtain an output feature map (outputs shown as green squares) of the same size. (b) shows the atrous convolution with a 3×1 kernel with a dilation rate of 2 where the input features are padded by 2 pixels on either side. The rate parameter denotes the stride with which the input features are sampled, and allows 'holes' in the convolution kernel. This allows computing an output feature map with the same size as the input feature map without increasing the number of convolution parameters. Further, atrous convolutions can be viewed as dilated convolutions since holes in the convolution kernel expand the field of view of the filter allowing a larger context to be captured.

original, unreferenced mapping. The success of residual networks on the Imagenet benchmark [Deng 2009] caused them to be quickly adapted to semantic segmentation networks by [Chen 2015b, Zhao 2016]. Today residual networks are also being used for other dense labeling tasks such as edge detection [Yu 2017] and depth estimation [Laina 2016]. The authors in [Zagoruyko 2016] show that a performance similar to residual networks can be achieved by shallower networks with more parameters. Inspired by residual networks, authors in [Huang 2017] proposed the Dense Convolutional Network (DenseNet) which connects each layer to every other layer in the network. Finally, the authors in [Fu 2017] use stacks of multiple shallow deconvolutional networks to capture multi-scale context, and demonstrate state of the art results on a variety of semantic segmentation benchmarks.

In the next section, we discuss the shortcomings of FCNs and introduce the notion of structured prediction which attempts to address them.

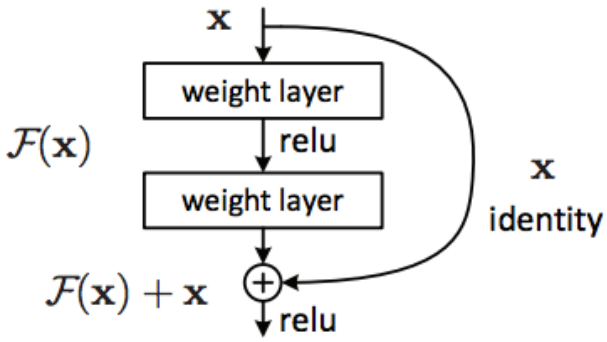
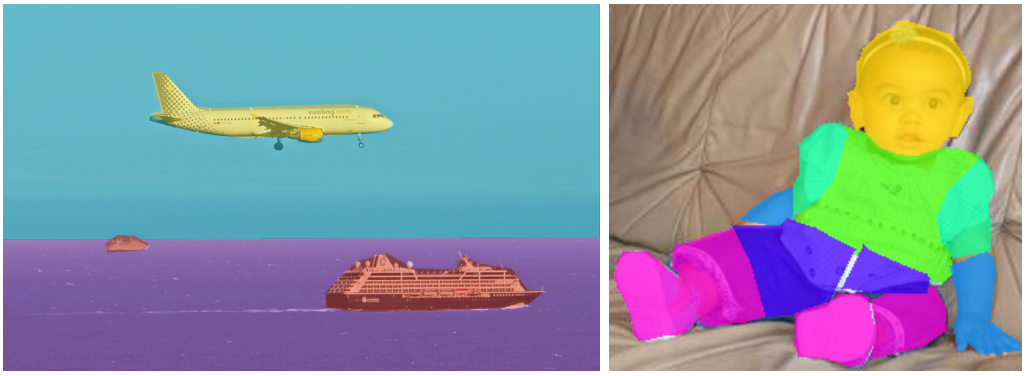


Figure 1.6: A residual block: fundamental unit of a residual network [He 2016]. In parallel to a stack of layers that perform non-linear mapping of the input data, the residual block contains a shortcut connection of identity mapping without adding any extra parameter or computational complexity. In simpler terms, a residual block recasts the original mapping as $F(x) + x$. It is hypothesized that optimizing the residual mapping is easier than optimizing the original, unreferenced mapping.

1.2 Structured Prediction

Shortcomings of FCNs. The FCN models have an important limitation: they do not explicitly capture visual context. The FCN models described so far look at each image patch in isolation and ignore any interdependencies between the labels of the patches. Some of these interdependencies are indeed captured by CNNs through a cascade of convolutions (since the patches corresponding to nearby pixels overlap, and the weights for the classifier and feature extractor are learnt from the data, taking local context into account). However, these interactions are implicit, and there is no means of enforcing pairwise constraints, such as *label similarity between neighbouring pixels*. In dense labeling tasks such as semantic segmentation, nearby pixels with similar colour intensities are likely to belong to the same class. In contrast, pixels with different colour intensities are more likely to belong to different classes. Contextual constraints like these are not explicitly modeled by CNNs.

We illustrate the importance of context for semantic segmentation and human part estimation in Fig. 1.7. Exploiting the knowledge that the airplane flies in the sky and the ship sails in the sea is crucial to discriminating between patches of the sky and the sea, because when seen in isolation these patches look similar owing to their common blue colour and uniform appearance. Similarly, a model estimating human parts using local appearance alone may confuse between the arms and the torso because the shirt has a uniform appearance. However, using knowledge of the geometry of the human body will help resolve such ambiguities. Thus context, both local and global, is an important cue to accurately classifying an image patch. We now formally introduce the notion of structured prediction which allows us to explicitly capture visual context for dense labeling tasks.



(a) Segmentation (b) Human Parts

Figure 1.7: The role of context in (a) semantic segmentation and (b) human part estimation. In (a), the goal is to assign a label (airplane, ship, sea, sky) to each pixel in the image. Structured prediction allows associating the predictions of the sky and the sea with the predictions of airplane and ship, thereby aiding resolve ambiguities in the discrimination of sea and sky patches. In (b), the goal is to assign a label (head, torso, upper arm, lower arm, upper leg, lower leg, or background) to each pixel in the image. Structured prediction considers interdependencies between the labels. This allows us to better capture the geometry of the human body, and learn a set of context-encoding rules based on the image, for example the head should be adjacent to the torso, and the arms should be on either side of the torso.

$\mathcal{I}$	Dataset of Images
I	An image from the dataset $I \in \mathcal{I}$
i	Index for the pixels in an image
P	Number of pixels in the image, thus $i \in \{1, 2, \ldots, P\}$
p_i	(r, g, b) intensities for pixel i
l_i	Label corresponding to pixel i
$\mathbf{l}$	Vector of labels for all pixels in the image, thus $\mathbf{l} = \{l_i \forall i \in I\}$
L	Number of candidate labels
u	Index for the labels, thus $u \in \{1, 2, \ldots, L\}$
$\mathcal{L}$	The set of all possible labelings
$\mathbf{x}_i(u)$ or $\mathbf{x}_i^u$	Score for assigning pixel i the label u, usually delivered by a model
$\mathbf{x}_i$	$L-$dimensional vector of scores corresponding to the pixel i
$\mathbf{x}$	Vector of scores for all the pixels in an image, $\mathbf{x} \in \mathbb{R}^{P \times L}$

Table 1.1: Notation used in this thesis.

Structured prediction [Nowozin 2011,Sutton 2012b] allows us to capture context by modeling the interdependencies between labels of the different image patches. Intuitively, structured prediction can be understood as a natural extension of the decoupled classification delivered by a sliding window classifier.

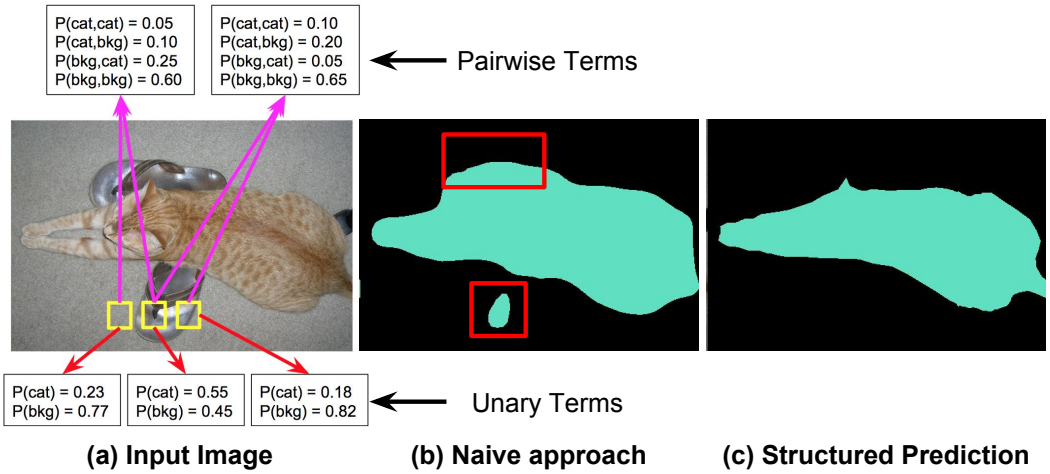


Figure 1.8: Naive approach vs Structured Prediction: The input image is shown in (a). We also show the unary terms for some patches, and pairwise terms computed for pairs of patches. The unary terms consist of two scores per patch. These two scores indicate the model's confidence in the patch belonging to 'cat' and 'background' categories when the patch is viewed in isolation. The pairwise terms are computed by looking at two patches at a time. These consist of four scores, one for each combination of classes which can be assigned to the two patches. In (b) we show the output of Deeplab Large-FOV network [Chen 2014a] on the input image, obtained by using the unary terms alone. We mark the false 'cat' predictions using red boxes. The image in (c) shows the output of our method which uses structured prediction and uses both the unary and pairwise terms to make a decision about the label taken by each patch: the model is able to eliminate isolated patches of false predictions and better capture object boundaries by capturing interdependencies between the two labels.

To better understand what this extension is and why it is desirable, we study the sliding window classifier / FCN with an example. Consider the problem of semantic segmentation in an image with two candidate classes: cat and background (Fig. 1.8). We use the notation described in Tab. 1.1. The input image containing P pixels is denoted by I, and its pixels are indexed by i, $i \in \{1, 2, \ldots, P\}$. The goal is to assign a label l_i to each pixel from the set $\{\text{cat}, \text{bkg}\}$, so we have number of labels $L = 2$. Rather than directly predicting the label l_i for each pixel, we predict two scores per pixel, $x_i(\text{cat})$ and $x_i(\text{bkg})$. The score $x_i(u)$ denotes the model's confidence in the pixel i taking the label u. These scores are referred to as unary terms (Fig. 1.8 (a)). If we assume that the labels $\mathbf{l}$ (or scores $\mathbf{x}$) of all the pixels are conditionally independent of each other, we can obtain the label l_i for each pixel simply by assigning it the label corresponding to the bigger score. This sliding window classifier can thus be expressed as a linear function of the features

computed at each pixel i as follows:

$$\begin{aligned} x_i(u) &= \underbrace{\theta_u^T \Psi(i)}_{\text{unary}} \\ l_i &= \arg\max_u \; x_i(u), \quad u \in \{\text{cat}, \text{bkg}\}. \end{aligned} \tag{1.1}$$

where $\Psi(i)$ represents features computed on an image patch centered at the pixel i, and θ_u denotes the model parameters (weights) for class u. The features $\Psi(i)$ in this example (Fig. 1.8) come from the penultimate layer of a deep-network [Chen 2014a], and the model parameters θ are the weights learnt by the linear classifier in the last layer of the network. For simplicity we omit the commonly used Softmax operator from this discussion and introduce it in Sec. 1.A.

This model is simple and efficient because it allows us to process each pixel independently. However, as seen in Fig. 1.8, by processing the pixels independently we ignore the context around the pixel which contains useful cues as to which label the pixel should assume. This results in spatially incoherent labels, seen as isolated cat patches in Fig. 1.8. This problem can be mitigated by using structured prediction.

In addition to looking at patches in isolation, structured prediction approaches further look at combinations of patches. In Fig. 1.8 (a) our structured prediction model looks at pairs of patches and estimates, for each pair, four scores corresponding to the four combinations of the labels the two patches can take. These are referred to as pairwise terms. These pairwise terms allow us to capture interdependencies between the labels taken by any pair of patches. These unary and pairwise terms are then used to jointly define a scoring function which assigns scores to all possible labelings of the image. This scoring function is often represented as a Conditional Random Field (CRF).

A CRF $(I, \mathbf{l})$ is characterized by a Gibbs distribution [Lafferty 2001], which in our example can be expressed as follows

$$\begin{aligned} p(\mathbf{l}|I) &= \frac{1}{Z_I} \exp(-E_I(\mathbf{l})) \\ E_I(\mathbf{l}) &= \underbrace{\sum_i \phi_i(l_i)}_{\text{unary}} + \underbrace{\sum_{i,j} \phi_{i,j}(l_i, l_j)}_{\text{pairwise}}; \quad i, j \in I. \end{aligned} \tag{1.2}$$

Here $p(\mathbf{l}|I)$ denotes the probability of a particular labeling $\mathbf{l}$ given the image I, and is defined as the negative exponential of the Gibbs energy $E_I(\mathbf{l})$. Z_I is the normalization constant and ensures the probabilities corresponding to all possible labelings sum to one. $\phi_i(l_i)$ denotes the unary term corresponding to the pixel i taking the label l_i. $\phi_{i,j}(l_i, l_j)$ denotes the pairwise term corresponding to pixels i, j taking the labels l_i, l_j respectively. The subscript I on E_I and Z_I in Eq. 1.2 denotes the dependence of these terms on the input image. We omit the conditioning on I in the rest of the chapter for notational convenience. As in case of the sliding

window classifier described in Eq. 1.1, the unary terms can be expressed as a linear function of the features computed at pixel i as follows:

$$\phi_i(u) = -\theta_u^T \Psi(i) \;\; u \in \{\text{cat}, \text{bkg}\}. \tag{1.3}$$

While the sliding window classifier in Eq. 1.1 uses only the unary terms, the CRF in Eq. 1.2 uses both unary and pairwise terms. Thus, as stated before, structured prediction can be understood as a natural extension of the decoupled classification delivered by a sliding window classifier.

The pairwise terms are typically hand-crafted expressions as in [Krähenbühl 2011, Zheng 2015] but they can also be discovered directly from the data via CNNs as we show in this thesis. The maximum a posteriori (MAP), or the most probable labeling of the random field is given by $\mathbf{l}^* = \arg\max_{\mathbf{l}\in\mathcal{L}} \; p(\mathbf{l}|I)$, Since the probabilities in Eq. 1.2 are defined as the negative exponential of $E(\mathbf{l})$, maximimization of the probability involves minimization of the corresponding energy. Therefore,

$$\mathbf{l}^* = \arg\min_{\mathbf{l}\in\mathcal{L}} E(\mathbf{l}). \tag{1.4}$$

While the unary terms support the presence or absence of a class based on local appearance alone, the pairwise terms allow information flow from other parts of the image to penalize incompatible combinations of classes. The use of the pairwise terms allows 'coupling' of the labels l_i for all pixels. Even though in this example we model these interdependencies by using pairwise terms only, we can further extend our scoring function in Eq. 1.2 by capturing 'higher-order' terms which depend on more than two pixels [Koller 2007, Liu 2015b, Arnab 2016].

This coupling of predictions allows us to enforce spatial constraints that ensure that the predicted labels are spatially coherent and consistent with the geometry of the scene (as discussed in Fig. 1.7). Using both unary and pairwise terms to determine the labeling results in a more accurate segmentation, as shown in Fig. 1.8 (c). In other domains, such as segmentation of human parts (Fig. 1.7), these spatial constraints can be understood as (i) the head sits on top of the torso, (ii) the arms project out from the torso, and so on. Depending on the difficulty of the problem we are trying to solve, these constraints can either come from domain knowledge, or can be directly discovered from the data.

While structured prediction clearly is a richer and more expressive approach, this richness comes at additional computational cost. The conditional independence assumption that we make in case of the sliding window model allows making predictions for each pixel in isolation. Thus the number of possible solutions is $P \times L$ (L possible solutions for each of the P pixels). For the structured prediction model described in Eq. 1.2, we make a 'global' prediction $\mathbf{l}^*$ instead, considering how the 'local' predictions l_i depend on each other (Eq. 1.4). Consequently, the number of possible solutions for the structured prediction model is L^P. The computational overhead comes from the increase in the number of possible solutions. Exhaustively

comparing the energy values of all the possible solutions is computationally prohibitive. The problem of finding the best solution out of the many possible solutions results in a combinatorial optimization problem and is referred to as 'inference'. In fact inference is the major bottleneck of structured prediction methods. Inference for general CRFs is NP-hard if no assumptions are made about the structure of the factor graph used [Cooper 1990], and even approximations are NP-hard [Roth 1996]. While efficient linear and polynomial time algorithms do exist for chains or tree structured graphs [Sutton 2012b], the applications of these specific kinds of graphs are limited. Consequently, the majority of structured prediction approaches use *approximate inference* [Sutton 2012b, Krähenbühl 2011, Chen 2015a].

The primary contribution of this thesis is therefore to propose strategies for efficient inference for structured prediction in the context of deep learning. While most recent approaches have used models which involve approximate inference, our methods (i) lend themselves to exact inference, (ii) are faster than competing approaches, (iii) use rich, CNN based expressions for pairwise terms, unlike common approaches [Krähenbühl 2011, Chen 2014a, Chen 2015b, Barron 2016], and (iv) all model parameters can be learnt directly from the data. To this end, we propose using a specific class of CRFs, called Gaussian-CRFs (G-CRFs) [Tappen 2007] which allow exact inference. We introduce G-CRFs in detail in the next section.

1.2.1 Gaussian Conditional Random Fields

We now give an overview of G-CRFs [Tappen 2007]. G-CRFs are CRFs with continuous and Gaussian prediction variables. They are a particularly convenient choice for a structured prediction method because exact inference in Gaussian models can be performed via the solution of a system of linear equations.

The simplest description of G-CRFs involves obtaining their mathematical expressions from the definintion of the multivariate Gaussian probability distribution.

Multivariate Gaussian Distribution. The multivariate Gaussian distribution, denoted by $\mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$ is a multivariate probability distribution of the following form

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right), \tag{1.5}$$

where $\mathbf{x}$ is a $N-$dimensional multivariate vector, μ is the $N-$dimensional mean, and Σ is the $N \times N$ covariance matrix. An equivalent form, referred to as the *canonical parameterization* of the Gaussian distribution [Rue 2005] is given by

$$p(\mathbf{x}) = \exp\left(-\frac{1}{2}\mathbf{x}^T \Theta \mathbf{x} + \theta^T \mathbf{x} + \alpha\right). \tag{1.6}$$

A comparison of Eq. 1.5 and Eq. 1.6 shows that $\Theta = \Sigma^{-1}$, $\theta = \Sigma^{-1}\mu$, and $\alpha = -\frac{1}{2}\left(N\log(2\pi) - \log(|\Theta|) + \theta^T \Theta^{-1}\theta\right)$ is a normalization constant which does

not depend on $\mathbf{x}$. Θ and θ are called the canonical parameters of $p(\mathbf{x})$, and Θ is also referred to as the *inverse covariance* matrix or the *precision* matrix.

As in Sec. 1.2, if we interpret Eq. 1.6 as a Gibbs distribution, the canonical form of a Gaussian distribution can be seen as a CRF of the form $p(\mathbf{x}|I) = \exp\left(-\frac{1}{2}\mathbf{x}^T\Theta_I\mathbf{x} + \theta_I^T\mathbf{x} + \alpha\right)$, where the subscript I denotes dependence of Θ, θ on the (image) data I. In the rest of the section, it is assumed that the parameters Θ, θ depend on the input, and we omit the conditioning on I for notational convenience.

We recall from Sec. 1.2 that the probability in conditional random fields is defined as the negative exponential of the corresponding energy. Therefore, the energy corresponding to the G-CRF is given by $\tilde{E}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\Theta\mathbf{x} - \theta^T\mathbf{x} - \alpha$. Since in this thesis our objective is to 'minimize' this energy to find the best solution to a labeling problem, we can ignore the normalization constant α because it does not depend on $\mathbf{x}$. Therefore, we define the energy of a G-CRF to be

$$E(\mathbf{x}) = \frac{1}{2}\underbrace{\mathbf{x}^T A\mathbf{x}}_{\text{pairwise}} - \underbrace{B^T\mathbf{x}}_{\text{unary}} \tag{1.7}$$

where we replace Θ, θ with A, B for notational convenience. We note that both the G-CRF energy in Eq. 1.7 and the CRF energy in Eq. 1.2 consist of unary and pairwise terms. This similarity is further explored in Sec. 1.A.

Even though we begin our discourse on G-CRFs as probabilistic models, in the rest of the thesis we discard the probabilistic underpinning of the G-CRF and understand G-CRF inference as an energy-based model, using it as a structured prediction module which can be used at any stage of a CNN. In Sec. 1.A we discuss how discrete CRFs with unary and pairwise terms can be modeled using the quadratic cost function in Eq. 1.7.

Inference in G-CRFs. We now discuss the problem of inference in G-CRFs. Given A and B, inference involves finding the output $\mathbf{x}$ which minimizes the energy in Eq. 1.7. We also refer to the energy function as the objective in mathematical optimization. We will use the terms energy function and objective interchangeably in the rest of the thesis.

Consider the energy (objective) function in Eq. 1.7:

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - B^T\mathbf{x}.$$

This is an example of an *unconstrained quadratic optimization* problem: it is called quadratic because it is quadratic in $\mathbf{x}$, and it is unconstrained because $\mathbf{x}$ is allowed to take continuous real values in the set $\mathbb{R}^{PL}$ without any constraints. An objective function of a quadratic form as in Eq. 1.7 has a unique global minimum if A is positive definite ($A \succ 0$) [Boyd 2004]. In this thesis, we therefore take precautions to ensure that our pairwise terms will constitute $A \succ 0$.

To find the minimum of our objective function, we set its derivative to zero. The derivative of the objective is

$$\frac{\partial}{\partial\mathbf{x}}E(\mathbf{x}) = \frac{1}{2}A^T\mathbf{x} + \frac{1}{2}A\mathbf{x} - B \tag{1.8}$$

In this thesis we will be dealing with symmetric pairwise terms, therefore A is symmetric by design, i.e. $A = A^T$. Thus, $\frac{\partial}{\partial \mathbf{x}} E(\mathbf{x}) = A\mathbf{x} - B$. Setting this to zero gives us $A\mathbf{x} = B$. Therefore, if our pairwise matrix A is positive definite, the energy function has a unique global minimum which can be obtained by the solution of a system of linear equations:

$$A\mathbf{x} = B. \tag{1.9}$$

Expressive power of G-CRFs. While a Gaussian model allows efficient inference, it is restrictive because it is always uni-modal and symmetric. However, in this thesis, all model parameters are computed (regressed) from the input data via non-linear, non-parametric CNNs which are trained in a supervised fashion. Thus, the G-CRF parameters are *conditioned on the data*, and represent different Gaussians for different image-dependent contexts. In other terms, even though Gaussian Random Fields are unimodal and as such less expressive, Gaussian *Conditional* Random Fields are unimodal *conditioned on the data*, effectively reflecting the fact that given the image one solution dominates the posterior distribution. Jancsary *et al.* [Jancsary 2012] also discuss this and make a similar conclusion. They add further that high dimensional encoding of labels (as described in Sec. 1.A) allows capturing associative as well as repulsive interactions. Associative interactions mean that the model encourages adjacent variables to take on the same labels, while repulsive interactions mean that the model encourages adjacent variables to take different labels. Common G-CRF based models [Tappen 2008], and even discrete models [Taskar 2004] suffer with the restriction that they can only expressive associative interactions. We provide empirical results in Chap. 2 and 3 which demonstrate that using G-CRFs to model the interactions between image regions helps boost performance on a variety of benchmarks.

1.3 Structured Prediction in Deep Learning

Having described G-CRFs in the previous section, we now discuss some of the relevant works which have contributed to the recent developments in the use of graphical models alongside deep learning for structured prediction.

Classification of deep structured prediction approaches. There are several characteristics of deep structured prediction approaches that can be used to group them together:

1. Complexity of spatial interactions i.e. the structure of the graphical model: while some approaches exploit simpler graphical models with only short-range interactions [Jampani 2016, Vemulapalli 2016b, Liu 2015a], i.e. interactions between an image patch and neighbouring patches, others exploit fully-connected (long-range) interactions [Chen 2014a, Zheng 2015, Lin 2016] where each image patch is connected to every other patch

method	*dense*	*end2end*	*non-parametric*	*exact*
[Chen 2014a, Chen 2015b]	✓	✗	✗	✗
[Zheng 2015]	✓	✓	✗	✗
[Liu 2015a]	✗	✓	✗	✓
[Vemulapalli 2016b]	✗	✓	✓	✗
[Vu 2015]	✗	✓	✓	✗
[Liu 2015b]	✗	✓	✓	✗
[Jampani 2016]	✗	✓	✓	✗
[Lin 2016]	✓	✗	✓	✗
[Barron 2016]	✓	✓	✗	✓
[Bratieres 2015]	✓	✗	✗	✗
[Kundu 2016]	✓	✗	✗	✗

Table 1.2: Comparison of deep structured prediction approaches in terms of whether they accommodate (i) dense connectivity, (ii) end-to-end training, (iii) use of non-parametric, CNN-based pairwise terms, and (iv) exact inference.

2. Ability to train all model parameters in an end-to-end fashion: some approaches use graphical models for structured prediction merely as a post-processing step [Chen 2014a, Lin 2016, Kundu 2016], and others allow learning of graphical model parameters in conjunction with the features in an end-to-end manner [Liu 2015a, Vemulapalli 2016b, Zheng 2015]

3. Assumptions on the form of spatial interactions: some approaches use hand-crafted expressions for pairwise terms [Chen 2014a, Zheng 2015, Barron 2016], while others allow their discovery through deep architectures [Jampani 2016, Vemulapalli 2016b]

4. Use of approximate or exact inference: while most approaches rely on approximate inference [Chen 2014a, Zheng 2015, Jampani 2016, Vemulapalli 2016b], there are some approaches which allow exact inference [Liu 2015a, Barron 2016]

In this thesis, we aim to develop an algorithm which possesses all the desirable properties: (i.) fully-connected interactions, (ii.) end-to-end training, (iii.) non-parametric potentials, and (iv.) exact inference, while ensuring (v.) fast inference. We classify the recent deep structured prediction approaches based on these characteristics in Tab. 1.2. In the rest of this section, we attempt to describe the relevant literature in a rough chronological order, grouping together works which share conceptual similarities. We will revisit some of the most relevant works in Chap. 2, 3 and 4 to discuss them in relation with our contributions in this thesis.

1.3.1 CRFs for Post Processing CNN Outputs: Dense-CRF and CNNs

We begin our discussion with methods that employ hand-crafted pairwise terms and approximate inference and use CRFs as a post processing strategy to refine the outputs delivered by a deep network. As described in Sec. 1.2, for general graphical models exact inference is intractable because the size of the solution space is exponential in the number of variables. Consequently, methods employing graphical models have to resort to using approximate solutions [Chen 2015a, Krähenbühl 2011, Chen 2014a, Vemulapalli 2016b, Vu 2015, Zheng 2015, Liu 2015b, Jampani 2016, Lin 2016]. An important work in this context is the Deeplab network from [Chen 2014a, Chen 2015b], which combined CNNs with the Dense-CRF method from [Krähenbühl 2011].

Deeplab. The Deeplab pipeline [Chen 2014a, Chen 2015b] was the first deep learning method to exploit fully-connected CRFs for structured prediction as a post processing step. This pipeline is described in Fig. 1.9. The CNN outputs are fed to a fully-connected CRF as unary terms alongside hand-crafted pairwise terms as inputs. The fully-connected CRF performs approximate inference to refine the predictions made by the CNN. They used the Dense-CRF work from [Krähenbühl 2011] for inference. We next discuss this work in detail.

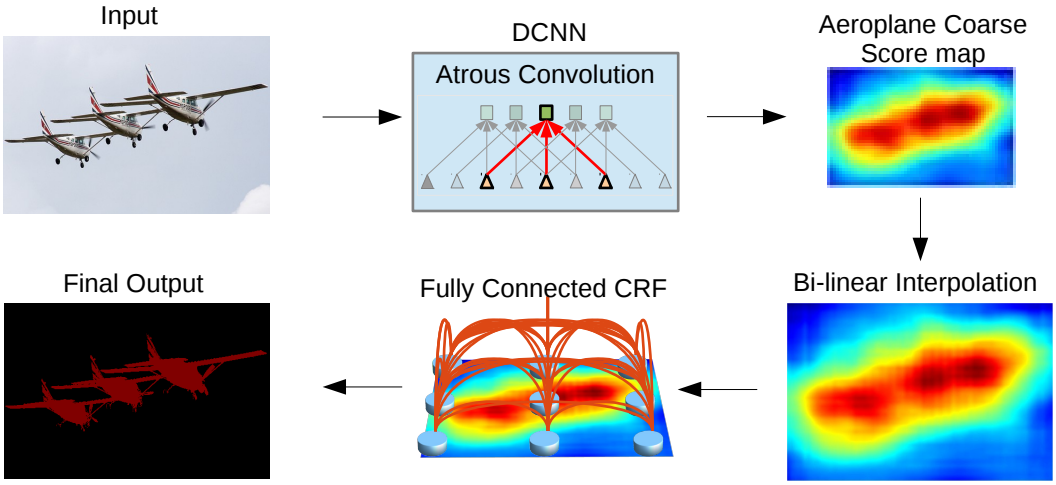


Figure 1.9: Illustration of the Deeplab [Chen 2014a] pipeline: The CNN generates unary scores, and these are upsampled to the original resolution via bilinear interpolation. These unary scores are fed to a fully-connected CRF alongside hand-crafted pairwise terms. The Dense-CRF algorithm is used as a post-processing step to obtain a sharper segmentation map which better captures finer details in the image.

Mean field inference and Dense-CRF. The Dense-CRF method of [Krähenbühl 2011] expresses the energy of a fully-connected CRF model as the sum of unary

and pairwise potentials given by

$$E_I(\mathbf{l}) = \sum_i \psi_u(l_i) + \sum_i \sum_{j<i} \psi_p(l_i, l_j), \tag{1.10}$$

where

$$\psi_p(l_i, l_j) = \\ \mu(l_i, l_j) \sum_{m=1}^{K} w_m^1 \exp(-\frac{|s_i - s_j|^2}{2\theta_\alpha^2} - \frac{|p_i - p_j|^2}{2\theta_\beta^2}) + w_m^2 \exp(-\frac{|s_i - s_j|^2}{2\theta_\gamma^2}). \tag{1.11}$$

Here $\mathbf{l} = \{l_i\}$ denotes the labels for all the pixels indexed by i coming from a set of candidate labels $l_i \in \{1, 2, \ldots, L\}$. ψ_u denotes the image dependent unary potentials, and the image dependent pairwise potentials $\psi_p(l_i, l_j)$ are expressed by the product of a label compatibility function μ and a weighted sum over Gaussian kernels. They use the pixel intensities $p_i = (r, g, b)$ and spatial positions $s_i = (\mathtt{x}, \mathtt{y})$ to define the appearance kernel, and the spatial positions alone to define the smoothness kernel. The appearance kernel tries to assign the same class labels to nearby pixels with similar colour, and the hyperparameters θ_α and θ_β control the degrees of nearness and similarity. The smoothness kernel aims to remove small isolated regions. The model parameters $(\theta_\alpha, \theta_\beta, \theta_\gamma, w_m^1, w_m^2)$ are set by doing parameter sweeps using a validation set. As described in Sec. 1.2, the Gibbs distribution corresponding to the energy in Eq. 1.10 is given by $p(\mathbf{l}|I) = \frac{1}{Z} \exp(-E_I(\mathbf{l}))$. The inference problem involves solving $\mathbf{l}^* = \arg\max_l p(l|I)$.

Rather than computing the exact distribution $p(\mathbf{l})$, the authors in [Krähenbühl 2011] propose using a mean-field approximation to the model p with a fully factorized distribution $q = \prod_i q_i(l_i)$ and solve for q by minimizing the KL divergence $KL(q||p)$. This simplifies to a fixed point equation which can be solved iteratively to update the marginal distributions q_i,

$$q_i^{t+1}(l_i) = \frac{1}{Z_I} \exp\{-\psi_u(l_i) - \underbrace{\sum_{j \neq i} \sum_{l_j} \psi_p(l_i, l_j) q_j^t(l_j)}_{\text{message passing}}\}, \tag{1.12}$$

where Z_I is a normalization coefficient.

Further, the authors realize that message passing in Eq. 1.12 can be performed using Gaussian filtering in feature space, and this observation allows them to exploit highly efficient approximations for high-dimensional filtering, reducing the complexity of message passing from quadratic in the number of pixels to linear. This results in an approximate inference algorithm for fully connected CRFs which is linear in the number of pixels and sublinear in the number of edges in the model.

The authors in [Krähenbühl 2011] justify the use of long-range connections by empirically studying the effect of variation of θ_α on the accuracy on multi-label image segmentation. They show that segmentation accuracy consistently increases as the longer-range connections are added, however in some cases these long-range

connections were shown to be propagating false information as well effectively decreasing accuracy.

We note that the pairwise terms in Eq. 1.11 are hand-crafted in that they are constrained to be Gaussian kernels in a 5−dimensional feature space given by the *(r,g,b)* colour, i.e. p_i and the `(x,y)` spatial location in the image plane, i.e. s_i. This limitation raises concerns about whether pairwise terms coming from 5−dimensional features are expressive enough. This question has inspired a number of recent works to extend this algorithm to make it more expressive. The authors in [Vineet 2013] extended the pairwise terms to include non-zero mean mixtures of Gaussians at some extra computational cost. Finally, the authors in [Campbell 2013] generalized the pairwise potential to a non-parametric model learnable from the data, via metric learning, while keeping the inference efficient.

While the approximate mean-field inference proposed in [Krähenbühl 2011] is efficient and works well in practice, it comes with no theoretical guarantees about how good or bad the approximation is. In a more recent work [Desmaison 2016], rather than using the mean-field approximation, the authors demonstrate that the high dimensional filtering approach used in [Krähenbühl 2011] can also be used to speed up linear- or quadratic- programming relaxations of the original CRF objective. These proposed relaxations have better theoretical bounds for energy minimization compared to the mean-field algorithm.

1.3.2 End-to-end Training and Approximate Inference

The use of Dense-CRF as a post processing step yielded significant improvements in performance. The next natural step was to try to learn all the model parameters via end-to-end training. Schwing *et al.* [Schwing 2015] exploited the mean-field algorithm to train semantic segmentation networks in an end-to-end manner. In an independent work, the authors of the *CRF as RNN* method from [Zheng 2015], posed the approximate mean-field CRF inference method as a Recurrent Neural Network (RNN). This formulation allowed end-to-end training of CRFs via back-propagation, using common deep learning modules, alleviating the need to post-process the output. Here the authors rephrased a single mean-field iteration (Eq. 1.12) as a sequence of common CNN modules. A simplified version of the mean-field inference in Eq. 1.12 can be performed as illustrated in Alg. 1. They perform CRF inference by the RNN forward pass amounting to 5 iterations at training and 10 at testing time. Fig. 1.10 gives a schematic overview of how the mean-field iteration is rephrased using common CNN modules.

Similarly, Jampani *et al.* [Jampani 2016] use the observation that mean-field inference can be approximated via high-dimensional filtering and devise strategies for learning pairwise terms from the data, eliminating the restriction on the pairwise potentials to be Gaussian in $(r, g, b, \mathtt{x}, \mathtt{y})$ unlike [Krähenbühl 2011, Zheng 2015]. Wang *et al.* [Wang 2016] show that inference in proximal methods can be expressed as an RNN, and show that these methods can be trained in an end-to-end fashion for tasks such as image denoising, depth refinement and optical flow estimation.

Algorithm 1 Mean-Field Inference using CNN operations [Zheng 2015]

procedure MEAN-FIELD INFERENCE
  $q_i(l_i) := \frac{1}{Z_I} \exp(-\psi_u(l_i))$ ▷ Initialize
  repeat
    $\check{q}_i^{(m)}(l) = \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) q_j(l) \; \forall m$ ▷ Message Passing
    $\widetilde{q}_i(l) = \sum_m w_m \check{q}_i^{(m)}(l)$ ▷ Weighting Filter Outputs
    $\hat{q}_i(l) = \sum_{l'} \mu(l, l') \widetilde{q}_i(l')$ ▷ Compatibility Transform
    $\bar{q}_i(l) = -\psi_u(l_i) - \hat{q}_i(l)$ ▷ Adding Unary Potentials
    $q_i = \frac{1}{Z_I} \exp(\bar{q}_i(l))$ ▷ Normalizing
  end repeat

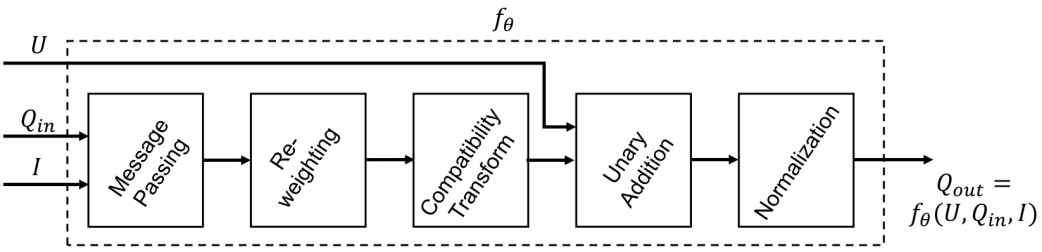


Figure 1.10: CRF-as-RNN: The authors of [Zheng 2015] reformulate a single mean-field iteration as a stack of common CNN modules. With this rephrasing, approximate CRF inference can be achieved with an RNN, allowing end-to-end learning via back-propagation.

Vemulapalli *et al.* [Vemulapalli 2016b] use G-CRFs (Sec. 1.2.1) for the task of semantic segmentation and use approximate mean-field inference for a sparsely connected graphical model. While we show in the rest of the thesis that exact yet efficient inference is feasible for G-CRFs, the authors, while acknowledging that a closed form solution to inference exists, refrain from using it citing computational challenges. We demonstrate in the following chapters how these computational challenges can be overcome.

Further, as in [Zheng 2015], the authors of the Deep Parsing Network [Liu 2015b] propose learning to approximate the mean-field inference using convolutional and pooling operations alone, and do not rely on iterative refinement of the solution. They also incorporate higher-order interactions among image regions. Independently of this work, Arnab *et al.* [Arnab 2016] enable training of end-to-end CRFs approximately with high-order interactions using the mean-field algorithm.

Lin *et al.* [Lin 2016] proposed using a two stage CRF for semantic segmentation: a coarse resolution CRF that uses unary and pairwise terms from a CNN and is trained using the approximate piecewise training method from [Sutton 2012a], and a full resolution Dense-CRF used as a post-processing method. Both of these approaches use a limited number of mean-field iterations (Deeplab uses 10 iterations while the authors in [Lin 2016] use 3), treating the inference problem as a

sequence of operations, thereby taking the pragmatic route of acknowledging and accommodating the approximation.

1.3.3 Deep Structured Prediction for Other Vision Tasks

Structured prediction has also been exploited for computer vision applications other than dense labeling. Vu *et al.* [Vu 2015] propose end-to-end training of CRFs with their network for the task of person head detection. In this work they use structured prediction to reason about the presence of multiple person heads in the image. In particular, they use a pretrained network to generate object proposals and construct a CRF with the top 16 proposals as nodes. The unary and pairwise potentials are delivered by a CNN which is trained with a structured output loss. They use the QPBO [Kolmogorov 2007] and TRW-S [Kolmogorov 2006] algorithms for approximate inference. Their approach can be visualized in Fig. 1.11.

Similarly, Chen *et al.* [Chen 2015a] use CRFs alongside CNNs for character/word recognition and image tagging in an end-to-end manner. They use an approximate message passing algorithm for inference.

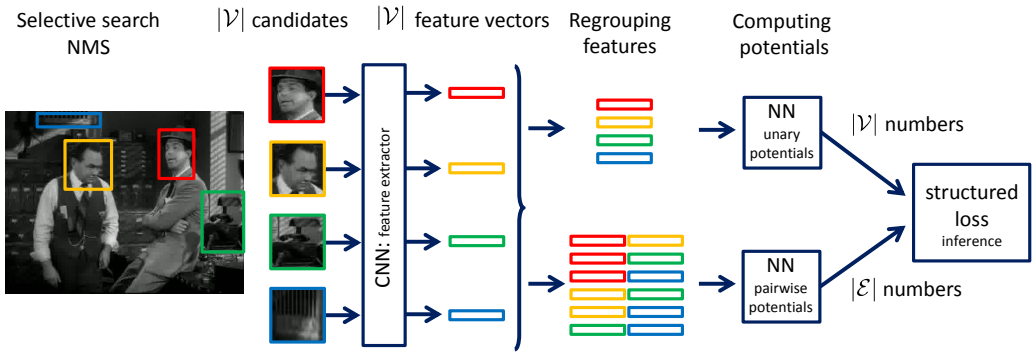


Figure 1.11: Context-aware CNNs for person head detection [Vu 2015]: Schematic visualization of the end-to-end trainable CRFs for object detection. The authors use QPBO and TRW-S for approximate CRF inference.

1.4 Contributions of this thesis

Having described the most relevant literature on structured prediction in deep learning, we now give an overview of our contributions in this thesis. The following chapters will discuss each of these in detail.

method	*dense*	*end2end*	*non-parametric*	*exact*	*temporal*
[Chen 2014a, Chen 2015b]	✓	✗	✗	✗	✗
[Zheng 2015]	✓	✓	✗	✗	✗
[Liu 2015a]	✗	✓	✗	✓	✗
[Vemulapalli 2016b]	✗	✓	✓	✗	✗
[Vu 2015]	✗	✓	✓	✗	✗
[Liu 2015b]	✗	✓	✓	✗	✗
[Jampani 2016]	✗	✓	✓	✗	✗
[Lin 2016]	✓	✗	✓	✗	✗
[Barron 2016]	✓	✓	✗	✓	✗
[Bratieres 2015]	✓	✗	✗	✗	✓
[Kundu 2016]	✓	✗	✗	✗	✓
Our contributions in this thesis					
Chap. 2	✗	✓	✓	✓	✗
Chap. 3	✓	✓	✓	✓	✗
Chap. 4	✓	✓	✓	✓	✓

Table 1.3: Comparison of deep structured prediction approaches in terms of whether they accommodate (i) dense connectivity, (ii) end-to-end training, (iii) use of non-parametric, CNN-based pairwise terms, (iv) exact inference and (v) temporal pairwise terms. We also classify our contributions in Chap. 2, Chap. 3 and Chap. 4 based on these properties.

Efficient Sparse Deep G-CRFs. (Chap. 2) Building on standard tools from numerical analysis we develop very efficient algorithms for inference and learning of deep G-CRFs for sparsely connected graphical models in FCN-based dense labeling pipelines. We derive analytical expressions to compute gradients of G-CRF parameters and use them to learn all model parameters directly from the training data. While recently published deep structured prediction approaches have resorted to inaccurate inference via mean-field approximations, our inference procedure is exact, and computes the unique global minimum of the G-CRF energy. Despite the exactness of our solution, our approach is efficient and orders of magnitude faster than contemporary structured prediction approaches. We additionally introduce multi-resolution architectures to couple contextual information across scales in a joint optimization framework. We demonstrate the utility of our approach on the challenging VOC PASCAL 2012 image segmentation benchmark, showing substantial improvements over strong baselines. Our implementation is fully GPU based, exploits sparse linear algebra optimizations using the CUDA Sparse library and is

publicly available.

Efficient Deep Fully Connected G-CRFs. (Chap. 3) We extend the deep G-CRF model to incorporate a fully-connected graph structure. Overcoming technical hurdles that accompany inference and learning over fully-connected graphical models, we keep memory and computational complexity under control by expressing the pairwise interactions as inner products of low-dimensional, learnable embeddings. This particular construction allows the G-CRF precision matrix to be low-rank, and we exploit this to develop efficient inference and learning strategies for deep fully-connected G-CRFs. We demonstrate empirically that the computational overhead when going from a sparsely-connected graphical model to a fully-connected one is negligible using our approach. As qualitative results, we show that the learned embeddings capture pixel-to-pixel affinities in a task-specific manner. As quantitative results, we demonstrate improvements in performance over sparse G-CRFs on three challenging dense labeling benchmarks, namely semantic segmentation, human part segmentation, and saliency estimation. Our results are competitive to the state of the art approaches on three different computer vision benchmarks at the time of publication.

Fully Connected G-CRFs for dense labeling in Videos. (Chap. 4) We extend the deep fully-connected G-CRF model to videos. In particular, we introduce a time- and memory-efficient method for structured prediction that couples neuron decisions across both space at time. We show that we are able to perform exact and efficient inference on a densely-connected spatio-temporal graph. We present an ablation study by experimenting with multiple connectivity patterns in the temporal domain. We present empirical improvements over strong baselines on the tasks of semantic and instance segmentation of videos. We also demonstrate improvements over the state of the art approach for the task of instance tracking.

We list the contributions of this thesis alongside other contemporary approaches for structured prediction in Tab. 1.3 and compare all methods on whether they accommodate (i.) fully-connected spatial interactions, (ii.) end-to-end training, (iii.) non-parametric potentials, (iv.) exact inference, and (v.) spatio-temporal structured prediction. Thus our contributions in this thesis combine all the desirable characterisics from Tab. 1.3. Further, our methods are also the most efficient in terms of inference speed. Inference times of our methods are available in Chapters 2, 3 and 4.

List of Publications

1. **Siddhartha Chandra**, Camille Couprie, Iasonas Kokkinos. Deep Spatio-Temporal Random Fields for Efficient Video Segmentation **CVPR 2017**

2. S. Kinauer, A. Guler, **Siddhartha Chandra**, Iasonas Kokkinos. Structured Output Prediction and Learning for Deep Monocular 3D Human Pose Estimation. **EMMCVPR 2017**

3. **Siddhartha Chandra**, N. Usunier, Iasonas Kokkinos. Dense and Low-Rank Gaussian CRFs Using Deep Embeddings. **ICCV 2017**

4. **Siddhartha Chandra**, Iasonas Kokkinos. Fast, Exact and Multi-Scale Inference for Semantic Image Segmentation with Deep Gaussian CRFs. **ECCV 2016**

5. Alp Guler, **Siddhartha Chandra**, Iasonas Kokkinos et.al. Human Joint Angle Estimation and Gesture Recognition for Assistive Robotic Vision. **Oral, ECCV Workshop 2016**

6. **Siddhartha Chandra**, S. Tsogkas, Iasonas Kokkinos. Accurate Human-Limb Segmentation in RGB-D images for Intelligent Mobility Assistance Robots. **Oral, ICCV Workshop 2015**

7. **Siddhartha Chandra**, Grigoris Chrysos, Iasonas Kokkinos. Surface Based Object Detection in RGBD Images. **Oral, BMVC 2015**

Please note that some of our contributions which were not directly related to the theme of this thesis have been left out of the present manuscript. However, all of these publications are publicly available on the website `https://siddharthachandra.github.io`. We are further planning a journal submission based on our contributions in this thesis and open sourcing all software for public use.

Appendix

1.A Modeling Discrete CRFs with Unary and Pairwise Terms using the Quadratic Cost Function.

In this section we establish a connection between the quadratic cost function used in our G-CRF formulation and discrete CRFs with unary and pairwise terms. Consider a toy labeling example with two (pixels) variable nodes p, q, each allowed to take one of 2 labels $l \in \{0, 1\}$. We denote the unary terms by b_p^l. Thus, b_p^0 is the energy/cost of assigning a label 0 to the pixel p, and so on. We denote the pairwise terms by $a_{pq}^{l_p l_q}$. Thus a_{pq}^{10} is the pairwise cost of assigning the label 1 to pixel p and 0 to pixel q. This toy problem is illustrated as a trellis graph in Fig. 1.A.1.

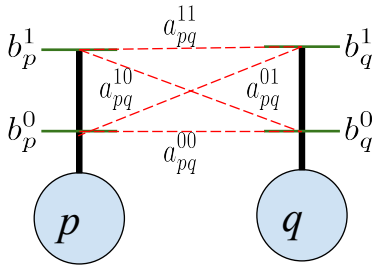


Figure 1.A.1: 2 pixels, 2 labels toy example. The green solid lines indicate unary terms, and the red dashed lines indicate pairwise terms.

Now, consider a particular labeling where p takes the label 1 and q takes the label 0. The total energy E of this labeling would then be the sum of unary costs, and the pairwise cost, i.e. $E = \left(b_p^1 + b_q^0 + a_{pq}^{10}\right)$.

Let us now express this labeling energy as a matrix equation. Consider the following matrices A and B, composed of the pairwise terms and the unary terms respectively :

$$A = \begin{bmatrix} 0 & 0 & a_{pq}^{00} & a_{pq}^{01} \\ 0 & 0 & a_{pq}^{10} & a_{pq}^{11} \\ a_{pq}^{00} & a_{pq}^{10} & 0 & 0 \\ a_{pq}^{01} & a_{pq}^{11} & 0 & 0 \end{bmatrix}; \quad B = \begin{bmatrix} -b_p^0 \\ -b_p^1 \\ -b_q^0 \\ -b_q^1 \end{bmatrix}. \tag{1.13}$$

We denote by P, L the number of variable nodes (pixels) and number of states (labels) respectively. In this case, $P = L = 2$. To understand the construction of these matrices, we first notice the sizes of A, B: $A \in \mathbb{R}^{4\times4}$ and $B \in \mathbb{R}^{4\times1}$. Since we have 2 pixels, both allowed to take one of 2 labels, we have $2 \times 2 = 4$ unary terms in B. To understand the construction of A, we note that A is a square matrix with a width of $2 \times 2 = 4$. The matrix A is constructed as $A := \{a_{p_1 p_2}^{l_1 l_2} | l_1, l_2 \in$

$\{1, \cdots, L\}, p_1, p_2 \in \{1, \cdots, P\}\}$. By virtue of this construction, A also contains terms of the form $a_{pp}^{l_1 l_2}$, i.e. pairwise terms corresponding to the same pixel taking different or same labels, but we have set them to 0, because we will not use them in the toy problem. We also notice that A is symmetric, since our pairwise terms are symmetric, i.e. the $a_{p_1 p_2}^{l_1 l_2} = a_{p_2 p_1}^{l_2 l_1}$. We will use this symmetry in all the labeling problems we study in this thesis.

In the general case, the size of matrix A is $(PL \times PL)$, and that of B is $(PL \times 1)$. As described above, the zero entries in the matrix A correspond to non-existent pairwise relationships, i.e. terms like $a_{pp}^{l_i l_j}$, and $a_{qq}^{l_i l_j}$ $\forall l_i, l_j$ etc. Now consider a labeling indicator (prediction) vector $\mathbf{x} = \left[x_p^0, x_p^1, x_q^0, x_q^1\right]^T$. The vector $\mathbf{x}$ has entries corresponding to each pairing of a pixel with a label, and the entry x_p^l indicates whether the pixel p takes the label l. For our chosen labeling, $\mathbf{x} = [0, 1, 1, 0]^T$. With this notation in place, we can now compute the G-CRF energy of the labeling E using Eq. 1.7, i.e. $E = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T B$ as,

$$E = \frac{1}{2}\begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & a_{pq}^{00} & a_{pq}^{01} \\ 0 & 0 & a_{pq}^{10} & a_{pq}^{11} \\ a_{pq}^{00} & a_{pq}^{10} & 0 & 0 \\ a_{pq}^{01} & a_{pq}^{11} & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -b_p^0 \\ -b_p^1 \\ -b_q^0 \\ -b_q^1 \end{bmatrix} = \left(b_p^1 + b_q^0 + a_{pq}^{10}\right), \tag{1.14}$$

and we arrive at the familiar expression. In this manner by constructing the unary and pairwise terms B, A, we can model dense labeling problems using the energy function in Eq. 1.7. We now note that Eq. 1.7 is equivalent to the following form

$$E(\mathbf{x}) = \frac{1}{2}\sum_{i,j} a_{ij}^{l_i l_j} x_i x_j - \sum_i b_i^{l_i} x_i; \quad i, j \in \{1, \cdots, P\}, \tag{1.15}$$

which is similar in structure to the energy function for CRFs with unary and pairwise terms as described in Eq. 1.2.

G-CRFs for discrete labeling tasks. As described in Sec. 1.1, our goal is to predict a distribution of scores $\mathbf{x}$ corresponding to each label u at each pixel i. We use x_i^u to denote the score of assigning a label u to pixel i. Each pixel is assigned the label with the maximum score, i.e. $l_i = \arg\max_u x_i^u$. The per-pixel prediction x_i is often referred to as high dimensional encoding of the discrete label l_i [Jancsary 2012]. With a high-dimensional encoding of the labels, we are no longer constrained to restrict our prediction $\mathbf{x}$ to be composed of *sparse* indicator vectors, rather we can let $\mathbf{x}$ take real values. This is advantageous because relaxing our predictions $\mathbf{x}$ to take continuous values allows us to use probabilistic loss functions such as the *softmax cross-entropy* loss to train our model parameters. Assigning a label to a pixel can then be understood as converting these scores to softmax probabilities, and assigning to each pixel the most probable label. This is described as a schematic diagram in Fig. 1.A.2.

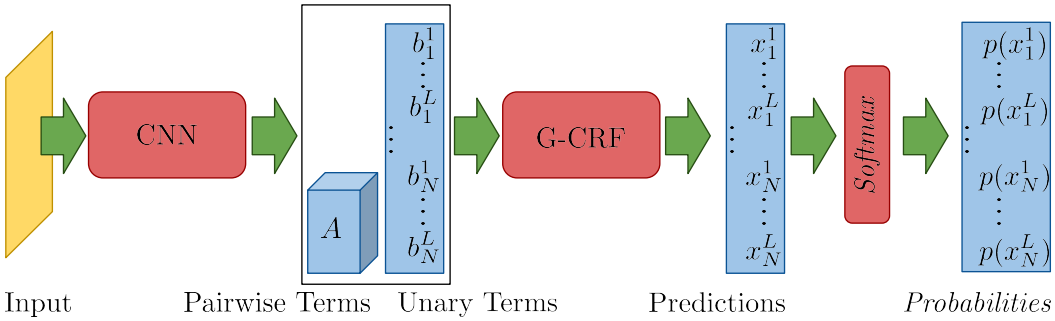


Figure 1.A.2: G-CRFs for discrete labeling tasks: The G-CRF module receives unary and pairwise terms (A) from a CNN. b_i^u denotes the unary score for pixel i taking the label u. The G-CRF module performs structured prediction and outputs the predictions: x_i^u denotes the prediction score for pixel i taking the label u. These predictions are fed as input to a Softmax module which converts these predictions to probabilities: $p(x_i^u)$ denotes the probability of pixel i taking the label u.

Chapter 2

Efficient Deep Sparse Gaussian CRFs

In this chapter we propose a structured prediction technique that combines the virtues of Gaussian Conditional Random Fields (G-CRF) with Deep Learning: (a) our structured prediction task has a unique global optimum that is obtained exactly from the solution of a linear system (b) the gradients of our model parameters are analytically computed using closed form expressions, in contrast to the memory-demanding contemporary deep structured prediction approaches [Zheng 2015, Vemulapalli 2016b] that rely on back-propagation-through-time, (c) our pairwise terms do not have to be simple hand-crafted expressions, as in the line of works building on the DenseCRF [Zheng 2015, Chen 2014a], but can rather be 'discovered' from data through deep architectures, and (d) out system can trained in an end-to-end manner. Building on standard tools from numerical analysis we develop very efficient algorithms for inference and learning, as well as a customized technique adapted to the semantic segmentation task. This efficiency allows us to explore more sophisticated architectures for structured prediction in deep learning: we introduce multi-resolution architectures to couple information across scales in a joint optimization framework, yielding systematic improvements. We demonstrate the utility of our approach on the challenging VOC PASCAL 2012 image segmentation benchmark, showing substantial improvements over strong baselines.

This work was published at the European Conference on Computer Vision (ECCV), 2016.

2.1 Introduction

Motivated by [Tappen 2007, Jancsary 2012], our starting point in this chapter is the observation that the Gaussian Conditional Random Field (G-CRF), allows us to perform exact and efficient Maximum-A-Posteriori (MAP) inference. Even though Gaussian Random Fields are unimodal and as such less expressive, Gaussian *Conditional* Random Fields are unimodal *conditioned on the data*, effectively reflecting the fact that given the image one solution dominates the posterior distribution. The G-CRF model thus allows us to construct rich expressive structured prediction models that still lend themselves to efficient inference. In particular, the log-likelihood of the G-CRF posterior has the form of a quadratic energy function which captures unary and pairwise interactions between random variables. There are two

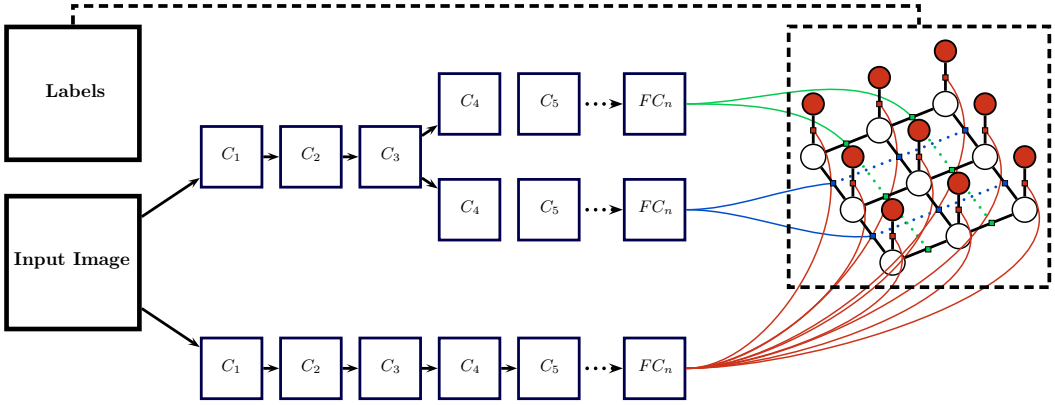
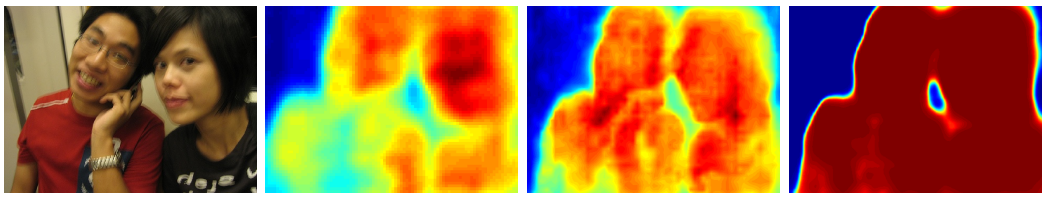


Figure 2.1: Schematic of a fully convolutional neural network with a G-CRF module: we show a detailed schematic representation of our fully convolutional neural network with a G-CRF module. The G-CRF module is shown as the box outlined by dotted lines. The factor graph inside the G-CRF module shows a 4−connected neighbourhood. The white blobs represent pixels, red blobs represent unary factors, the green and blue squares represent vertical and horizontal connectivity factors. The input image is shown in (b). The network populates the unary terms (c), and horizontal and vertical pairwise terms. The G-CRF module collects the unary and pairwise terms from the network and proposes an image hypothesis, i.e. scores (d) after inference. These scores are finally converted to probabilities using the Softmax function (e), which are then thresholded to obtain the segmentation.



(a) Input Image (b) Person unary (c) G-CRF output (d) Person Probability

Figure 2.2: G-CRF Inference: This figure compares the unary terms with the G-CRF prediction for semantic segmentation. It can be seen that the unary scores in (b) miss part of the torso because it is occluded behind the hand. The flow of information from the neighbouring region in the image, via the pairwise terms, encourages pixels in the occluded region to take the same label as the rest of the torso (c). It can also be seen that the person boundaries are more pronounced in the output (c) due to pairwise constraints between pixels corresponding to the person and background classes.

advantages to using a quadratic function: (a) unlike the energy of general graphical models, a quadratic function has a unique global minimum if the system matrix is positive definite, and (b) this unique minimum can be efficiently found by solving a system of linear equations. We can actually discard the probabilistic underpinning of the G-CRF and understand G-CRF inference as an energy-based model, casting structured prediction as sparse G-CRF (sparse G-CRF). This allows us to look at G-CRFs as generic deep learning modules which can be used at any stage in a network for structured prediction. Owing to their continuous nature, G-CRFs can also be exploited for dense regression tasks such as the estimation of depth or surface normals at each pixel in an image. Thus, G-CRFs can be seen as a computation module or a network *layer* which performs structured prediction: it models interdependencies among the output variables for any dense labeling/regression task via unary and pairwise terms. When employed in the *feature extraction* stage of a CNN, G-CRFs can be used to 'discover' mid-level features, which can better capture the spatial context. In this context, these mid-level features represent the output of a G-CRF and these naturally take continuous real values. We use G-CRFs for mid-level feature learning in Sec. 4.3.1 for the task of instance segmentation on videos. Jampani and Gehler also use CRFs for mid-level feature learning in [Jampani 2016]

Secondly, building further on the connection between MAP inference and linear system solutions, we propose memory- and time-efficient algorithms for weight-sharing (Sec. 2.3.5) and multi-scale inference (Sec. 2.4.2). In particular, in Section 2.3.5 we show that one can further reduce the memory footprint and computation demands of our method by introducing a Potts-type structure in the pairwise term. This results in multifold accelerations, while delivering results that are competitive to the ones obtained with the unconstrained pairwise term. In Sec. 2.4.2 we show that our approach allows us to work with arbitrary neighbourhoods that go beyond the common 4−connected neighbourhoods. In particular we explore the merit of using multi-scale networks, where variables computed from different image scales interact with each other. This gives rise to a flow of information across different-sized neighborhoods. We show experimentally that this yields substantially improved results over single-scale baselines.

In this chapter we focus our attention on the the image segmentation task. In Chap. 3, we will use the approach developed in this chapter for other dense labeling tasks.

In Sec. 2.2, we discuss relationship of our contributions in this chapter with some related methods. In Sec. 2.3 we describe our approach in detail, and derive the expressions for weight update rules for parameter learning that are used to train our networks in an end-to-end manner. In Sec. 2.4 we analyze the efficiency of the linear system solvers and present our multi-resolution structured prediction algorithm. In Sec. 2.5 we report consistent improvements over well-known baselines and state-of-the-art results on the VOC PASCAL test set.

2.2 Relation to Previous Works

G-CRFs were exploited for instance in the regression tree fields model of Jancsary *et al.* [Jancsary 2012] where decision trees were used to construct G-CRF's and address a host of vision tasks, including inpainting, segmentation and pose estimation. In an independent work [Vemulapalli 2016b], the authors proposed a similar approach for the task of image segmentation with CNNs, whereas in [Lin 2016,Liu 2015b,Vu 2015] FCNs are augmented with discriminatively trained convolutional layers that model and enforce pairwise consistencies between neighbouring regions.

One major difference to [Vemulapalli 2016b], as well as other prior works [Zheng 2015, Chen 2014a, Vemulapalli 2016a, Lin 2016, Liu 2015b], is that we use exact inference and do not use back-propagation-through-time during training. In particular building on the insights of [Tappen 2007, Jancsary 2012], we observe that the MAP solution, as well as the gradient of our objective with respect to the inputs of our structured prediction module can be obtained through the solution of linear systems. Casting the learning and inference tasks in terms of linear systems allows us to exploit the wealth of tools from numerical analysis. As we show in Sec. 2.4, for Gaussian CRFs sequential/parallel mean-field inference amounts to solving a linear system using the classic Gauss-Seidel/Jacobi algorithms respectively. Instead of these under-performing methods we use conjugate gradients which allow us to perform exact inference and back-propagation in a small number (typically 10) iterations, with a negligible cost (0.02s for the general case in Sec. 2.3, and 0.003s for the simplified formulation in Sec. 2.3.5) when implemented on the GPU.

A number of other recent approaches have used CRF models that lend themselves to exact inference. Barron *et al.* [Barron 2016] propose an end-to-end trainable algorithm for bilateral filtering, which can be rephrased as a G-CRF. Please note that this work was done independently and published after our work; However, while the application of this work is limited to bilateral filtering, our contributions in this thesis allow learning of data-driven pairwise terms, and therefore generalize this work.

In another work, authors in [Liu 2015a] use a G-CRF for depth estimation from a single image. Their pipeline involves computing super-pixels on the image and the super-pixels act as the nodes of their graphical model. They use data-driven unary terms and hand-crafted similarity metrics to get the pairwise terms, and use direct solvers for inference. We hypothesize that the use of direct solvers in [Liu 2015a] is feasible since the super-pixel computation keeps a check on the number of nodes in their graphical model. Further, their network takes 10 seconds per image which can be prohibitive for some applications. In fact, efficiency is one of the primary objectives of this thesis, and our methods are more than 2 orders of magnitude faster than the direct solver approach, while dealing with much bigger graphical models.

2.3 Sparse G-CRF Formulation

We now describe our approach. Consider an image $\mathcal{I}$ containing P pixels. Each pixel $p \in \{p_1, \dots, p_P\}$ can take a label $l \in \{1, \dots, L\}$. Although our objective is to assign discrete labels to the pixels, we phrase our problem as a continuous inference task. Rather than performing a discrete inference task that delivers one label per variable, we use a continuous function of the form $\mathbf{x}(p, l)$ which gives a score for each pairing of a pixel to a label. This score can be intuitively understood as being proportional to the log-odds for the pixel p taking the label l, if a 'softmax' unit is used to post-process $\mathbf{x}$.

We denote the pixel-level ground-truth labeling by a discrete valued vector $\mathbf{y} \in \mathbb{Y}^P$ where $\mathbb{Y} \in \{1, \dots, L\}$, and the inferred hypothesis by a real valued vector $\mathbf{x} \in \mathbb{R}^N$, where $N = P \times L$. Our formulation is posed as an energy minimization problem. In the following subsections, we describe the form of the energy function, the inference procedure, and the parameter learning approach, followed by some technical details pertinent to using our framework in a fully convolutional neural network. Finally, we describe a simpler formulation with pairwise weight sharing which achieves competitive performance while being substantially faster. Even though our inspiration was from the probabilistic approach to structured prediction (G-CRF), from now on we treat our structured prediction technique as a Sparse G-CRF (sparse G-CRF) module, and will refer to it as sparse G-CRF henceforth.

2.3.1 Energy of a Hypothesis

We define the energy of a hypothesis in terms of a function of the following form:

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T(A + \lambda\mathbf{I})\mathbf{x} - B\mathbf{x} \tag{2.1}$$

where A denotes the symmetric $N \times N$ matrix of pairwise terms, and B denotes the $N \times 1$ vector of unary terms. In our case, as shown in Fig. 3.2, the pairwise terms A and the unary terms B are learned from the data using a fully convolutional network. In particular and as illustrated in Fig. 3.2, A and B are the outputs of the pairwise and unary streams of our network, computed by a forward pass on the input image. These unary and pairwise terms are then combined by the sparse G-CRF module to give the final per-class scores for each pixel in the image. As we show below, during training we can easily obtain the gradients of the output with respect to the A and B terms, allowing us to train the whole network end-to-end.

Eq. 2.1 is a standard way of expressing the energy of a system with unary and pair-wise interactions among the random variables [Jancsary 2012] in a vector labeling task. We chose this function primarily because it has a unique global minimum and allows for exact inference, alleviating the need for approximate inference. Note that in order to make the matrix A strictly positive definite, we add to it λ times the Identity Matrix $\mathbf{I}$, where λ is a design parameter set empirically in the experiments.

2.3.2 Inference

Given A and B, inference involves solving for the value of $\mathbf{x}$ that minimizes the energy function in Eq. 2.1. If $(A + \lambda\mathbf{I})$ is symmetric positive definite, then $E(\mathbf{x})$ has a unique global minimum [Shewchuk 1994] at:

$$(A + \lambda\mathbf{I})\mathbf{x} = B. \tag{2.2}$$

As such, inference is exact and efficient, only involving a system of linear equations.

2.3.3 Learning Unary and Pairwise Terms

Our model parameters A and B are learned in an end-to-end fashion via the back-propagation method. In the back-propagation training paradigm each module or *layer* in the network receives the derivative of the final loss $\mathcal{L}$ with respect to its output $\mathbf{x}$, denoted by $\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$, from the layer above. $\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$ is also referred to as the gradient of $\mathbf{x}$. The module then computes the gradients of its inputs and propagates them down through the network to the layer below.

To learn the parameters A and B via back-propagation, we require the expressions of gradients of A and B, i.e. $\frac{\partial\mathcal{L}}{\partial A}$ and $\frac{\partial\mathcal{L}}{\partial B}$ respectively. We now derive these expressions. Details on these derivations can be found in the Appendix 2.A.

2.3.3.1 Derivative of Loss with respect to B

To compute the derivative of the loss with respect to B, we use the chain rule of differentiation: $\frac{\partial\mathcal{L}}{\partial\mathbf{x}} = \frac{\partial\mathcal{L}}{\partial B}\frac{\partial B}{\partial\mathbf{x}}$. Application of the chain rule yields the following closed form expression, which is a system of linear equations:

$$(A + \lambda\mathbf{I})\frac{\partial\mathcal{L}}{\partial B} = \frac{\partial\mathcal{L}}{\partial\mathbf{x}}. \tag{2.3}$$

When training a deep network, the right hand side $\frac{\partial\mathcal{L}}{\partial B}$ is delivered by the layer above, and the derivative on the left hand side is sent to the unary layer below.

2.3.3.2 Derivative of Loss with respect to A

The expression for the gradient of A is derived by using the chain rule of differentiation again: $\frac{\partial\mathcal{L}}{\partial A} = \frac{\partial\mathcal{L}}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial A}$.

Using the expression $\frac{\partial\mathbf{x}}{\partial A} = \frac{\partial}{\partial A}(A+\lambda\mathbf{I})^{-1}B$, substituting $\frac{\partial}{\partial A}(A+\lambda\mathbf{I})^{-1} = -(A+\lambda\mathbf{I})^{-T} \otimes (A+\lambda\mathbf{I})^{-1}$, and simplifying the right hand side, we arrive at the following expression:

$$\frac{\partial\mathcal{L}}{\partial A} = -\frac{\partial\mathcal{L}}{\partial B} \otimes \mathbf{x}, \tag{2.4}$$

where $\otimes$ denotes the kronecker product. Thus, the gradient of A is given by the negative of the kronecker product of the output $\mathbf{x}$ and the gradient of B.

2.3.4 Softmax Cross-Entropy Loss

Please note that while in this work we use the sparse G-CRF module as the penultimate layer of the network, followed by the softmax cross-entropy loss, it can be used at any stage in a network and not only as the final classifier. We now give the expressions for the softmax cross-entropy loss and its derivative for sake of completeness.

We denote the score of pixel i taking the label u by x_i^u. The softmax probability for the pixel i taking the label u is then given by $p_i^u = \frac{e^{x_i^u}}{\sum_{u=1}^{L} e^{x_i^u}}$. Dropping the subscript i for brevity, these probabilities are penalized by the cross-entropy loss defined as $\mathcal{L} = -\sum_u \mathbf{y}^u \log p^u$, where $\mathbf{y}^u$ is the indicator function for the ground truth label u^*, i.e. $\mathbf{y}^u = 0$ if $u \neq u^*$, and $\mathbf{y}^u = 1$ otherwise. Finally the derivative of the softmax-loss with respect to the input is given by: $\frac{\partial \mathcal{L}}{\partial x^u} = p^u - y^u$.

2.3.5 Sparse G-CRF with Shared Pairwise Terms

We now describe a simplified *sparse G-CRF* formulation with shared pairwise terms which is significantly faster in practice than the one described above. We denote by $A_{p_i,p_j}(l_i, l_j)$ the pairwise energy term for pixel p_i taking the label l_i, and pixel p_j taking the label l_j. In this section, we propose a *Potts*-type pairwise model, described by the following equation:

$$A_{p_i,p_j}(l_i, l_j) = \left\{ \begin{array}{ll} 0 & l_i = l_j \\ A_{p_i,p_j} & l_i \neq l_j. \end{array} \right\} \tag{2.5}$$

In simpler terms, unlike in the general setting, the pairwise terms here depend on whether the pixels take the same label or not, and not on the particular labels they take. Thus, the pairwise terms are *shared* by different pairs of classes. While in the general setting we learn $PL \times PL$ pairwise terms, here we learn only $P \times P$ terms. To derive the inference and gradient equations after this simplification, we rewrite our inference equation $(A + \lambda \mathbf{I})\,\mathbf{x} = B$ as,

$$\begin{bmatrix} \lambda\mathbf{I} & \hat{A} & \cdots & \hat{A} \\ \hat{A} & \lambda\mathbf{I} & \cdots & \hat{A} \\ & & \vdots & \\ \hat{A} & \hat{A} & \cdots & \lambda\mathbf{I} \end{bmatrix} \times \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_L \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_L \end{bmatrix} \tag{2.6}$$

where $\mathbf{x}_k$, denotes the vector of scores for all the pixels for the class $k \in \{1, \cdots, L\}$. The per-class unaries are denoted by $\mathbf{b}_k$, and the pairwise terms $\hat{A}$ are shared between each pair of classes. The equations that follow are derived by specializing the general inference (Eq. 2.2) and gradient equations (Eq. 2.3,2.4) to this particular setting. Following simple manipulations, the inference procedure becomes a two step process where we first compute the sum of our scores $\sum_i \mathbf{x}_i$, followed by $\mathbf{x}_k$, i.e. the scores for the class k as:

$$\left(\lambda \mathbf{I} + (L-1)\,\hat{A}\right) \sum_i \mathbf{x}_i = \sum_i \mathbf{b}_i, \quad (2.7) \qquad (\lambda \mathbf{I} - \hat{A})\mathbf{x}_k = \mathbf{b}_k - \hat{A} \sum_i \mathbf{x}_i. \quad (2.8)$$

Derivatives of the unary terms with respect to the loss are obtained by solving:

$$\left(\lambda \mathbf{I} + (L-1)\,\hat{A}\right) \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{b}_i} = \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i}, \quad (2.9) \qquad (\lambda \mathbf{I} - \hat{A}) \frac{\partial \mathcal{L}}{\partial \mathbf{b}_k} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_k} - \hat{A} \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{b}_i}. \quad (2.10)$$

Finally, the gradients of $\hat{A}$ are computed as

$$\frac{\partial \mathcal{L}}{\partial \hat{A}} = \sum_k \frac{\partial \mathcal{L}}{\partial \mathbf{b}_k} \otimes \sum_{i \neq k} \mathbf{x}_i. \quad (2.11)$$

Thus, rather than solving a system with $A \in \mathbb{R}^{PL \times PL}$, we solve $L+1$ systems with $\hat{A} \in \mathbb{R}^{P \times P}$. In our case, where $L = 21$ for 20 object classes and 1 background class, this simplification empirically reduces the inference time by a factor of 6, and the overall training time by a factor of 3. We expect even larger acceleration for the MS-COCO dataset which has 80 semantic classes. Despite this simplification, the results are competitive to the general setting as shown in Sec. 2.5.

In this section we have established that in case of Potts type pairwise terms, the system of linear equations with all variables is equivalent to solving $L+1$ smaller linear systems. The same gain in computation speed can be achieved by adapting the conjugate gradient implementation to this structure and avoiding redundant computations. We further explore this direction in and propose customizations to the conjugate gradient algorithm to this effect.

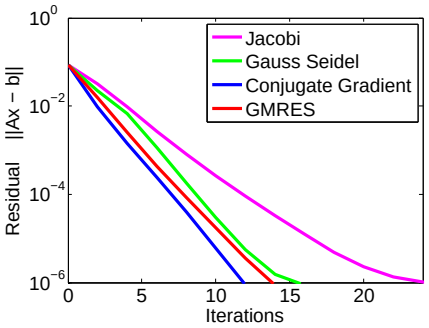
2.4 Linear Systems for Efficient Structured Prediction

Having identified that both the inference problem in Eq. 2.2 and computation of pairwise gradients in Eq. 2.3 require the solution of a linear system of equations, we now discuss methods for accelerated inference that rely on standard numerical analysis techniques for linear systems [Press 1992, Golub 1996]. Our main contributions consist in (a) using fast linear system solvers that exhibit fast convergence (Sec. 2.4.1) and (b) performing inference on multi-scale graphs by constructing block-structured linear systems (Sec. 2.4.2).

Our contributions in (a) indicate that standard conjugate gradient based linear system solvers can be up to 2.5 faster than the solutions one could get by a naive application of parallel mean-field when implemented on the GPU. Our contribution in (b) aims at accuracy rather than efficiency, and is experimentally validated in Sec. 2.5

Method	Iterations
Jacobi	24.8
Gauss Seidel	16.4
GMRES	14.8
Conjugate Gradient	13.2

(a) Linear Solver Statistics



(c) Iterative Solvers Convergence

Figure 2.1: The table in (a) shows the average number of iterations required by various algorithms, namely Jacobi, Gauss Seidel, Conjugate Gradient, and Generalized Minimal Residual (GMRES) iterative methods to converge to a residual of tolerance 10^{-6}. Figure (b) shows a plot demonstrating the convergence of these iterative solvers. The conjugate gradient method outperforms the other competitors in terms of number of iterations taken to converge.

2.4.1 Fast Linear System Solvers

The computational cost of solving the linear system of equations in Eq. 2.2 and Eq. 2.3 depends on the size of the matrix A, i.e. $N \times N$, and its sparsity pattern. In our experiments, while $N \sim 10^5$, the matrix A is quite sparse, since we deal with small 4−connected, 8−connected and 12−connected neighbourhoods. While a number of direct linear system solver methods exist, the sheer size of the system matrix A renders them prohibitive, because of large memory requirements. For large problems, a number of iterative methods exist, which require less memory, come with convergence (to a certain tolerance) guarantees under certain conditions, and can be faster than direct methods. In this work, we considered the *Jacobi, Gauss-Seidel, Conjugate Gradient, and Generalized Minimal Residual* (GMRES) methods [Press 1992], as candidates for iterative solvers. The table in Fig. 2.1 (a) shows the average number of iterations required by the aforementioned methods for solving the inference problem in Eq. 2.2. We used 25 images in this analysis, and a tolerance of 10^{-6}. Fig. 2.1 shows the convergence of these methods for one of these images. Conjugate gradients clearly stand out as being the fastest of these methods, so our following results use the conjugate gradient method. Our findings are consistent with those of Grady in [Grady 2006].

As we show below, mean-field inference for the Gaussian CRF can be understood as solving the linear system of Eq. 2.2, namely parallel mean-field amounts to using the Jacobi algorithm while sequential mean-field amounts to using the Gauss-Seidel algorithm, which are the two weakest baselines in our comparisons. This indicates that by resorting to tools for solving linear systems we have introduced faster alternatives to those suggested by mean field.

In particular the *Jacobi* and *Gauss-Seidel* methods solve a system of linear

equations $A\mathbf{x} = B$ by generating a sequence of approximate solutions $\left\{\mathbf{x}^{(k)}\right\}$, where the current solution $\mathbf{x}^{(k)}$ determines the next solution $\mathbf{x}^{(k+1)}$.

The update equation for the *Jacobi* method [Golub 1996] is given by

$$x_i^{(k+1)} \leftarrow \frac{1}{a_{ii}} \left\{ b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right\}. \tag{2.12}$$

The updates in Eq. 2.12 only use the previous solution $\mathbf{x}^{(k)}$, ignoring the most recently available information. For instance, $x_1^{(k)}$ is used in the calculation of $x_2^{(k+1)}$, even though $x_1^{(k+1)}$ is known. This allows for parallel updates for $\mathbf{x}$. In contrast, the *Gauss-Seidel* [Golub 1996] method always uses the most current estimate of x_i as given by:

$$x_i^{(k+1)} \leftarrow \frac{1}{a_{ii}} \left\{ b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right\}. \tag{2.13}$$

As in [Rue 2005], the Gaussian Markov Random Field (GMRF) in its canonical form is expressed as $\pi(\mathbf{x}) \propto \exp\left\{\frac{1}{2}\mathbf{x}^T\Theta\mathbf{x} + \theta^T\mathbf{x}\right\}$, where θ and Θ are called the canonical parameters associated with the multivariate Gaussian distribution $\pi(\mathbf{x})$. The update equation corresponding to mean-field inference is given by [Wainwright 2008],

$$\mu_i \leftarrow -\frac{1}{\Theta_{ii}} \left\{ \theta_i + \sum_{j \neq i} \Theta_{ij}\mu_j \right\}, \tag{2.14}$$

The expression in Eq. 2.14 is exactly the expression for the *Jacobi* iteration (Eq. 2.12), or the *Gauss-Seidel* iteration in Eq. 2.13 for solving the linear system $\mu = -\Theta^{-1}\theta$, depending on whether we use sequential or parallel updates.

One can thus understand sequential and parallel mean-field inference and learning algorithms as relying on weaker system solvers than the conjugate gradient-based ones we propose here. The connection is accurate for Gaussian CRFs, as in our work and [Vemulapalli 2016b], and only intuitive for Discrete CRFs used in [Zheng 2015, Chen 2014a].

2.4.2 Multiresolution Graph Architecture

We now turn to incorporating computation from multiple scales in a single system. Even though CNNs are designed to be largely scale-invariant, it has been repeatedly reported [Chen 2016b] that fusing information from a CNN operating at multiple scales can improve image labeling performance. These results have been obtained for feedforward CNNs - we consider how these could be extended to CNNs with lateral connections, as in our case. A simple way of achieving this would be to use multiple image resolutions, construct one structured prediction module per resolution, train these as disjoint networks, and average the final results. This amounts

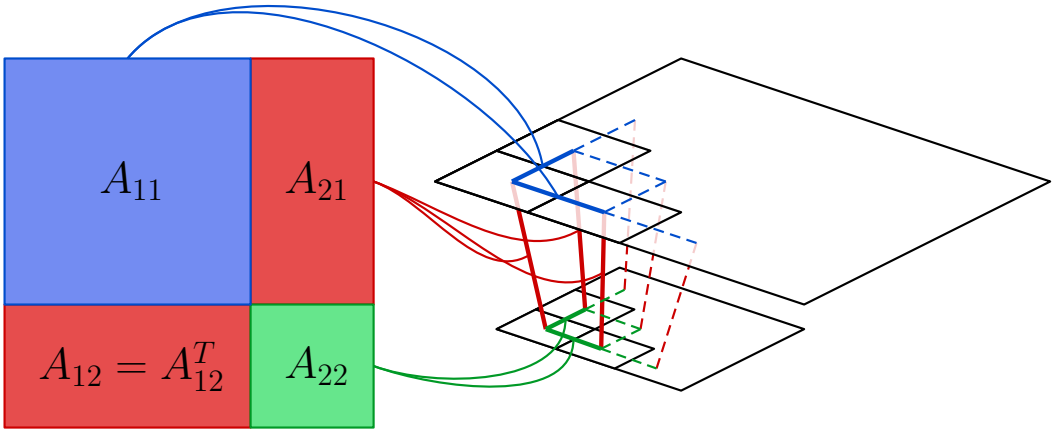


Figure 2.2: Schematic diagram of matrix A for the multi-resolution formulation in Sec. 2.4.2. In this example, we have the input image at 2 resolutions. The pairwise matrix A contains two kinds of pairwise interactions: (a) neighbourhood interactions between pixels at the same resolution (these interactions are shown as the blue and green squares), and (b) interactions between the same image region at two resolutions (these interactions are shown as red rectangles). While interactions of type (a) encourage the pixels in a neighbourhood to take the same or different label, the interactions of type (b) encourage the same image region to take the same labels at different resolutions.

to solving three decoupled systems which by itself yields a certain improvement as reported in Sec. 2.5

We advocate however a richer connectivity that couples the scale-specific systems, allowing information to flow across scales. As illustrated in Fig. 2.2 the resulting linear system captures the following multi-resolution interactions simultaneously: (a) pairwise constraints between pixels at each resolution, and (b) pairwise constraints between the same image region at two different resolutions. These inter-resolution pairwise terms connect a pixel in the image at one resolution, to the pixel it would spatially correspond to at another resolution. The inter-resolution connections help enforce a different kind of pairwise consistency: rather than encouraging pixels in a neighbourhood to have the same/different label, these encourage image regions to have the same/different labels across resolutions. This is experimentally validated in Sec. 2.5 to outperform the simpler multi-resolution architecture outlined above.

2.4.3 Implementation Details and Computational Efficiency

Our implementation is fully GPU based, and implemented using the *Caffe* library. Our network processes input images of size 865×673, and delivers results at a resolution that is 8 times smaller, as in [Chen 2014a]. The input to our sparse G-CRF modules is thus a feature map of size 109×85. Our inference procedure takes ~ 0.02s for the general setting in Sec. 2.3, and 0.003s for the simplified formulation

(Sec. 2.3.5). This is significantly faster than dense CRF postprocessing, which takes 2.4s for a 375×500 image on a CPU and the 0.24s on a GPU. Please note that Dense-CRF operates at full image resolution. Our implementation uses the highly optimized *cuBlas* and *cuSparse* libraries for linear algebra on large sparse matrices. The *cuSparse* library requires the matrices to be in the compressed-storage-row (CSR) format in order to fully optimize linear algebra for sparse matrices. Our implementation caches the indices of the CSR matrices, and as such their computation time is not taken into account in the calculations above, since their computation time is zero for streaming applications, or if the images get warped to a canonical size. In applications where images may be coming at different dimensions, considering that the indexes have been precomputed for the changing dimensions, an additional overhead of ~ 0.1s per image is incurred to read the binary files containing the cached indexes from the hard disk.

2.5 Experiments

In this section, we describe our experimental setup, network architecture and results.

Dataset. We evaluate our methods on the *VOC PASCAL 2012 image segmentation benchmark*. This benchmark uses the VOC PASCAL 2012 dataset, which consists of 1464 training and 1449 validation images with manually annotated pixel-level labels for 20 foreground object classes, and 1 background class. In addition, we exploit the additional pixel-level annotations provided by [Hariharan 2015], obtaining 10582 training images in total. The test set has 1456 unannotated images. The evaluation criterion is the pixel intersection-over-union (IOU) metric, averaged across the 21 classes.

Baseline network (basenet). Our basenet is based on the Deeplab-LargeFOV network from [Chen 2014a]. We extend it to get a multi-resolution network, which operates at three resolutions with tied weights. More precisely, our network downsamples the input image by factors of 2 and 3 and later *fuses* the downsampled activations with the original resolution via concatenation followed by convolution. The layers at three resolutions share weights. This acts like a strong baseline for a purely feedforward network. Our basenet has 49 convolutional layers, 20 pooling layers, and was pretrained on the MS-COCO 2014 trainval dataset [Len 2014]. The initial learning rate was set to 0.01 and decreased by a factor of 10 at 5K iterations. It was trained for 10K iterations.

sparse G-CRF network. We extend our basenet to accommodate the binary stream of our network. Fig. 2.1 shows a rough schematic diagram of our network. The basenet forms the unary stream of our sparse G-CRF network, while the pairwise stream is composed by concatenating the 3^{rd} pooling layers of the three resolutions followed by *batch normalization* [Ioffe 2015] and two convolutional layers. Thus, in Fig. 2.1, layers $C_1 - C_3$ are shared by the unary and pairwise streams in our experiments. Like our basenet, the sparse G-CRF networks were

Method	sparse G-CRF$_4$	sparse G-CRF$_8$	sparse G-CRF$_{12}$
IoU	76.36	76.40	76.42

Table 2.1: Connectivity

trained for 10K iterations; The initial learning rate was set to 0.01 which was decreased by a factor of 10 at 5K iterations. We consider three main types of sparse G-CRF networks: plain ($sparseG-CRF$), shared weights ($sparseG-CRF^s$) and multi-resolution ($sparseG-CRF^{mres}$).

2.5.1 Experiments using the Validation Set

In this set of experiments we train our methods on the *train+aug* images, and evaluate them on the *val* images. All our images were upscaled to an input resolution of 865×673. The hyper-parameter λ was set to 10 to ensure positive definiteness. We first study the effect of having larger neighbourhoods among image regions, thus allowing richer connectivity. More precisely, we study three kinds of connectivities: (a) 4−connected (sparse G-CRF$_4$), where each pixel is connected to its left, right, top, and bottom neighbours, (b) 8−connected (sparse G-CRF$_8$), where each pixel is additionally connected to the 4 diagonally adjacent neighbours, and (c) 12−connected (sparse G-CRF$_{12}$), where each pixel is connected to 2 left, right, top, bottom neighbours besides the diagonally adjacent ones. Table 2.1 demonstrates that while there are improvements in performance upon increasing connectivities, these are not substantial. Given that we obtain diminishing returns, rather than trying even larger neighbourhoods to improve performance, we focus on increasing the richness of the representation by incorporating information from various scales. As described in Sec. 2.4.2, there are two ways to incorporate information from multiple scales; the simplest is to have one sparse G-CRF unit per resolution ($sparseG-CRF^{res}$), thereby enforcing pairwise consistencies individually at each resolution before fusing them, while the more sophisticated one is to have information flow both within and across scales, amounting to a joint multi-scale CRF inference task, illustrated in Fig. 2.2. In Table 2.2, we compare 4 variants of our sparse G-CRF network: (a) sparse G-CRF (Sec. 2.3), (b) sparse G-CRF with shared weights (Sec. 2.3.5), (c) three sparse G-CRF units, one per image resolution, and (d) multi-resolution sparse G-CRF (Sec. 2.4.2). It can be seen that our weight sharing simplification, while being significantly faster, also gives better results than sparse G-CRF. Finally, the multi-resolution framework outperforms the other variants, indicating that having information flow both within and across scales is desirable, and a unified multi-resolution framework is better than merely averaging sparse G-CRF scores from different image resolutions.

Method	sparse G-CRF	sparse G-CRF^{s}	sparse G-CRF^{res}	sparse G-CRF^{mres}
IoU	76.36	76.59	76.69	76.93

Table 2.2: Comparison of 4 variants of our G-CRF network.

Method	IoU	IoU after *Dense CRF*
Basenet	72.72	73.78
sparse G-CRF	73.41	75.13
sparse G-CRF^{s}	73.20	75.41
sparse G-CRF^{mres}	73.86	75.46

Table 2.3: Performance of our methods on the VOC PASCAL 2012 Image Segmentation Benchmark. Our baseline network (Basenet) is a variant of Deeplab-LargeFOV [Chen 2014a] network. In this table, we demonstrate systematic improvements in performance upon the introduction of our Sparse G-CRF (sparse G-CRF), and multi-resolution (sparse G-CRF^{mres}) approaches. DenseCRF post-processing gives a consistent boost in performance.

2.5.2 Experiments using the Test Set

In this set of experiments, we train our methods on the *train+aug+val* images, and evaluate them on the *test* images. The image resolutions and λ values are the same as those in Sec. 2.5.1. In these experiments, we also use the Dense CRF post processing as in [Chen 2014a, Chen 2015b]. Our results are tabulated in Tables 2.3 and 2.4. We first compare our methods sparse G-CRF, sparse G-CRF^{s} and sparse G-CRF^{mres} with the basenet, where the relative improvements can be most clearly demonstrated. Our multi-resolution network outperforms the basenet and other sparse G-CRF networks. We achieve a further boost in performance upon using the Dense CRF post processing strategy, consistently for all methods. We observe that our method yields an improvement that is entirely complementary to the improvement obtained by combining with Dense-CRF.

We also compare our results to previously published benchmarks in Table 2.4. When benchmarking against directly comparable techniques, we observe that even though we do not use end-to-end training for the CRF module stacked on top of our sparse G-CRF network, our method outperforms the previous state of the art CRF-RNN system of [Zheng 2015] by a margin of 0.8%. We anticipate further improvements by integrating end-to-end CRF training with our sparse G-CRF. In Table 2.4, we compare our methods to previously published, directly comparable methods, namely those that use a variant of the VGG [Simonyan 2015] network, are trained in an end-to-end fashion, and use structured prediction in a fully-convolutional framework.

Method	mean IoU (%)
Deeplab-Cross-Joint [Chen 2015b]	73.9
CRFRNN [Zheng 2015]	74.7
Basenet	73.8
sparse G-CRF	75.1
sparse G-CRF^{s}	75.4
sparse G-CRF^{mres}	75.5

Table 2.4: Comparison of our method with directly comparable previously published approaches on the VOC PASCAL 2012 image segmentation benchmark.

2.5.3 Experiments with Deeplab-V2 Resnet-101

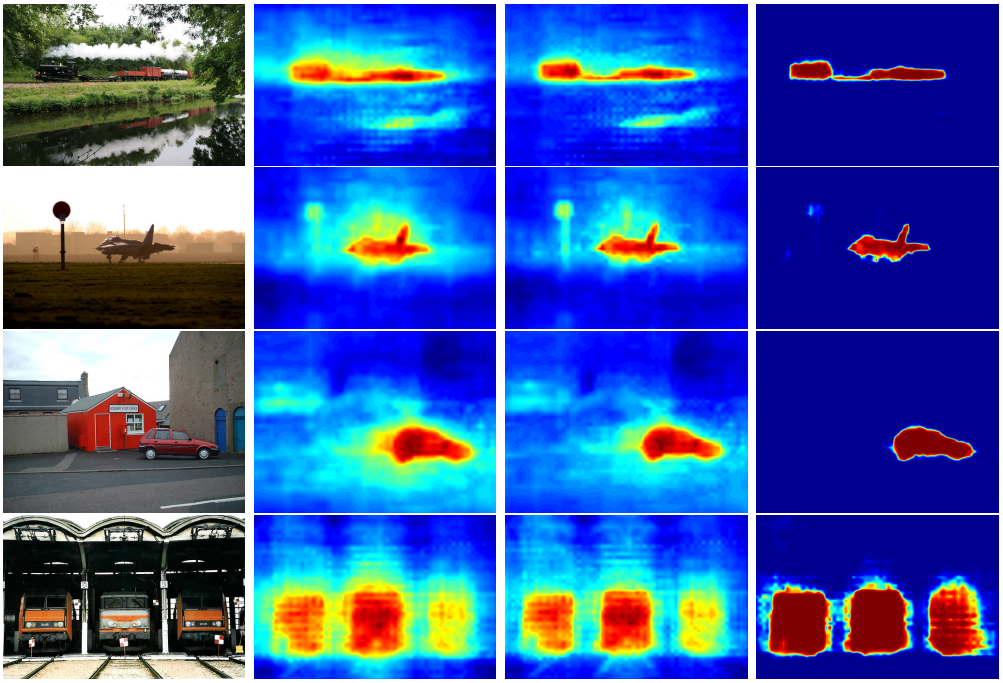
In this section we use our Potts-type model alongside the deeplab-v2 [Chen 2016a] Resnet-101 network. This network is a 3 branch multi-resolution version of the Resnet-101 network from [He 2016]. It processes the input image at 3 resolutions, with scaling factors of $0.5, 0.75$, and 1.0, and then combines the network responses at the different resolutions by upsampling the responses at the lower scales to the original scale, and taking an element-wise maximum of the responses corresponding to each pixel. We learn Potts type shared pairwise terms, and these pairwise terms are drawn from a parallel Resnet-101 network which has layers through `conv-1` to `res5c`, and processes the input image at the original scale. Table 2.5 reports quantitative results on the PASCAL VOC 2012 test set. We show some qualitative results in Fig. 2.1. It can be seen that our method refines the object boundaries, leading to a better segmentation performance.

Method	mean IoU (%)
Deeplab-v2 + CRF [Chen 2016a]	79.7
sparse G-CRF^{s}	79.5
sparse G-CRF^{s} + CRF	80.2

Table 2.5: Performance of our Potts type pairwise terms on the VOC PASCAL 2012 test set with the deeplab-v2 Resnet-101 network.

2.6 Summary

In this chapter we propose a sparse G-CRF method for deep networks which can be used for predicting continuous vector-valued variables. The inference is efficient and exact and can be solved in 0.02 seconds on the GPU for each image in the general setting, and 0.003 seconds for the Potts-type pairwise case using the conjugate gradient method. We propose a deep-learning framework which learns features and model parameters simultaneously in an end-to-end FCN training algorithm. Our

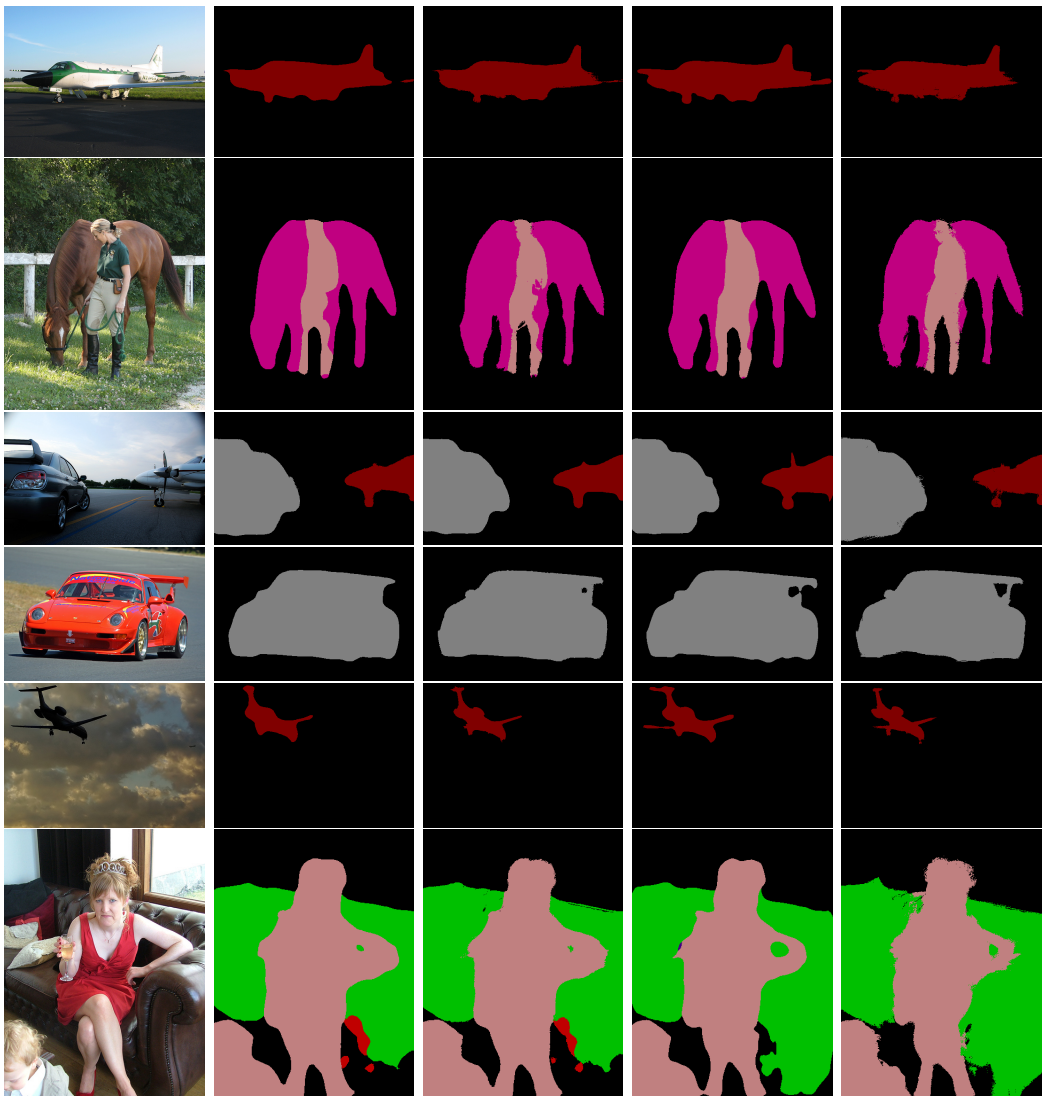


(a) image **(b) unary** **(c) output** **(d) probability**

Figure 2.1: Qualitative results when our Potts type pairwise terms are used in combination with the deeplab-V2 Resnet-101 network. Column (a) shows the input image, (b) shows the heatmap of the unary scores, (c) shows the heatmap of the scores after inference, and (d) shows the softmax probabilities. We notice that the object boundaries are significantly finer after incorporating cues from the pairwise terms.

implementation is fully GPU based, and implemented using the *Caffe* library. Our experimental results indicate that using pairwise terms boosts performance of the network on the task of image segmentation, and our results are competitive with the state of the art methods on the VOC 2012 benchmark, while being substantially simpler.

While in this chapter we focused on simple $4-12$ connected neighbourhoods, in the next chapter we extend this framework for inference on a fully-connected graphical model.



(a) Image (b) Basenet (c) Basenet + DCRF (d) sparse G-CRF^{mres} (e) sparse G-CRF^{mres} + DCRF

Figure 2.1: Visual results on the VOC PASCAL 2012 test set. The first column shows the colour image, the second column shows the basenet predicted segmentation, the third column shows the basenet output after Dense CRF post processing. The fourth column shows the sparse G-CRF^{mres} predicted segmentation, and the final column shows the sparse G-CRF^{mres} output after Dense CRF post processing. It can be seen that our multi-resolution network captures the finer details better than the basenet: the tail of the airplane in the first image, the person's body in the second image, the aircraft fan in the third image, the road between the car's tail in the fourth image, and the wings of the aircraft in the final image, all indicate this. While Dense CRF post-processing quantitatively improves performance, it tends to miss very fine details.

Appendix

In this appendix, we derive the expressions for weight-update rules for learning for the theory developed in Chap. 2. We begin by recalling the energy function of the G-CRF model.

2.A Energy of a Hypothesis

We define the energy of a hypothesis by a function of the following form:

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T(A + \lambda\mathbf{I})\mathbf{x} - B\mathbf{x} \tag{2.15}$$

where A denotes the symmetric $N \times N$ matrix of pairwise terms, and B denotes the $N \times 1$ vector of unary terms. To make the matrix A strictly positive definite, we add to it λ times the Identity Matrix $\mathbf{I}$. We can understand $E(\mathbf{x})$ as being the posterior log-likelihood of the labelling $\mathbf{x}$ according to a gaussian CRF with pairwise terms $U_{i,j}(x_i, x_j) = \exp(-\frac{1}{2}x_i(A_{i,j} + \lambda d_{i,j})x_j)$ and unary terms $V_i(x_i) = \exp(-B_i x_i)$.

2.B Inference

Given A and B, inference involves solving for the value of $\mathbf{x}$ that minimizes the energy function in 2.15. If can be shown, as in [Shewchuk 1994], that if $(A + \lambda\mathbf{I})$ is symmetric positive definite, then $E(\mathbf{x})$ has a unique global minimum at

$$(A + \lambda\mathbf{I})\mathbf{x} = B. \tag{2.16}$$

This property yields two benefits: (a) inference is exact, and (b) inference is quick, since it involves the solution of a system of linear equations.

2.C Learning Unary and Pairwise Terms

In this work, we learn model parameters A and B by stochastic gradient descent. The loss between the prediction $\mathbf{x}$ and the ground truth $\mathbf{y}$ is denoted by $\mathcal{L}(\mathbf{x}, \mathbf{y})$, and we denote the derivative of the loss with respect to the prediction by $\nabla_{\mathcal{L}}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} \equiv \nabla_{\mathcal{L}}. \tag{2.17}$$

Kindly note that while in our experiments we use the softmax loss function, in theory any differentiable loss function can be used in this formulation. For sake of completeness, we give the expressions for the softmax loss, and its derivative in the next subsection.

To learn the parameters A and B via gradient descent, we need the derivatives of the loss $\mathcal{L}(\mathbf{x}, \mathbf{y})$ with respect to A and B, i.e. $\frac{\partial \mathcal{L}}{\partial A}$ and $\frac{\partial \mathcal{L}}{\partial B}$ respectively.

2.C.1 Derivative of Loss with respect to Unary Terms

To compute the derivative of the loss with respect to B, we use the chain rule of differentiation.

$$\frac{\partial B_i}{\partial \mathbf{x}_i}\frac{\partial \mathcal{L}}{\partial B_i} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i} \tag{2.18}$$

We rewrite equation 2.18 in the matrix notation as:

$$\frac{\partial B}{\partial \mathbf{x}}\frac{\partial \mathcal{L}}{\partial B} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}}. \tag{2.19}$$

Since both B and $\mathbf{x}$ are $N-$dimensional, $\frac{\partial B}{\partial \mathbf{x}}$ is an $N \times N$ matrix. Since the loss $\mathcal{L}$ is a scalar quantity, both $\frac{\partial \mathcal{L}}{\partial B}$ and $\frac{\partial \mathcal{L}}{\partial B}$ are $N \times 1$ vectors.

Differentiating both sides of equation 2.16 with respect to $\mathbf{x}$ gives,

$$\frac{\partial B}{\partial \mathbf{x}} = (A + \lambda \mathbf{I}). \tag{2.20}$$

Substituting terms from equations 2.20 and 2.17 into 2.19, we have:

$$(A + \lambda \mathbf{I})\frac{\partial \mathcal{L}}{\partial B} = \nabla_{\mathcal{L}}. \tag{2.21}$$

Thus $\frac{\partial \mathcal{L}}{\partial B}$ can be obtained by solving the system of linear equations in 2.21.

2.C.2 Derivative of Loss with respect to Pairwise Terms

As in section 2.C.1, we use the chain rule of differentiation to express $\frac{\partial \mathcal{L}}{\partial A}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial A_{ij}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_k}\frac{\partial \mathbf{x}_k}{\partial A_{ij}}. \tag{2.22}$$

Again, we rewrite equation 2.22 in the matrix notation for brevity as follows:

$$\frac{\partial \mathcal{L}}{\partial A} = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{x}}\right)^T \frac{\partial \mathbf{x}}{\partial A}. \tag{2.23}$$

We transpose $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$ in equation 2.23 only to achieve consistency of dimensions in the matrix notation. Since A is an $N \times N$ matrix, and $\mathcal{L}$ is a scalar quantity, $\frac{\partial \mathcal{L}}{\partial A}$ is an $N \times N$ matrix. In the matrix notation, $\left(\frac{\partial \mathcal{L}}{\partial \mathbf{x}}\right)^T$ would be a $1 \times N$ row vector, while $\frac{\partial \mathbf{x}}{\partial A}$ would be an $N \times N \times N$ tensor matrix. The right hand side thus can be seen as a matrix of size $N \times N$, just like the left hand side.

We know that $(A+\lambda \mathbf{I})$ is a positive definite matrix. All positive definite matrices are invertible. We can therefore rewrite Equation 2.16 as,

$$\mathbf{x} = (A + \lambda \mathbf{I})^{-1} B. \tag{2.24}$$

We recall from [Fackler 2005], the expression for the derivative of the inverse of a matrix with respect to the matrix itself to be

$$\frac{\partial A^{-1}}{\partial A} = -A^{-T} \otimes A^{-1} \tag{2.25}$$

where $\otimes$ denotes the kronecker product of two matrices. As expected, the kronecker product has N^4 terms, because we are differentiating an $N \times N$ matrix with respect to another $N \times N$ matrix.

Using equations 2.24 and 2.25, we obtain the following expression:

$$\frac{\partial \mathbf{x}}{\partial A} = -\left((A + \lambda \mathbf{I})^{-T} \otimes (A + \lambda \mathbf{I})^{-1}\right) B. \tag{2.26}$$

Substituting terms from equations 2.26 and 2.17 into 2.23, we obtain:

$$\frac{\partial \mathcal{L}}{\partial A} = -\left(\nabla_{\mathcal{L}}\right)^T (A + \lambda \mathbf{I})^{-T} \otimes (A + \lambda \mathbf{I})^{-1} B. \tag{2.27}$$

Substituting terms from equations 2.21 and 2.24, we obtain the final expression for the derivative as follows:

$$\frac{\partial \mathcal{L}}{\partial A} = -\frac{\partial \mathcal{L}}{\partial B} \otimes \mathbf{x}. \tag{2.28}$$

This kronecker product of two $N \times 1$ vectors will have N^2 terms, which is exactly the number of terms we expect when we differentiate a scalar $\mathcal{L}(\mathbf{x}, \mathbf{y})$ with respect to the $N \times N$ matrix A. Since both matrices in the kronecker product in equation 2.28 are essentially $N-$dimensional vectors, we can express the kronecker product as a matrix product as follows:

$$\frac{\partial \mathcal{L}}{\partial A} = -\frac{\partial \mathcal{L}}{\partial B} \mathbf{x}^T. \tag{2.29}$$

2.C.3 Parameter Update Rules

Now that we know the expressions for the derivatives of the loss $\mathcal{L}$ with respect to our model parameters A and B, we use the following parameter update rules for gradient descent:

$$A = A - \eta \frac{\partial \mathcal{L}}{\partial A} - \eta \alpha A \tag{2.30}$$

$$B = B - \eta \frac{\partial \mathcal{L}}{\partial B} - \eta \alpha B \tag{2.31}$$

where η is the learning rate, and α is the weight decay hyper-parameter.

Chapter 3

Dense and Low-Rank Gaussian CRFs Using Deep Embeddings

In this chapter we introduce a structured prediction model that endows the Deep Gaussian Conditional Random Field (G-CRF) described in the previous chapter with a *fully-connected* graph structure. We keep memory and computational complexity under control by expressing the pairwise interactions as inner products of low-dimensional, learnable embeddings. The G-CRF system matrix is therefore low-rank, allowing us to solve the resulting system in a few milliseconds on the GPU by using conjugate gradient. As in the previous chapter, inference is exact, the unary and pairwise terms are jointly trained end-to-end by using analytic expressions for the gradients, while we also develop even faster, Potts-type variants of our embeddings.

We show that the learned embeddings capture pixel-to-pixel affinities in a task-specific manner, while at the time of publication our approach achieves state of the art results on three challenging benchmarks, namely semantic segmentation, human part segmentation, and saliency estimation.

This work was published at the International Conference on Computer Vision (ICCV), 2017.

3.1 Introduction

Our contribution in this chapter is the first work that combines (a) inference on fully-connected graphical models, (b) end-to-end training, (c) non-parametric pairwise terms, and (d) exact inference, while using efficient inference by relying on linear system methods. For this, we build on the sparse GCRF model (Chap. 2) which combined these advances for sparsely-connected CRFs and extend it to make the densely-connected case tractable. Figure 3.1 provides an overview of our approach. As in Chap. 2 we perform structured prediction by solving a linear system $A\mathbf{x} = B$, where A and B respectively correspond to pairwise and unary terms, delivered by an end-to-end trainable CNN. Solving this system of linear equations results in couplings among all the node variables.

The core development (Sec. 3.2) consists in replacing the sparse system matrix used to couple the labels of neighboring nodes in Chap. 2 with a low-rank matrix that connects any node with all other image nodes through inner products of learnable, D-dimensional embeddings: $A_{i,j} = \langle \mathcal{A}_i, \mathcal{A}_j \rangle$, where $i, j \in \{1, \dots, N\}$, with N

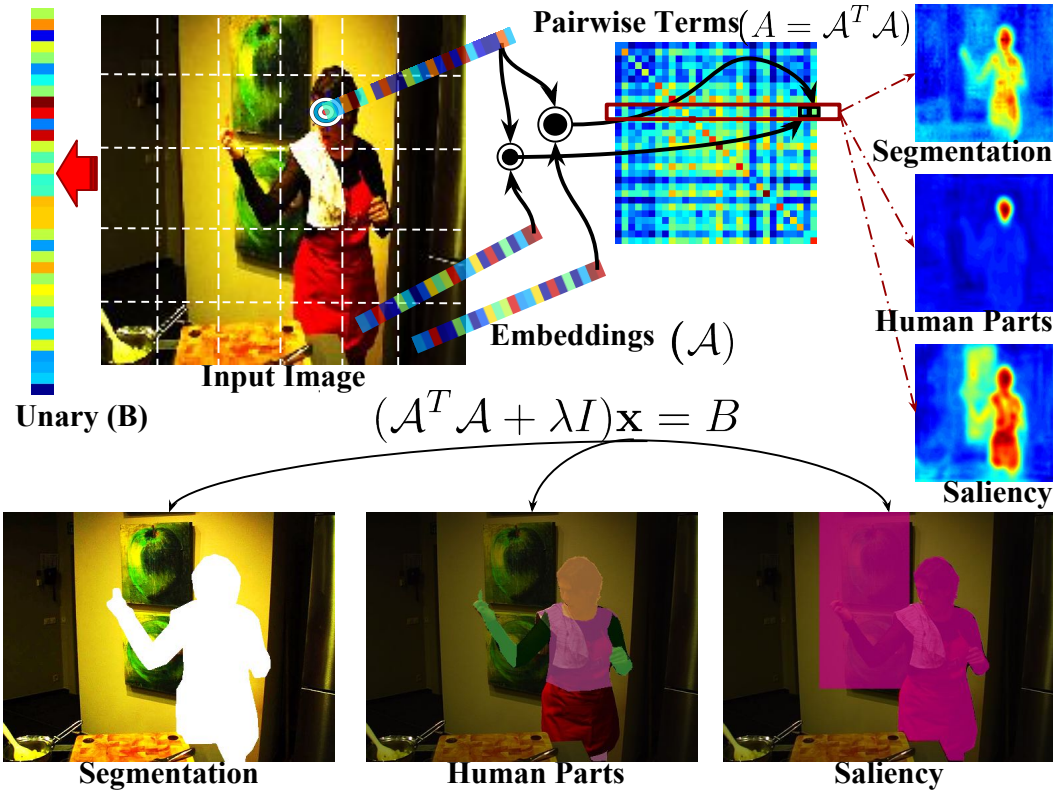


Figure 3.1: Method overview: each image patch amounts to a node in our fully-connected graph structure. As in the G-CRF model, we infer the prediction $\mathbf{x}$ by solving a system of linear equations $A\mathbf{x} = B$, based on CNN-based unary (B) and pairwise (A) terms. We express pairwise terms as dot products of low-dimensional embeddings ($A_{i,j} = \langle \mathcal{A}_i, \mathcal{A}_j \rangle$) , delivered by a devoted sub-network. This ensures that A is low-rank, allowing for efficient, conjugate gradient-based solutions. The embeddings are optimized in a task-specific manner through end-to-end training.

indexing the Cartesian product of pixels and labels. Rather than computing and inverting the full $N \times N$ matrix A, our network only needs to deliver the much smaller $N \times D$ embedding matrix $\mathcal{A}$, which is all that is needed by the conjugate gradient method. Apart from low memory complexity, this can also result in fast conjugate-gradient based structured prediction.

We note that several other works have concurrently explored the use of embeddings in the context of grouping tasks, employing them as a soft, differentiable proxy for cluster assignments [Fathi 2017, Harley 2015, Harley 2017, Newell 2016]. Ours however is the first to make the connection between embeddings, low-rank matrices and densely connected random fields, effectively training embeddings for the propagation of information across the full image domain through the solution of a linear system.

We further exploit the structure of the problem by developing Potts-type embeddings that allow us to reduce the memory complexity by L^2 and computational

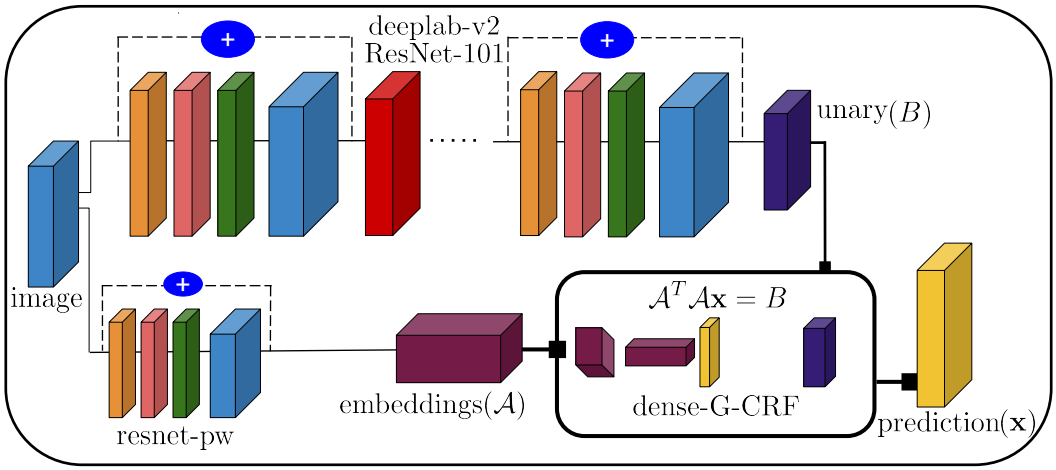


Figure 3.2: Illustration of our end-to-end trainable, fully convolutional network employing a dense-G-CRF module. We get our unary terms from Deeplab-v2 (we only show one of its three ResNet-101 branches, for simplicity). Our pairwise (pw) terms are generated by a parallel sub-network, resnet-pw, which outputs the pixel embeddings of our formulation. The unary terms and pairwise embeddings are combined by our fully connected G-CRF module (dense-G-CRF). This outputs the prediction $\mathbf{x}$ by solving the linear system $\mathcal{A}^T\mathcal{A}\mathbf{x} = B$.

complexity by a factor of L, where L is the number of classes. The computation time of our fastest method is 0.004s on a GPU for a 321×321 image, 2 orders of magnitude less than GPU-based implementations of Dense-CRF inference, while at the same time achieving higher accuracy across all tasks. Compared to the sparse Potts variant described in Chap. 2 which takes 0.003s on a GPU, this additional complexity comes at neglegible cost.

Our approach is *loss-agnostic* and works with arbitrary differentiable losses. as shown in Fig. 3.1, 3.2, and 3.3, our embeddings can learn task-specific affinities through end-to-end training. The resulting networks deliver systematic improvements when compared to strong baselines on saliency estimation, human part segmentation, and semantic segmentation.

We provide a detailed description of our approach in Sec. 3.2, and finally demonstrate the merits of our approach on three challenging tasks, namely, semantic segmentation (Sec. 3.3.1), human part segmentation (Sec. 3.3.2), and saliency estimation (Sec. 3.3.3).

3.2 Deep-and-Dense Gaussian-CRF

3.2.1 Low-Rank G-CRF through Embeddings

While the Deep G-CRF model described in Chap. 2 allows for efficient and exact inference, in practice it only captures interactions in small (4−,8− and 12−connected)

neighborhoods. The model may thereby lose some of its power by ignoring a richer set of long-range interactions. The extension to fully-connected graphs is technically challenging because of the non-sparse matrix A it involves. Assuming an image size of 800×800 pixels, 21 labels (PASCAL VOC benchmark), and a network with a spatial downsampling factor of 8 [Chen 2014a, Chen 2016a], the number of variables is $N = (100 \times 100) \times 21$ and the number of elements in A would be $N^2 \sim 10^{10}$. This is prohibitively large due to both memory and computational requirements.

To overcome this challenge, we advocate forcing A to be a *low-rank*. In particular, we propose decomposing the $N \times N$ matrix A into a product of the form

$$A = \mathcal{A}^T \mathcal{A}, \tag{3.1}$$

where $\mathcal{A}$ is a $D \times N$ matrix associating every pixel-label combination with a D-dimensional vector ('embedding'), where $D << N$. This amounts to expressing the pairwise terms for every pair of pixels and labels in the label set as the inner product of their respective embeddings, as follows:

$$A_{p_i,p_j}(l_m, l_n) = \langle \mathcal{A}_{p_i}^{l_m}, \mathcal{A}_{p_j}^{l_n} \rangle,$$

where $i, j \in \{1, \ldots, P\}$ and $m, n \in \{1, \ldots, L\}$.

Since A is symmetric and positive semi definite by design, $A' = \mathcal{A}^T\mathcal{A} + \lambda \mathbf{I}$ is positive definite for any $\lambda > 0$, unlike the case Chap. 2, where λ had to be set empirically.

Adapting the development leading to Eq. 2.2, we see that we now have to solve the system:

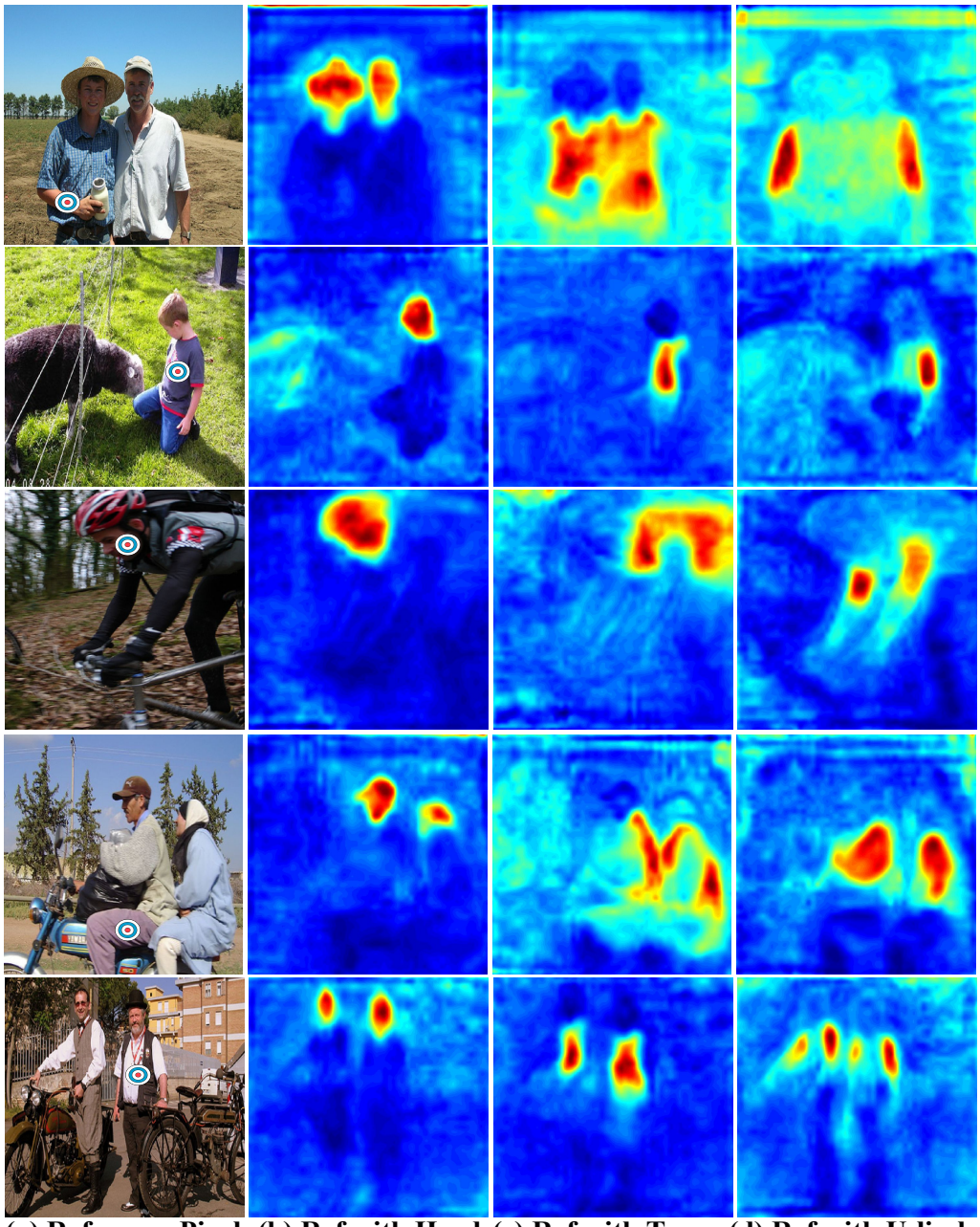
$$(\mathcal{A}^T\mathcal{A} + \lambda \mathbf{I})\mathbf{x} = B. \tag{3.2}$$

We take advantage of the positive definiteness of A' and use the conjugate gradient method [Shewchuk 1994] for solving the system of linear equations iteratively.

Setting D allows us to control both the memory and the computational complexity of inference: solving the linear system with conjugate gradient only requires keeping $\mathcal{A}$ in memory and forming inner products between $\mathcal{A}$ and a vector. As such we have a way of trading-off accuracy with speed and memory demands; as indicated in our experiments, with a sufficiently low embedding dimension we obtain excellent results.

3.2.2 Gradients of the Dense G-CRF parameters

We now turn to learning the model parameters via end-to-end network training. To achieve this we require derivatives of the overall loss $\mathcal{L}$ with respect to the model parameters, namely $\frac{\partial \mathcal{L}}{\partial \mathcal{A}}$ and $\frac{\partial \mathcal{L}}{\partial B}$. As described in Eq. 3.2, we have an analytical closed form relationship between our model parameters $\mathcal{A}$,B, and the prediction $\mathbf{x}$. Therefore, by applying the chain rule of differentiation, we can analytically express the gradients of the model parameters in terms of the gradients of the prediction.



(a) Reference Pixel **(b) Ref with Head** **(c) Ref with Torso** **(d) Ref with U-limb**

Figure 3.1: Visualization of pairwise terms obtained by our G-CRF embeddings trained for the human part segmentation. Column (a) shows the reference pixel (p^*), marked with a dartboard, on the image. The pairwise term corresponding to p^* taking the ground truth label l^* and any other pixel p taking the label l is given by the inner product $A_{p^*,p}(l^*, l) = \langle \mathcal{A}_p^l, \mathcal{A}_{p^*}^{l^*} \rangle$. We show the pairwise terms $A_{p^*,p}(l^*, \textit{head})$ in (b), $A_{p^*,p}(l^*, \textit{torso})$ in (c), and $A_{p^*,p}(l^*, \textit{upper-limb})$ in (d).

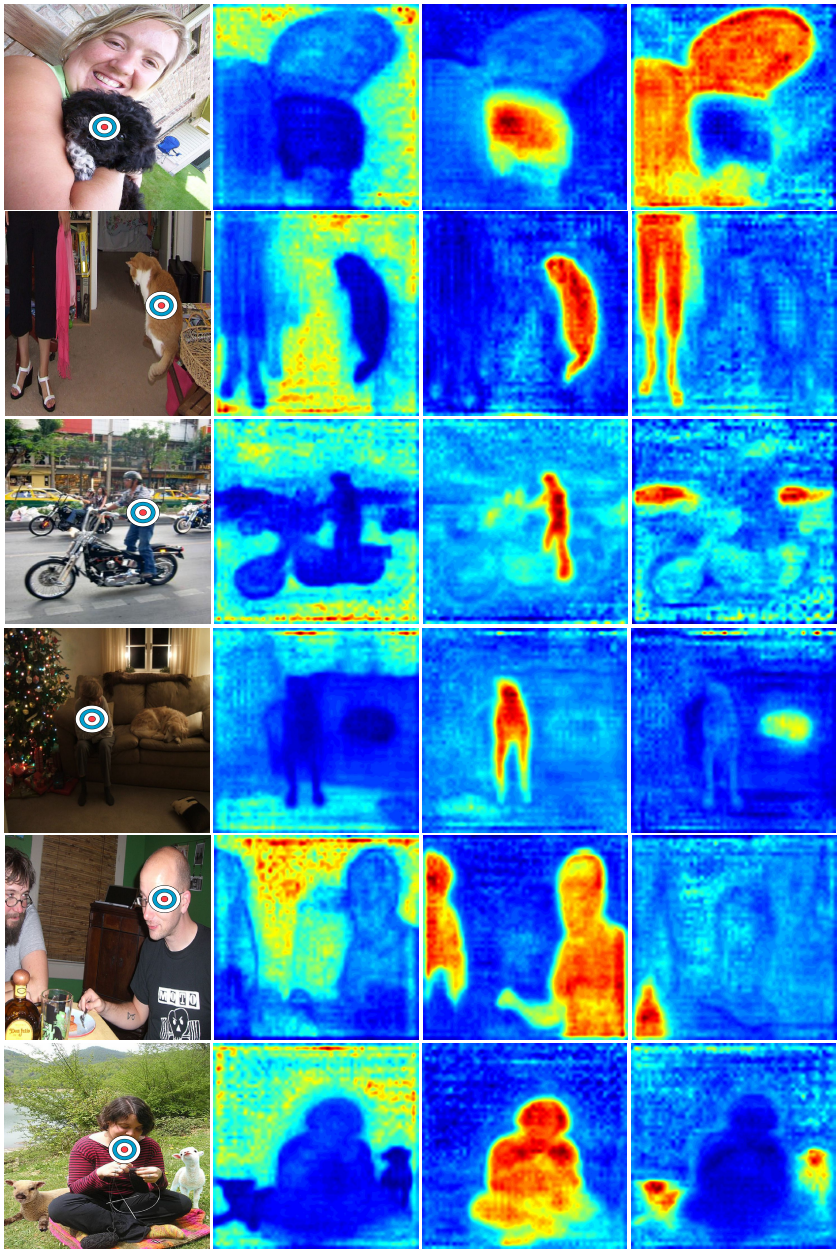


Figure 3.2: Visualization of pairwise terms obtained by our G-CRF embeddings trained for the semantic segmentation task. Column (a) shows the reference pixel (p^*), marked with a dartboard, on the image. The pairwise term corresponding to p^* taking the ground truth label l^* and any other pixel p taking the label l is given by the inner product $A_{p^*,p}(l^*, l) = \langle \mathcal{A}_p^l, \mathcal{A}_{p^*}^{l^*} \rangle$. We show the pairwise terms $A_{p^*,p}(l^*, bkg)$ in (b), $A_{p^*,p}(l^*, l^*)$ in (c), and $A_{p^*,p}(l^*, l_2)$ in (d), where l_2 is the most dominant class in the image besides l^*.

The gradients of the prediction are delivered by the neural network layer on top of our dense-G-CRF module through backpropagation.

The gradients of the unary terms are straightforward to obtain by substituting Eq. 3.1 in Eq. 2.3 as:

$$(\mathcal{A}^T\mathcal{A} + \lambda\mathbf{I})\frac{\partial\mathcal{L}}{\partial B} = \frac{\partial\mathcal{L}}{\partial\mathbf{x}}. \tag{3.3}$$

We thus obtain the gradients of the unary terms by solving a system of linear equations.

Turning to the gradients of the pixel embeddings, $\mathcal{A}$, we use the chain rule of differentiation as follows:

$$\frac{\partial\mathcal{L}}{\partial\mathcal{A}} = \left(\frac{\partial\mathcal{L}}{\partial A}\right)\left(\frac{\partial A}{\partial\mathcal{A}}\right) = \left(\frac{\partial\mathcal{L}}{\partial A}\right)\left(\frac{\partial}{\partial\mathcal{A}}\mathcal{A}^T\mathcal{A}\right). \tag{3.4}$$

We know the expression for $\frac{\partial\mathcal{L}}{\partial A}$ from Eq. 2.4, but to obtain the expression for $\frac{\partial}{\partial\mathcal{A}}\mathcal{A}^T\mathcal{A}$ we need to follow some more tedious steps. As in [Fackler 2005], we define a permutation matrix $T_{m,n}$ of size $mn \times mn$ as follows:

$$T_{m,n}\text{vec}(M) = \text{vec}(M^T), \tag{3.5}$$

where $\text{vec}(M)$ is the vectorization operator that vectorizes a matrix M by stacking its columns. When premultiplied with another matrix, $T_{m,n}$ rearranges the ordering of rows of that matrix, while when postmultiplied with another matrix, $T_{m,n}$ rearranges its columns. Using this matrix, we can form the following expression [Fackler 2005]:

$$\frac{\partial}{\partial\mathcal{A}}\mathcal{A}^T\mathcal{A} = \left(\mathbf{I}\otimes\mathcal{A}^T\right) + \left(\mathcal{A}^T\otimes\mathbf{I}\right)T_{D,N}, \tag{3.6}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix. Substituting Eq. 2.4 and Eq. 3.6 into Eq. 3.4, we obtain:

$$\frac{\partial\mathcal{L}}{\partial\mathcal{A}} = -\left(\frac{\partial\mathcal{L}}{\partial B}\otimes\mathbf{x}\right)\left(\left(\mathbf{I}\otimes\mathcal{A}^T\right) + \left(\mathcal{A}^T\otimes\mathbf{I}\right)T_{D,N}\right). \tag{3.7}$$

Despite the apparently complex form, this final expression is particularly simple to implement.

These equations allow us to train embeddings in a task-specific manner, capturing the patch-to-patch affinities that are desirable for a particular structured prediction task. We visualize the affinities learned by our embeddings in Fig. 3.1 and 3.2 - we observe that our embeddings indeed learn to group pixels in a way that is dictated by the task: on the left pixels belonging to similar human parts are grouped together, while on the right this is done for patches belonging to similar object classes. Similar results can also be seen in Fig. 3.3 for the more compact embeddings described below.

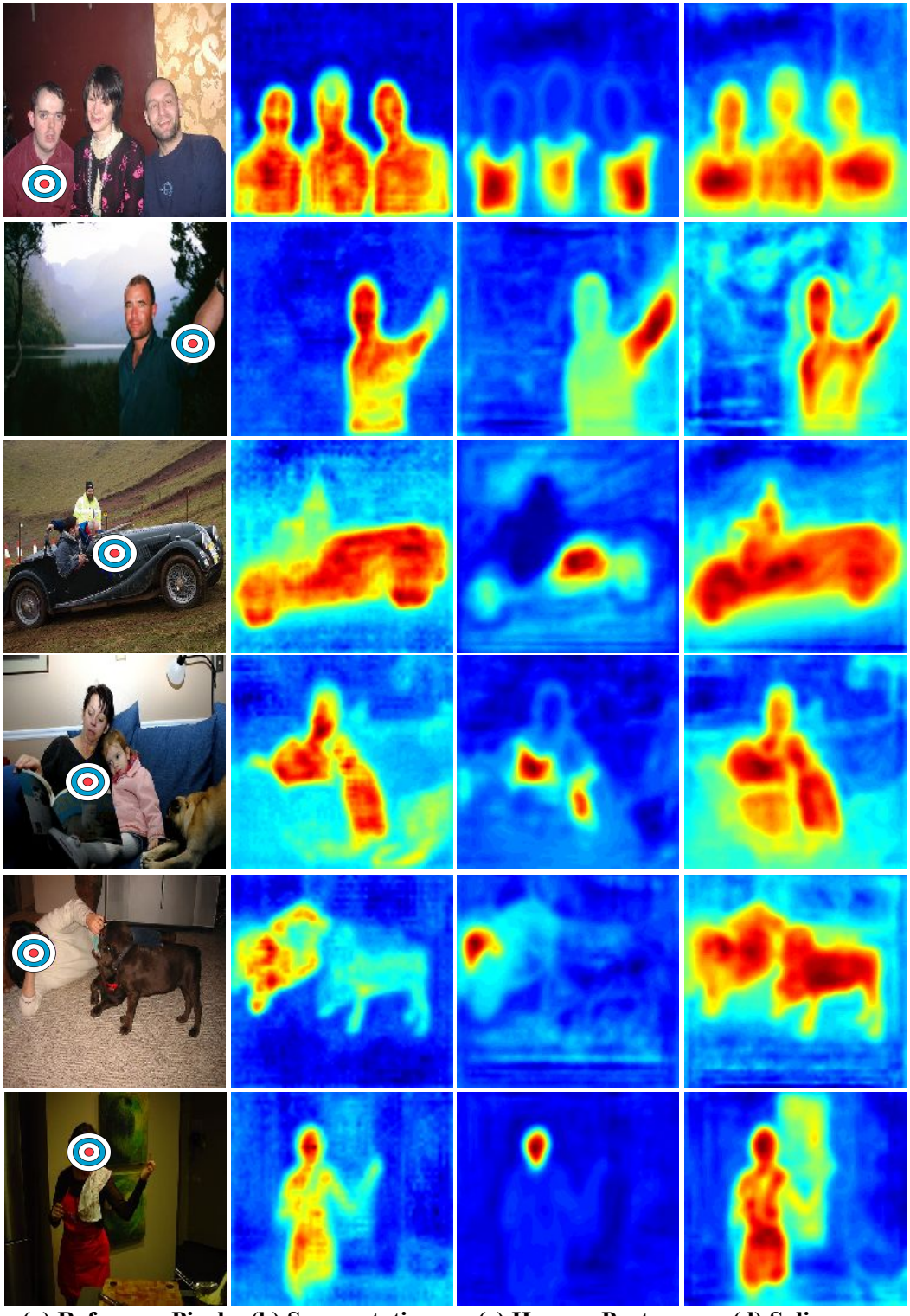


Figure 3.3: Visualization of pairwise terms obtained by our Potts-Type *task-specific* G-CRF embeddings. The first column shows the reference pixel (p^*), marked with a dartboard, on the image. The pairwise term between p^* and any other pixel p is given by the dot product $A_{p^*,p} = \mathcal{A}_p{}^T \mathcal{A}_{p^*}$. We show the pairwise terms $A_{p^*,p}$ for the (b) segmentation task, (c) human part estimation, (d) and saliency estimation.

3.2.3 Potts Type G-CRF Pixel Embeddings

We now describe *class-agnostic* G-CRF pixel embeddings, which simplify and accelerate the G-CRF model by sharing the pairwise terms between pairs of classes. More specifically, these *Potts*-type embeddings compose pairwise terms between a pair of pixels that depend only on whether they take the same label or not, and are invariant to the particular labels they take. As in Chap. 2 we denote by $A_{p_i,p_j}(l_i, l_j)$ the pairwise energy term for pixel p_i taking the label l_i, and pixel p_j taking the label l_j. The *Potts*-type embeddings describe the following model:

$$A_{p_i,p_j}(l_i, l_j) = \left\{ \begin{array}{ll} 0 & l_i = l_j \\ A_{p_i,p_j} & l_i \neq l_j. \end{array} \right\} \tag{3.8}$$

The model in Eq. 3.8 reduces the size of the embeddings from $P \times L$ to P, and allows for significantly faster inference (Sec. 3.2.4) since the number of computations are reduced by a factor of L. As demonstrated in Sec. 3.3, this leads to fewer model parameters and better performance. The *Potts*-type embeddings are realized by posing our inference problem in Eq. 3.2 as:

$$\begin{bmatrix} \lambda\mathbf{I} & \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \cdots & \hat{\mathcal{A}}^T\hat{\mathcal{A}} \\ \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \lambda\mathbf{I} & \cdots & \hat{\mathcal{A}}^T\hat{\mathcal{A}} \\ & & \vdots & \\ \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \cdots & \lambda\mathbf{I} \end{bmatrix} \times \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_L \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_L \end{bmatrix} \tag{3.9}$$

where $\mathbf{x}_k$, denotes the scores for all the pixels for the class k. The per-class unaries are denoted by $\mathbf{b}_k$, and the embeddings $\hat{\mathcal{A}}$ are shared between all class pairs. In the following subsection, we demonstrate how we can adapt the linear system solving method to better exploit the structure of the matrix A and avoid redundant computations.

3.2.4 Implementation and Efficiency

We now provide numerical analysis details that will be useful for the reproduction of our method. Our approach is implemented as a layer in *Caffe* [Jia 2014]. We exploit fast linear algebra routines of the CUDA blas library to efficiently implement the conjugate gradient method.

For these timing comparisons, we use a GTX-1080 GPU. Our general-inference procedure takes 0.029s, and Potts-type inference takes 0.004s on average for the semantic segmentation task (21 labels) for an image patch of size 321×321 pixels downsampled by a factor of 8, and for an embedding dimension of 128. This is an order of magnitude faster than the approximate dense CRF mean-field inference which takes 0.2s on average. The sparse G-CRF, and the Potts-type sparse G-CRF from Chap. 2 take 0.021s and 0.003s respectively for the same input size. Thus, our dense inference procedure comes at negligible extra cost compared to the sparse G-CRF. A more comprehensive study of inference time of our methods for

Embedding Size	Dense	DensePotts
32	0.0094	0.0017
64	0.0163	0.0019
128	0.0286	0.0037
256	0.0301	0.0039
512	0.0404	0.0043
1024	0.0509	0.0062
2048	0.1008	0.0121

Table 3.1: Average Inference Time in seconds for General and Potts Inference for varying embedding dimensions. The input image size is 321×321 pixels, and the number of labels is 21. A network downsampling factor of 8 is assumed.

Image Size	Dense	DensePotts
256×256	0.0148	0.0024
512×512	0.0328	0.0051
1024×1024	0.1056	0.0221
2048×2048	0.2703	0.0457
4096×4096	0.8407	0.1429

Table 3.2: Average Inference Time in seconds for General and Potts Inference for varying input image height/width. A network downsampling factor of 8 is assumed. The embedding size is 128, and the number of labels is 21.

varying sizes of the embedding and input image is available in Tab. 3.1 and Tab. 3.2 respectively.

We now describe our approach to efficiently implement the conjugate gradient method for G-CRF pixel embeddings. We begin by describing the conjugate gradient algorithm in Algorithm 2. The conjugate gradient algorithm poses the solution of the system of linear equations $A\mathbf{x} = B$ as finding the minimum of the energy function $\frac{1}{2}\mathbf{x}^T A\mathbf{x} - B^T\mathbf{x}$. It then solves this minimization by building a sequence of points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots\}$, iteratively progressing towards the solution starting from an initial solution $\mathbf{x}_0$. At each step, the error residual is computed and a gradient descent step is taken while ensuring that the direction of descent is orthogonal (conjugate) to all previous directions taken.

The conjugate gradient algorithm thus relies on computing the matrix-vector product $\mathbf{q} = A\mathbf{p}$ in each iteration (Algorithm 2, `line:10`). This operation is computationally the most expensive step of this method. We now describe how to efficiently compute this quantity for our case.

Conjugate Gradient for G-CRF Embeddings. To solve Eq. 3.2, each iteration of the conjugate gradient algorithm (Algorithm 2, `line:10`) involves computing $\mathbf{q} = (\mathcal{A}^T\mathcal{A} + \lambda\mathbf{I})\mathbf{p}$. Explicitly computing $(\mathcal{A}^T\mathcal{A} + \lambda\mathbf{I})$ is unnecessary because (a)

Algorithm 2 Conjugate Gradient Algorithm

procedure ConjugateGradient
  Input: $\mathbf{A}$, $\mathbf{B}$, $\mathbf{x}_0$
  Output: $\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{B}$
  $\mathbf{r}_0 := \mathbf{B} - \mathbf{A}\mathbf{x}_0$
  $\mathbf{p}_0 := \mathbf{r}_0$
  $k := 0$
  repeat
    $\alpha_k := \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
    $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
    $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
    if r_{k+1} is sufficiently small, then exit loop
    $\beta_k := \frac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
    $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
    $k := k + 1$
  end repeat
  $\mathbf{x} = \mathbf{x}_{k+1}$

it requires us to keep $PL \times PL$ terms in memory, and (b) it is computationally expensive. We therefore compute $\mathbf{q}$ as

$$\bar{\mathbf{q}} = \mathcal{A}\mathbf{p}; \quad \mathbf{q} = \mathcal{A}^T\bar{\mathbf{q}} + \lambda\mathbf{p}. \tag{3.10}$$

Conjugate Gradient for Potts-type G-CRF Embeddings. The recurring matrix-vector product for this case is given by

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_L \end{bmatrix} = \begin{bmatrix} \lambda\mathbf{I} & \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \cdots & \hat{\mathcal{A}}^T\hat{\mathcal{A}} \\ \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \lambda\mathbf{I} & \cdots & \hat{\mathcal{A}}^T\hat{\mathcal{A}} \\ & & \vdots & \\ \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \cdots & \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_L \end{bmatrix}. \tag{3.11}$$

We make two observations by carefully examining Eq. 3.11:

(1) The terms $\hat{\mathcal{A}}^T\hat{\mathcal{A}}$ are repeated $L-1$ times per column of the precision matrix. A naive implementation would compute $(\hat{\mathcal{A}}^T\hat{\mathcal{A}})\mathbf{p}_k$ exactly $L-1$ times for each class k.

(2) Each $\mathbf{q}_k$ can be computed as a sum of L terms, and for each pair $(\mathbf{q}_k, \mathbf{q}_{k'\neq k})$, $L-2$ of these terms are equal.

Using these observations, and further simplifications, we compute $\mathbf{q}_k$ for each

class as

$$\bar{\hat{\mathbf{q}}} = \hat{\mathcal{A}} \sum_{i=1}^{L} \mathbf{p}_i; \quad \hat{\mathbf{q}} = \hat{\mathcal{A}}^T \bar{\hat{\mathbf{q}}} \tag{3.12}$$

$$\bar{\hat{\mathbf{q}}}_k = \hat{\mathcal{A}} \mathbf{p}_k; \quad \mathbf{q}_k = \hat{\mathbf{q}} + \lambda \mathbf{p}_k - \hat{\mathcal{A}}^T \bar{\hat{\mathbf{q}}}_k \tag{3.13}$$

Please note that the quantity $\hat{\mathbf{q}}$ in Eq. 3.12 is computed once, and used to compute $\mathbf{q}_k$ for each class using Eq. 3.13.

3.3 Experiments and Results

Base network. Our base network is Deeplab-v2-ResNet-101 [Chen 2016a], a three branch multi-resolution network which processes the input image at scale factors of $1, 0.75, 0.5$ and then combines the network responses by upsampling the lower scales and taking an element-wise maximum. It uses random horizontal flipping, and random scaling of the input image to achieve data augmentation.

Fully-Connected G-CRF network. Our fully-connected G-CRF (dense-G-CRF) network is shown in Fig. 3.2. It uses the base network to provide unaries, and a sub-network (ResNet-pw) in parallel to the base network to construct the pixel embeddings for the pairwise terms. As dictated by our experiments in Sec. 3.3.1 the ResNet-pw has layers conv1 through res4a. We use a $3-$phase training strategy. We first train the unary network without the pairwise stream. We train the pairwise sub-network next, with the softmax cross-entropy loss to enforce the following objective: $A_{p_1,p_2}(l_1, l_2) < A_{p_1,p_2}(l_1' \neq l_1, l_2' \neq l_2)$, where l_1, l_2 are the ground truth labels for pixels p_1, p_2. Finally, we combine the unary and pairwise networks, and train them together in end-to-end fashion. Each training phase uses 20K iterations with a batch size of 10. The initial learning rate for the first two phases is fixed to 0.001, while for the third phase we set it to $2.5e^{-4}$. We use a polynomial decaying learning rate with power$=$ 0.9. Training each network takes around 2.5 days on a GTX-1080 GPU.

3.3.1 Semantic Segmentation

Dataset. We use the PASCAL VOC 2012 dataset which has 1464 training, 1449 validation and 1456 test images containing 20 foreground object classes. We also use the additional ground-truth from [Hariharan 2015], obtaining 10582 training images in total. The evaluation criterion is the mean pixel intersection-over-union (IOU) metric.

Ablation Studies. In these experiments, we train on the train set, and evaluate on the val set. We study the effect of varying the depth of the pairwise network stream by chopping the ResNet-101 at three lengths, indicated by the standard

ResNet layer names. We also study the effect of changing the size of G-CRF pixel-embeddings. These results are reported in Tab. 3.1. The best results are obtained at embedding size of 128 and 1024 for general- and Potts-type embeddings respectively. Results improve as we increase the depth of ResNet-pw. Even though the Potts-type embeddings are higher dimensional than the general embeddings, we learn less than half the parameters ($128 \times 21 = 2688 > 1024$). Improvement over the base-network is 0.91%.

Base network [Chen 2016a]	**76.30**			
dense-G-CRF	Embedding Dimension →			
ResNet-pw depth↓	64	128	256	512
res2a	76.79	76.81	76.80	76.80
res3a	76.98	76.85	76.84	76.71
res4a	76.95	**77.05**	76.95	76.97
densepotts-G-CRF	Embedding Dimension →			
ResNet-pw depth ↓	256	512	1024	2048
res2a	76.95	76.86	77.10	76.82
res3a	76.98	76.86	77.15	76.85
res4a	76.99	77.10	**77.21**	76.92

Table 3.1: Ablation study- mean Intersection Over Union (IOU) accuracy on PASCAL VOC 2012 validation set. We compare the performance of our method against that of the base network, and study the effect of varying the depth of the pairwise stream network, and the size of pixel embeddings.

Performance on test set. We now compare our approach with the base network [Chen 2016a], the base network with the sparse deep G-CRF from Chap. 2, as well as other leading approaches on this benchmark. In these experiments, we train with the train and val sets, and evaluate performance on the test set. In all of the following sections we use our best configurations from table 3.1.

Baselines. The mainstream approach on this task is to use fully convolutional networks [Chen 2014a, Chen 2016a, Long 2015] trained with the Softmax cross-entropy loss. For this task, we compare our approach with the state of the art methods on this benchmark. The baselines include (a) the CRF as RNN network [Zheng 2015], (b) the Deeplab+Boundary network [Kokkinos 2016] which exploits an edge detection detection network to boost the performance of the Deeplab network, (c) the Adelaide Context network [Lin 2016], (d) the deep parsing network [Liu 2015b], (e) the Deeplab-v2 base network [Chen 2016a] and (f) the sparse-G-CRF network (Chap. 2) which combines the Deeplab-v2 network with sparse, Potts-type pairwise terms.

We report the results in table 3.2. With our dense-Potts embeddings, we get an

improvement of 0.8% over the sparse deep G-CRF approach, and 1.3% over the base network. We get a 0.1% boost in performance when we train our dense-Potts model with the sparse G-CRF from Chap. 2 (the output after dense-GCRF inference is fed as input to the sparse-GCRF inference module). Further experiments in the rest of the section indicate that coupling sparse and dense versions of G-CRF leads to very minor improvements and may not be necessary. Qualitative results are shown in Fig. 3.1. We note that performances of two recent deep-architectures namely PSPNet [Zhao 2016] and Deeplab-v3 [Chen 2017] are significantly better than those of our baseline and other competing approaches. However, the authors of these works have not yet released their training pipelines publicly. We expect similar improvements by using our approach on these networks. We will experiment with these networks once their training pipelines are made available.

Method	mean IoU
CRFRNN [Zheng 2015]	74.7
Deeplab Multi-Scale + CRF [Kokkinos 2016]	74.8
Adelaide Context [Lin 2016]	77.8
Deep Parsing Network [Liu 2015b]	77.4
Deeplab V2 (base network) [Chen 2016a]	79.0
Deeplab V2 + CRF [Chen 2016a]	79.7
sparsepotts-G-CRF (Chap. 2)	79.5
dense-G-CRF	80.1
densepotts-G-CRF	80.3
densepotts+sparsepotts-G-CRF	**80.4**

Table 3.2: Semantic segmentation - mean Intersection Over Union (IOU) accuracy on PASCAL VOC 2012 test.

3.3.2 Human Part Segmentation

Dataset. We use the PASCAL Person Parts dataset [Chen 2014b]. As in [Liang 2016a], we merge the annotations to obtain six person part classes, namely the head, torso, upper arms, lower arms, upper legs, and lower legs. This dataset has 1716 train images and 1817 test images. The evaluation criterion is the mean pixel intersection-over-union (IOU) metric.

Baselines. The state of the art approaches on human part segmentation also use fully convolutional networks, sometimes additionally exploiting Long Short Term Memory Units [Liang 2016a, Liang 2016b]. For this task, we compare our approach to the following methods: (a) the Deeplab attention to scale network [Chen 2016b], (b) the Auto Zoom network [Xia 2016], (c) the Local Global LSTM network [Liang 2016b] which combines local and global cues via LSTM units,

(d) the Graph LSTM network [Liang 2016a], (e) the base network with and without dense CRF post-processing, and (f) the sparse G-CRF Potts model.

We report the results in table 3.3. While the previous state of the art approach Deeplab-v2 achieves 64.94 with Dense-CRF post-processing, out Potts-type model outperforms it by 1.33% mean IoU without using Dense-CRF post-processing. Additionally, we outperform the Deeplab-V2 G-CRF Potts baseline from Chap. 2 by 1.06%. Using the sparse-G-CRF on top of our results gives us a minor boost of 0.04%. We show qualitative results in Fig. 3.2.

Attention [Chen 2016b]	56.39
Auto Zoom [Xia 2016]	57.54
LG-LSTM [Liang 2016b]	57.97
Graph LSTM [Liang 2016a]	60.16
Deeplab V2 (base network) [Chen 2016a]	64.40
Deeplab V2 + CRF [Chen 2016a]	64.94
sparsepotts-G-CRF (Chap. 2)	65.21
dense-G-CRF	66.10
densepotts-G-CRF	66.27
densepotts+sparsepotts-G-CRF	**66.31**

Table 3.3: Part segmentation - mean Intersection-Over-Union accuracy on the PASCAL Parts dataset of [Chen 2014b].

Method	PASCAL-S	HKU-IS
LEGS [Wang 2015b]	0.752	0.770
MC [Zhao 2015]	0.740	0.798
MDF [Li 2015]	0.764	0.861
FCN [Li 2016]	0.793	0.867
DCL [Li 2016]	0.815	0.892
DCL + CRF [Li 2016]	0.822	0.904
Ubernet 1-Task [Kokkinos 2017]	0.835	-
Deeplab V2 (base network) [Chen 2016a]	0.859	0.916
sparse-G-CRF (Chap. 2)	0.861	0.914
dense-G-CRF	**0.872**	**0.927**
dense+sparse-G-CRF	0.864	0.927

Table 3.4: Saliency estimation results: we report the Maximal F-measure (MF) on the PASCAL Saliency dataset of [Li 2014], and the HKU-IS dataset of [Li 2015].

3.3.3 Saliency Estimation

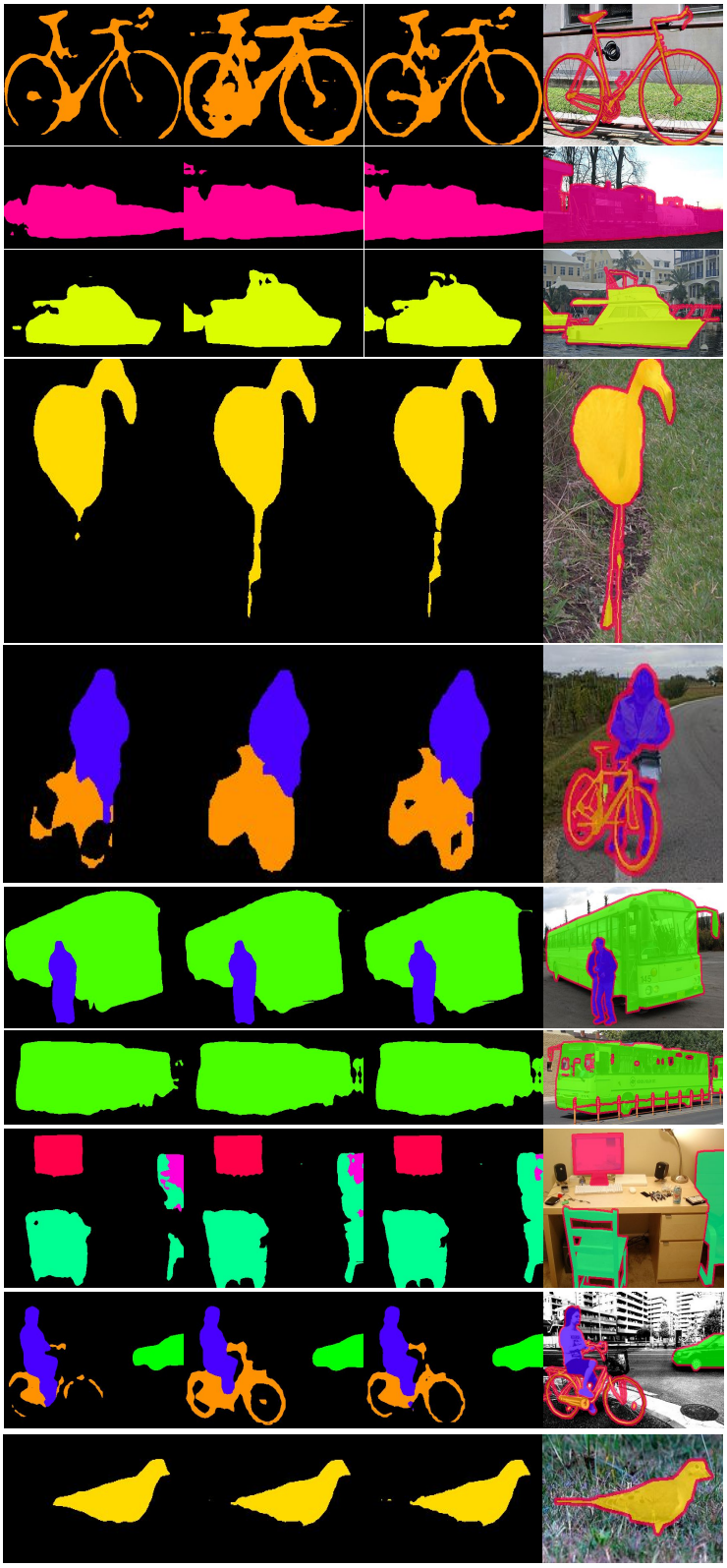
Datasets. As in [Kokkinos 2017], we use the MSRA-10K saliency dataset [Wang 2015a] for training, and evaluate our performance on the PASCAL-S [Li 2014], and the HKU-IS [Li 2015] datasets. The MSRA-10K dataset contains 10000 images with annotated pixel-wise segmentation masks for salient objects. The Pascal-S saliency dataset contains pixel-wise saliency for 850 images. The HKU-IS dataset has 4447 images, with multiple salient objects in each image. The evaluation criterion is the maximal F-Measure as in [Kokkinos 2017, Li 2014].

Baselines. Our baselines for the saliency estimation task include (a) the Local Estimation and Global Search (LEGS) framework [Wang 2015b], (b) the multi-context network [Zhao 2015], (c) the multiscale deep features network [Li 2015], (d) the deep contrast learning networks [Li 2016] which proposes a network structure that better exploits object boundaries to improve saliency estimation and additionally uses a fully connected CRF model, (e) the Ubernet architecture [Kokkinos 2017] which demonstrates that sharing parameters for mutually symbiotic tasks can help improve overall performance of these tasks, (f) our base network, i.e. Deeplab-v2, and (g) the sparse G-CRF Potts model alongside the base network.

Results are tabulated in table 3.4. Our method significantly outperforms the competing methods on both datasets. Additionally, we do not obtain improvements when combining our method with the sparse G-CRF approach. Qualitative results can be seen in Fig. 3.3.

3.4 Summary

In this Chapter we propose a fully-connected G-CRF model for end-to-end training of deep architectures. We propose strategies for efficient implementation and show that inference over a fully-connected graph comes with negligible computational overhead compared to a sparsely connected graph. Our experimental evaluation indicates consistent improvements over the state of the art approaches on three challenging public benchmarks for semantic segmentation, human part segmentation and saliency estimation.



(a) Unary (b) sparse G-CRF (c) dense G-CRF (d) Image+GT

Figure 3.1: Qualitative Results of Semantic Segmentation. (a) shows the unary network output, (b) shows the sparsepotts-G-CRF output, (c) shows the densepotts-G-CRF output, and (d) shows the input image and ground truth.

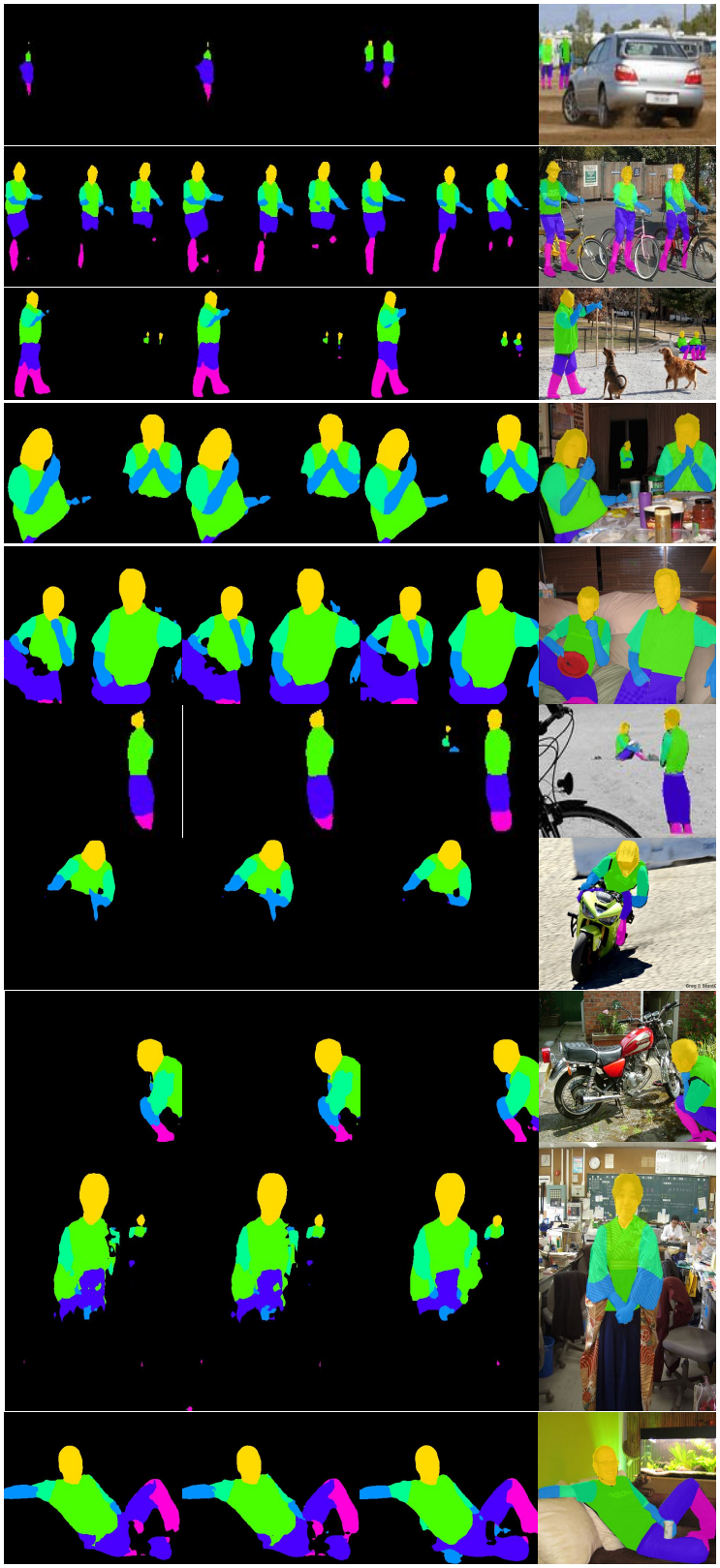


Figure 3.2: Qualitative Results of Part Segmentation. (a) shows the unary network output, (b) shows the sparsepotts-G-CRF output, (c) shows the densepotts-G-CRF output, and (d) shows the input image and ground truth.

Figure 3.3: Qualitative Results of Saliency Estimation. (a) shows the unary network output, (b) shows the sparsepotts-G-CRF output, (c) shows the densepotts-G-CRF output, and (d) shows the input image and ground truth.

Chapter 4

Deep Spatio-Temporal Random Fields for Efficient Video Segmentation

---

In this chapter we extend the fully-connected Deep G-CRF model described in the previous chapter to videos. In particular, we introduce a time- and memory-efficient method for structured prediction in videos that couples neuron decisions across both space and time. We show that we are able to perform exact and efficient inference on a densely-connected spatio-temporal graph by capitalizing on recent advances on deep Gaussian random fields. We experiment with multiple connectivity patterns in the temporal domain, and present empirical improvements over strong baselines on the tasks of both semantic and instance segmentation of videos. Our proposed approach is (a) efficient, (b) has a unique global minimum, and (c) can be trained end-to-end alongside contemporary deep networks for video understanding.

This work will be published at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

4.1 Introduction

Video understanding remains largely unsolved despite dramatic improvements in image understanding over the past five years. The accuracy of current image classification, or semantic segmentation models is not yet matched in action recognition and video segmentation, to some extent due to the lack of large-scale benchmarks, but also due to the complexity introduced by introducing the time variable. Combined with increase in memory and computation demands, video understanding poses additional challenges that call for novel methods.

Our objective in this chapter is to go beyond the frame-by-frame processing currently used in most CNN-based architectures. We focus on coupling the decisions taken by a neural network in time, in a manner that allows information to flow across frames resulting in decisions that are consistent both spatially and temporally. Towards this goal we pursue a structured prediction approach, where the structure of the output space is exploited in order to train classifiers of higher accuracy. For this we introduce into video segmentation the Deep G-CRF method proposed for single-frame structured prediction in Chap. 3. Our main technical contribution in this chapter consists in adapting this method so that it becomes affordable for

video segmentation, both from a time- and memory- complexity viewpoint. To this end, we propose a customized conjugate gradient method that eliminates redundant computations by exploiting the structure of the temporal neighbourhood.

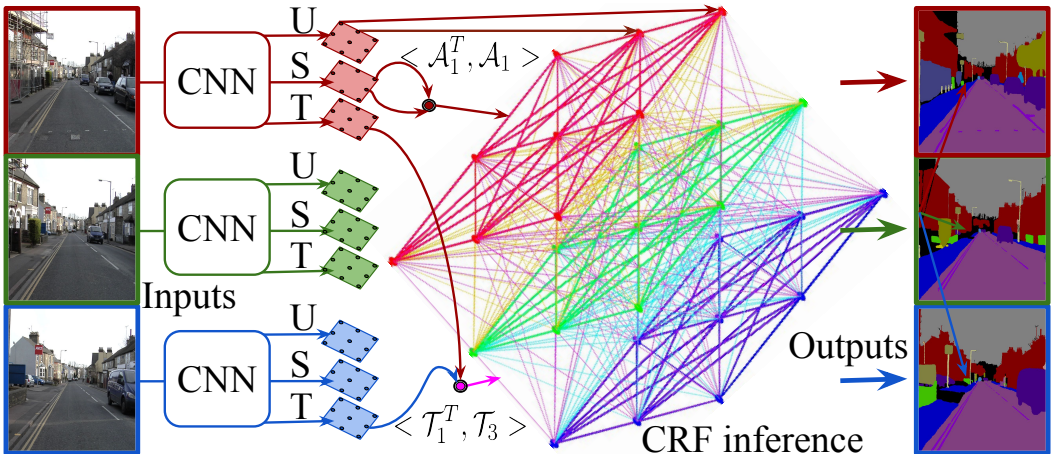


Figure 4.1: Overview of our approach: our deep network takes in V input video frames and delivers unary terms (U), spatial (S) and temporal (T) embeddings for each of the frames to a dense G-CRF module. The G-CRF module uses the unary terms and spatio-temporal embeddings to express the unary, the intra- and inter-frame pairwise terms for a densely connected Gaussian-CRF. The dense G-CRF module does spatio-temporal structured prediction via efficient G-CRF inference to deliver output which is used to generate segmentations. Our network can be trained in an end-to-end manner. Please note that the 3 frames in this example are colour coded as red, green, and blue: these colours are used to indicate correspondences between the inputs, network outputs, and unary, pairwise interactions in the G-CRF.

We show that our algorithm can be used for a variety of video segmentation tasks: semantic segmentation (CamVid dataset), instance tracking (DAVIS dataset), and a combination of instance segmentation with Mask-RCNN-style object detection, customized in particular for the person class (DAVIS Person dataset).

Our approach proposed in this chapter inherits all favorable properties of the G-CRF method: in particular, our method has the advantage of delivering (a) exact inference results through the solution of a linear system, unlike contemporary video understanding approaches such as [Kundu 2016], (b) allowing for exact computation of the gradient during back-propagation, (c) making it possible to use rich CNN-based expression for the pairwise term, rather than confining ourselves to pairwise terms of a predetermined form, and (d) facilitating end-to-end training of all model parameters.

Within the literature on spatio-temporal structured prediction, the work that is closest in spirit to ours is the work of [Kundu 2016] on Feature Space Optimization. Even though our works share several conceptual similarities, our method is entirely different at the technical level in the sense that it is conceived as a neural network

module for structured prediction that is trained jointly with CNNs that process the individual frames, while the method of [Kundu 2016] is applied at a post-processing stage to refine a classifier's results.

4.1.1 Previous work

In Chap. 1 Sec. 1.3, we discuss contemporary structured prediction approaches which capture spatial constraints within an image frame. These approaches may be extended naively to videos, by making predictions individually for each frame. However, in doing so, we ignore the temporal context, thereby ignoring the tendency of consecutive video frames to be similar to each other. To address this shortcoming, a number of deep learning methods employ some kind of structured prediction strategy to ensure temporal coherence in the predictions. Initial attempts to capturing spatio-temporal context involved designing deep learning architectures [Karpathy 2014] that implicitly learnt interactions between consecutive image frames. A number of subsequent approaches used Recurrent Neural Networks (RNNs) [Adi 2017, Donahue 2015] to capture interdependencies between the image frames. Further, several approaches have exploited optical flow computed from state of the art approaches, as in [Ilg 2017], as additional input to the network [Gadde 2017, Jain 2017]. Finally, methods that rely on explicit capturing of these temporal constraints via pairwise terms over probabilistic graphical models also exist [Bratieres 2015, Kundu 2016].

In this chapter, we focus on three problems, namely (i) semantic and (ii) instance video segmentation, and (iii) semantic instance tracking. Semantic instance tracking refers to the problem where we are given the ground truth for the first frame of a video, and the goal is to predict these instance masks on the subsequent frames of the video. Contemporary deep learning literature describes two distinct approaches to this task. The first set of approaches start with a deep network pretrained for image classification on large datasets such as Imagenet or COCO, and finetune it on the first frame of the video with labeled ground truth [Caelles 2017, Voigtlaender 2017], optionally leveraging a variety of data augmentation regimes [Khoreva 2017] to increase robustness to scale/pose variation and occlusion/truncation in the subsequent frames of the video. The second set of approaches pose this problem as a warping problem [Perazzi 2017], where the goal is to warp the segmentation of the first frame using the images and optical flow as additional inputs [Jampani 2017, Khoreva 2017, Li 2017].

A number of approaches have attempted to exploit temporal information to improve over static image segmentation approaches for video segmentation. Clockwork convnets [Shelhamer 2016] were introduced to exploit the persistence of features across time and schedule the processing of some layers at different update rates according to their semantic stability. Similar feature flow propagation ideas were employed in [Kundu 2016, Zhu 2016]. In [Nilsson 2016], the images are warped using the flow and spatial transformer networks. Rather than using optical flow, the prediction of future segmentations [Jin 2016] may also temporally smooth frame

by frame results. Finally, the state-of-the-art improving over PSPnet [Zhao 2016] is achieved by warping the feature maps of a static segmentation CNN to emulate a video segmentation network [Gadde 2017].

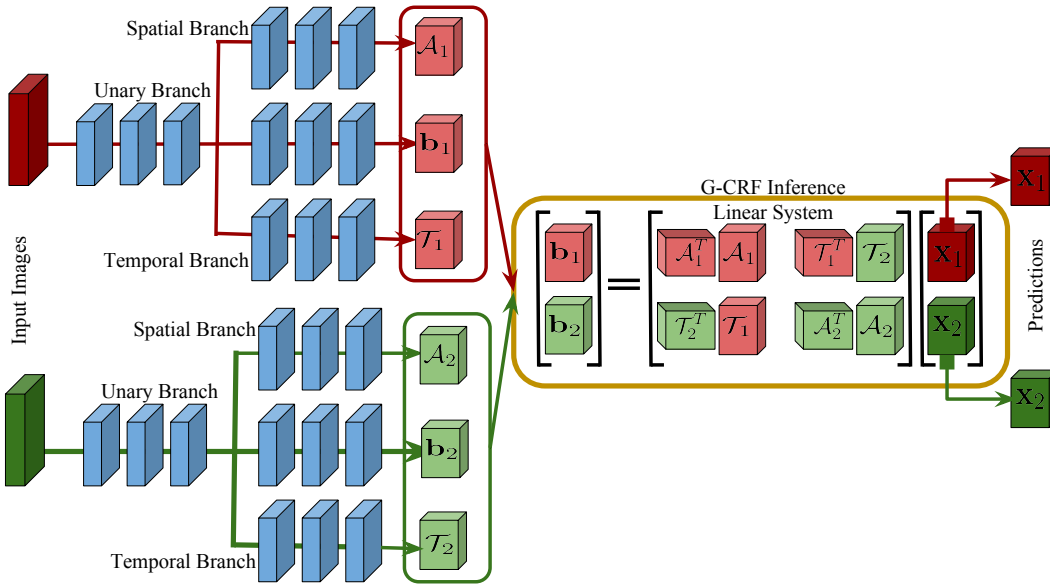


Figure 4.2: Spatio-Temporal G-CRF schematic for 2 video frames. Our network takes in two input images, and delivers the per frame unaries $\mathbf{b}_1, \mathbf{b}_2$, spatial embeddings $\mathcal{A}_1, \mathcal{A}_2$, and temporal embeddings $\mathcal{T}_1, \mathcal{T}_2$ in the feed-forward mode. Our spatio-temporal G-CRF module collects these and solves the inference problem described in Eq. 4.2 to recover predictions $\mathbf{x}_1, \mathbf{x}_2$ for the two frames. During backward pass, the gradients of the predictions are delivered to the spatio-temporal G-CRF model. It uses these to compute the gradients for the unary terms as well as the spatio-temporal embeddings and back-propagates them through the network.

4.2 Spatio-Temporal Gaussian Random Fields

In this chapter we extend the fully-connected Deep Gaussian CRF approach introduced in Chap. 3 to operate efficiently for video segmentation. Introducing a CRF allows us to couple the decisions between sets of variables that should be influencing each other; spatial connections were already explored in Chap. 2 and 3 and can be understood as propagating information from distinctive image positions (e.g. the face of a person) to more ambiguous regions (e.g. the person's clothes). In this chapter we introduce temporal connections, which allow us to smooth information over time, allowing us for instance to correctly segment frames where the object is not clearly visible by propagating information from different time frames.

We consider that the input to our system is a video $\mathcal{V} = \{I_1, I_2, \ldots, I_V\}$ containing V frames. We denote our network's prediction as $\mathbf{x}_v,\ v = 1, \ldots, V$, where

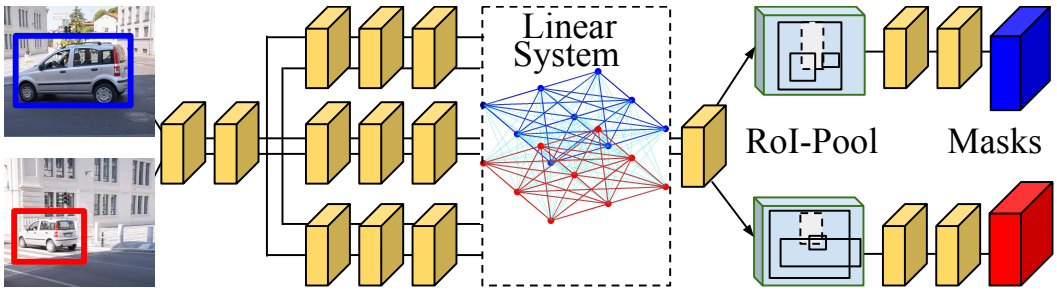


Figure 4.3: Spatio-temporal structured prediction in Mask-RCNN. Here we use the G-CRF linear system in the feature learning stage before the ROI-Pooling (and not as the final classifier). This helps learn mid-level features which are better aware of the spatio-temporal context.

at any frame the prediction $\mathbf{x}_i \in \mathbb{R}^{PL}$ provides a real valued vector giving a distribution of scores over the L classes for each of the P image patches; for brevity, we denote by $N = P \times L$ the number of prediction variables. As in earlier chapters, the L scores corresponding to a patch can be understood as inputs to a softmax function that yields the label posteriors.

As in the previous chapters, we treat the G-CRF as a structured prediction module that is part of a deep network. In the forward pass, the unary and the pairwise terms $B_{\mathcal{V}}$ and $A_{\mathcal{V}}$, delivered by a feed-forward CNN described in Sec. 4.2.1, are fed to the G-CRF module which performs inference to recover the prediction $\mathbf{x}$ by solving a system of linear equations given by

$$(A_{\mathcal{V}} + \lambda \mathbf{I})\mathbf{x} = B_{\mathcal{V}}, \tag{4.1}$$

Our first contribution in this chapter consists in designing the structure of the matrix $A_{\mathcal{V}}$ so that obtaining the resulting system solution remains manageable as the number of frames increases. Once we describe how we structure $A_{\mathcal{V}}$, we then will turn to learning our network in an end-to-end manner. In the rest of this chapter we omit the conditioning on $\mathcal{V}$ for notational convenience.

4.2.1 Spatio-temporal Connections

In order to capture the spatio-temporal context, we are interested in capturing two kinds of pairwise interactions: (a) pairwise terms between patches in the same frame, and (b) pairwise terms between patches in different frames.

Denoting the spatial pairwise terms at frame v by A_v and the temporal pairwise terms between frames u, v as $T_{u,v}$ we can rewrite Eq. 4.1 as follows:

$$\begin{bmatrix} A_1 + \lambda \mathbf{I} & T_{1,2} & \cdots & T_{1,V} \\ T_{2,1} & A_2 + \lambda \mathbf{I} & \cdots & T_{2,V} \\ & & \vdots & \\ T_{V,1} & T_{V,2} & \cdots & A_V + \lambda \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_V \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_V \end{bmatrix}, \tag{4.2}$$

where we group the variables by frames. Solving this system allows us to couple predictions $\mathbf{x}_v$ across all video frames $v \in \{1, \dots, V\}$, positions, p and labels l. If furthermore $A_v = A_v^T, \forall v$ and $T_{u,v} = T_{v,u}^T, \forall u, v$ then the resulting system is positive definite for any positive λ.

We start by describing how the pairwise terms $A_v, T_{u,v}$ are constructed through our CNN, and then turn to how the solution of the linear system in Eq. 4.2 can be accelerated by exploiting its structure.

Spatial Connections: We define the spatial pairwise terms in terms of inner products of pixel-wise embeddings, following Chap. 3. At frame v we couple the scores for a pair of patches p_i, p_j taking the labels l_m, l_n respectively as follows:

$$A_{v,p_i,p_j}(l_m, l_n) = \langle \mathcal{A}_{v,p_i}^{l_m}, \mathcal{A}_{v,p_j}^{l_n} \rangle, \tag{4.3}$$

where $i, j \in \{1, \dots, P\}$ and $m, n \in \{1, \dots, L\}$, $v \in \{1, \dots, V\}$, and $\mathcal{A}_{v,p_j}^{l_n} \in \mathbb{R}^D$ is the embedding associated to point p_j. In Eq. 4.3 the $\mathcal{A}_{v,p_j}^{l_n}$ terms are image-dependent and delivered by a fully-convolutional "embedding" branch that feeds from the same CNN backbone architecture, and is denoted by $\mathcal{A}_v$ in Fig. 4.2.

The implication of this form is that we can afford inference with a fully-connected graph. In particular the rank of the block matrix $A_v = \mathcal{A}_v^\top \mathcal{A}_v$, equals the embedding dimension D, which means that both the memory- and time- complexity of solving the linear system drops from $O(N^2)$ to $O(ND)$, which can be several orders of magnitude smaller.

Temporal Connections: Turning to the *temporal* pairwise terms, we couple patches p_i, p_j coming from different frames u, v taking the labels l_m, l_n respectively as

$$T_{u,v,p_i,p_j}(l_m, l_n) = \langle \mathcal{T}_{u,p_i}^{l_m}, \mathcal{T}_{v,p_j}^{l_n} \rangle, \tag{4.4}$$

where $u, v \in \{1, \dots, V\}$. The respective embedding terms are delivered by a branch of the network that is separate, temporal embedding network, denoted by $\mathcal{T}_v$ in Fig. 4.2. In short, both the spatial pairwise and the temporal pairwise terms are composed as Gram matrices of spatial and temporal embeddings as $A_v = \mathcal{A}_v^\top \mathcal{A}_v$, and $T_{u,v} = \mathcal{T}_u^\top \mathcal{T}_v$. We visualize our spatio-temporal pairwise terms in Fig. 4.1.

Spatio-Temporal G-CRFs in Deep Learning: Our st-G-CRFs can be viewed as generic deep learning modules for spatio-temporal structured prediction, and as such can be used at any stage of a deep learning pipeline: either as the last layer, i.e. classifier, as in our semantic segmentation experiments (Sec. 4.3.3), or even in the low-level feature learning stage, as in our instance segmentation experiments (Sec. 4.3.1).

4.2.2 Efficient Conjugate-Gradient Implementation

We now describe an efficient implementation of the conjugate gradient method [Shewchuk 1994], described in in Algorithm 3 that is customized for our spatio-temporal G-CRFs.

Algorithm 3 Conjugate Gradient Algorithm

1: **procedure** ConjugateGradient
2: **Input:** $\mathbf{A}$, $\mathbf{B}$, $\mathbf{x}_0$ **Output:** $\mathbf{x} \mid \mathbf{Ax} = \mathbf{B}$
3: $\mathbf{r}_0 := \mathbf{B} - \mathbf{Ax}_0$; $\mathbf{p}_0 := \mathbf{r}_0$; $k := 0$
4: repeat
5: $\alpha_k := \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A} \mathbf{p}_k}$
6: $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
7: $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$
8: if $\|\mathbf{r}_{k+1}\|$ is sufficiently small, then exit loop
9: $\beta_k := \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$
10: $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
11: $k := k + 1$
12: end repeat
13: $\mathbf{x} = \mathbf{x}_{k+1}$

The computational complexity of the conjugate gradient algorithm is determined by the computation of the matrix-vector product $\mathbf{q} = A\mathbf{p}$, corresponding to `line :7` of Algorithm 3 (we drop the subscript k for convenience).

We now discuss how to efficiently compute $\mathbf{q}$ in a manner that is customized for this work. In our case, the matrix-vector product $\mathbf{q} = A\mathbf{p}$ is expressed in terms of the spatial ($\mathcal{A}$) and temporal ($\mathcal{T}$) embeddings as follows:

$$\begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_V \end{bmatrix} = \begin{bmatrix} \mathcal{A}_1^T\mathcal{A}_1 + \lambda\mathbf{I} & \mathcal{T}_1^T\mathcal{T}_2 & \cdots & \mathcal{T}_1^T\mathcal{T}_V \\ \mathcal{T}_2^T\mathcal{T}_1 & \mathcal{A}_2^T\mathcal{A}_2 + \lambda\mathbf{I} & \cdots & \mathcal{T}_2^T\mathcal{T}_V \\ & & \vdots & \\ \mathcal{T}_V^T\mathcal{T}_1 & \mathcal{T}_V^T\mathcal{T}_2 & \cdots & \mathcal{A}_V^T\mathcal{A}_V + \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_V \end{bmatrix} \tag{4.5}$$

From Eq. 4.5, we can express $\mathbf{q}_i$ as follows:

$$\mathbf{q}_i = \mathcal{A}_i^T\mathcal{A}_i\mathbf{p}_i + \lambda\mathbf{p}_i + \sum_{j \neq i} \mathcal{T}_i^T\mathcal{T}_j\mathbf{p}_j. \tag{4.6}$$

One optimization that we exploit in computing $\mathbf{q}_i$ efficiently is that we do not 'explicitly' compute the matrix-matrix products $\mathcal{A}_i^T\mathcal{A}_i$ or $\mathcal{T}_i^T\mathcal{T}_j$. We note that $\mathcal{A}_i^T\mathcal{A}_i\mathbf{p}_i$ can be decomposed into two matrix-vector products as $\mathcal{A}_i^T(\mathcal{A}_i\mathbf{p}_i)$, where the expression in the brackets is evaluated first and yields a vector, which can then be multiplied with the matrix outside the brackets. This simplification alleviates the need to keep $N \times N$ terms in memory, and is computationally cheaper.

Further, from Eq. 4.6, we note that computation of $\mathbf{q}_i$ requires the matrix-vector product $\mathcal{T}_j\mathbf{p}_j \ \forall j \neq i$. A *black-box* implementation would therefore involve

redundant computations. We eliminate this redundancy by rewriting Eq. 4.6 as

$$\mathbf{q}_i = \mathcal{A}_i^T \mathcal{A}_i \mathbf{p}_i + \lambda \mathbf{p}_i + \mathcal{T}_i^T \left((\sum_j \mathcal{T}_j \mathbf{p}_j) - \mathcal{T}_i \mathbf{p}_i \right). \tag{4.7}$$

This rephrasing allows us to precompute and cache $\sum_j \mathcal{T}_j \mathbf{p}_j$, thereby eliminating redundant calculations.

While so far we have assumed dense connections between the image frames, if we have sparse temporal connections (Sec. 4.3.1), i.e. each frame is connected to a subset of neighbouring frames in the temporal domain, the linear system matrix A is sparse, and $\mathbf{q}_i$ is written as

$$\mathbf{q}_i = \mathcal{A}_i^T \mathcal{A}_i \mathbf{p}_i + \lambda \mathbf{p}_i + \sum_{j \in \mathcal{N}(i)} \mathcal{T}_i^T \mathcal{T}_j \mathbf{p}_j, \tag{4.8}$$

where $\mathcal{N}(i)$ denotes the temporal neighbourhood of frame i. For very sparse connections caching may not be necessary because these involve little or no redundant computations.

4.2.3 Backward Pass

By virtue of relying on the Gaussian CRF we can get the back-propagation equation for the gradient of the loss with respect to the unary terms, $\mathbf{b}_v$, and the spatial/temporal embedding terms $\mathcal{A}_v, \mathcal{T}_v$ in closed form, thereby sparing us from having to perform back-propagation in time which was needed e.g. in [Zheng 2015] for DenseCRF inference. Following Chap. 3, the gradients of the unary terms $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_v}$ are obtained from the solution of the following system:

$$\begin{bmatrix} A_1 + \lambda \mathbf{I} & T_{1,2} & \cdots & T_{1,V} \\ T_{2,1} & A_2 + \lambda \mathbf{I} & \cdots & T_{2,V} \\ & & \vdots & \\ T_{V,1} & T_{V,2} & \cdots & A_V + \lambda \mathbf{I} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{b}_1} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}_V} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_1} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{x}_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \mathbf{x}_V} \end{bmatrix}. \tag{4.9}$$

Once these are computed, the gradients of the spatial embeddings can be computed as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_v} = -\left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_v} \otimes \mathbf{x}_v \right) \left(\left(\mathbf{I} \otimes \mathcal{A}_v^\top \right) + \left(\mathcal{A}_v^\top \otimes \mathbf{I} \right) Q_{D,N} \right) \tag{4.10}$$

while the gradients of the temporal embeddings are given by the following form:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{T}_v} = -\sum_u \left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_u} \otimes \mathbf{x}_v \right) \left(\left(\mathbf{I} \otimes \mathcal{T}_u^\top \right) + \left(\mathcal{T}_u^\top \otimes \mathbf{I} \right) Q_{D,N} \right) \tag{4.11}$$

where $Q_{D,N}$ is a permutation matrix, defined along the lines of Chap. 3; we provide the details of the derivation in the Appendix 4.A.

4.2.4 Implementation and Inference Time

Our implementation is GPU based and exploits fast *CUDA-BLAS* routines for linear algebra. It is implemented as a module in the Caffe2 library. For spatial and temporal embeddings of size 128, 12 classes (Sec. 4.3.3), a 321×321 input image, and network stride of 8, our $2, 3, 4$ frame inferences take 0.032s, 0.045s and 0.061s on average respectively. Without the caching procedure described in Sec. 4.2.2, the 4 frame inference takes 0.080s on average. This is orders of magnitude faster than the Dense-CRF method [Krähenbühl 2011] which takes 0.2s on average for spatial CRF per frame; For 4 frames this would be 0.8s if we ignore any computational overheads to to the addition of temporal connections, compared to 0.08s in our case. These timing statistics were estimated on a `GTX-1080` GPU.

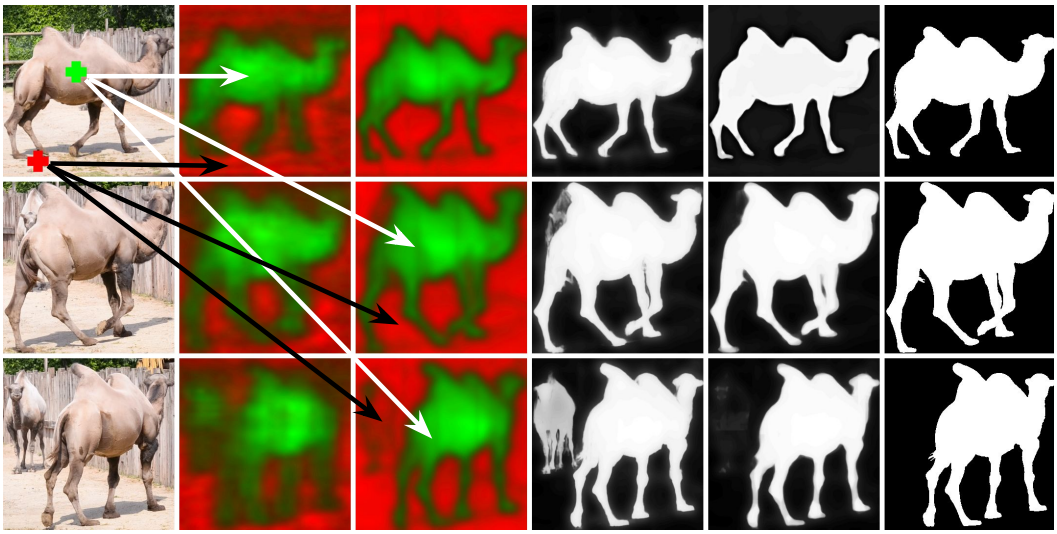


Figure 4.1: Spatio-temporal G-CRF visualization: We mark two points on the reference frame of the video with + signs. We plot the heatmaps of the spatial and temporal pairwise affinities between these 2 points and all other points on the reference frame (row 1) as well two subsequent frames (rows $2, 3$). Further, we show heatmaps of the unary scores, followed by our predictions and the ground truth. We note that while the spatial embeddings may be confused by the presence of a second camel, as in the case of the unary classifier, the temporal context helps recover the correct instance mask.

4.3 Experiments

Experimental Setup. We describe the basic experimental setup followed for our experiments. As in Chap. 3, we use a 3−phase training strategy for our methods. We first train the unary network without the spatio-temporal embed-

dings. We next train the subnetwork delivering the spatio-temporal embeddings with the cross-entropy loss to enforce the following objectives: $A_{p_1,p_2}(l_1,l_2) < A_{p_1,p_2}(l_1' \neq l_1, l_2' \neq l_2)$, and $T_{u,v,p_1,p_2}(l_1,l_2) < T_{u,v,p_1,p_2}(l_1' \neq l_1, l_2' \neq l_2)$, where l_1, l_2 are the ground truth labels for pixels p_1, p_2. Finally, we combine the unary and pairwise networks, and train them together in end-to-end fashion. For the embedding branches, we use sub-networks of 10 layers each on top of the standard ResNet-101 `conv-4` layer. Unless otherwise stated, we use stochastic gradient descent to train our networks with a momentum of 0.9 and a weight decay of $5e^{-4}$. For segmentation experiments, we use a base-learning rate of $2.5e^{-3}$ for training the unaries, $2.5e^{-4}$ for training the embeddings, and $1e^{-4}$ for finetuning the unary and embeddings together, using a polynomial-decay with power of 0.9. For the instance segmentation network, we use a single stage training for the unary and pairwise streams: we train the network for 16K iterations, with a base learning rate of 0.01 which is reduced to 0.001 after 12K iterations. The weight decay is $1e^{-4}$. For our instance tracking experiments, we use unaries from [Voigtlaender 2017] and do not refine them, rather use them as an input to our network. We employ horizontal flipping and scaling by factors between 0.5 and 1.5 during training/testing for all methods, except in the case of instance segmentation experiments (Sec. 4.3.1).

Datasets. We now describe the datasets for experiments.

DAVIS. The DAVIS dataset [Perazzi 2016] consists of 30 training and 20 validation videos containing 2079 and 1376 frames respectively. Each video comes with manually annotated segmentation masks for foreground object instances.

DAVIS-Person. While the DAVIS dataset [Pont-Tuset 2017] provides densely annotated frames for instance segmentation, it lacks object category labels. For category prediction tasks such as semantic and instance segmentation, we create a subset of the DAVIS dataset containing videos from the category person. By means of visual inspection, we select 35 and 18 video sequences from the training and validation sets respectively containing 2463 training and 1182 validation images, each containing at least one person. Since the DAVIS dataset comes with only the *foreground* instances labeled, we manually annotate the image regions containing *unannotated person* instances with the *do-not-care* label. These image regions do not participate in the training or the evaluation. We call this the DAVIS-person dataset.

CamVid. The CamVid dataset [Brostow 2017], is a dataset containing videos of driving scenarios for urban scene understanding. It comes with 701 images annotated with pixel-level category labels at 1 fps. Although the original dataset comes with 32 class-labels, as in [Badrinarayanan 2015, Kundu 2016, Jégou 2017], we predict 11 semantic classes and use the train-val-test split of 367, 101 and 233 frames respectively.

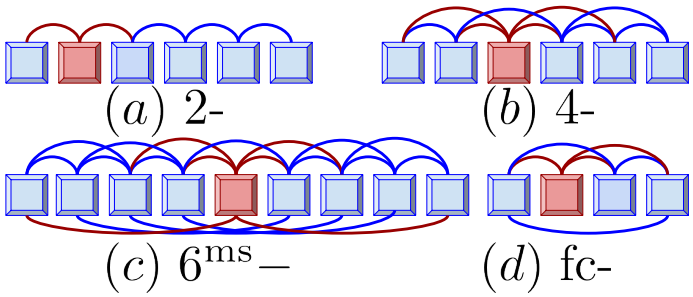


Figure 4.1: Illustration of the various temporal neighbourhoods we consider in our ablation study. Each box denotes a video frame and the arcs connecting them are pairwise connections. We show a frame in red with all neighbours present in the temporal context.

4.3.1 Ablation Study on Semantic and Instance Segmentation Tasks

In these experiments, we use the DAVIS Person dataset described in Sec. 4.3. The aim here is to explore the various design choices available to us when designing networks for spatio-temporal structured prediction for semantic segmentation, and proposal based instance segmentation tasks.

Semantic Segmentation Experiments. Our first set of experiments studies the effect of varying the sizes of the spatial and temporal embeddings, the degree of the temporal connections, and multi-scale temporal connections for the spatio-temporal G-CRFs. For these set of experiments, our baseline network, or *base-net* is a single resolution ResNet-101 network, with altered network strides as in [Chen 2016a] to produce a spatial down-sampling factor of 8. The network was pretrained on the PASCAL VOC 2012 dataset for semantic segmentation. The evaluation metric used in these experiments is the mean pixel Intersection over Union.

In Tab. 4.1 we study the effect of varying the sizes of the spatial and temporal embeddings, and compare the performance of our methods that couple 2−frames at a time, against that of the base-net. Our best results are achieved at when the sizes of our spatio-temporal embeddings are 128. The improvement over the base-net is 4.2%. In the rest of our experiments in this work, we fix the size of our embeddings to 128. We next study the effect of varying the size of the temporal context and temporal neighbourhoods. The temporal context is defined as the number of video frames $\mathcal{V}$ which are considered simultaneously in one linear system (Eq. 4.2). The temporal context $\mathcal{V}$ is limited by the size of the network and GPU RAM: for a ResNet-101 network, an input image of size 321×321, embeddings of size 128, we can currently fit $\mathcal{V} = 7$ frames on 12 GB of GPU RAM, thus the maximum temporal context possible in this setting is 7 frames. Since $\mathcal{V}$ is smaller than the number of frames in the video, we divide the video into overlapping sets of $\mathcal{V}$ frames, and average the predictions for the common frames.

The temporal neighbourhood for a frame (Fig. 4.1) is defined as the number of frames it is directly connected to via pairwise connections. A fully connected neighbourhood (fc−) is one in which there are pairwise terms between every pair of frames available in the temporal context. We experiment with 2−, 4−, multiscale 6^{ms}− and fc− connections. The 6^{ms}− neighbourhood connects a frame to neighbours at distances of 2^0, 2^1 and 2^2 (or $1, 2, 4$) frames on either side. Tab. 4.2 reports our results for different combinations of temporal neighbourhood and context. It can be seen that dense connections improve performance for smaller temporal contexts, but for a temporal context of 7 frames, an increase in the complexity of temporal connections leads to decrease in performance.

base-net	**81.16**			
st-G-CRF	spatial dimension→			
temporal dimension↓	64	128	256	512
64	84.89	85.21	85.20	84.98
128	85.18	**86.38**	86.34	84.91
256	85.92	86.37	85.95	84.92
512	84.85	85.95	84.95	84.21

Table 4.1: Ablation study: mean IoU on the DAVIS-person dataset using 2 frame fc− connections. Here we study the effect of varying the size of the spatial & temporal embeddings. We fix the size of these embeddings to 128 for subsequent experiments.

base-net	**81.16**			
st-G-CRF	temporal neighbourhood →			
temporal context↓	2−	4−	6^{ms}−	fc−
2	–	–	–	86.38
3	86.42	–	–	86.51
4	86.70	–	–	86.82
7	**86.98**	86.79	86.82	86.42

Table 4.2: Ablation study: mean IoU on the DAVIS-person dataset. Here we study the effect of varying the size of the temporal context and neighbourhood.

Instance Segmentation Experiments. In this set of experiments, we demonstrate the utility of our spatio-temporal G-CRF method for the task of proposal based instance segmentation. Our hypothesis is that coupling of predictions across frames is advantageous for instance segmentation methods, and our goal is to show that the performance of the instance segmentation methods improves as we increase the temporal context via spatio-temporal G-CRFs: we use fully connected temporal neighbourhoods. Our baseline for this task is the Mask-RCNN framework of [He 2017] using the ResNet-50 network as the convolutional backbone.

The Mask-RCNN framework uses precomputed bounding box proposals for this task. It computes convolutional features on the input image using the convolutional backbone network, crops out the features corresponding to image regions in the proposed bounding boxes via Region-Of-Interest (RoI) pooling, and then has 3 head networks to predict (i) class scores and bounding box regression parameters, (ii) keypoint locations, and (iii) instance masks. Structured prediction coupling the predictions of all the proposals over all the video frames is a computationally challenging task, since typically we have $100 - 1000$s of proposals per image, and it is not obvious which proposals from one frame should influence which proposals in the other frame. To circumvent this issue, we use our G-CRFs before the RoI pooling stage as shown in Fig. 4.3. Thus, rather than coupling final predictions, we are coupling mid-level features over the video frames in an attempt to improve the features which will ultimately be used to make predictions.

For evaluation, we use the standard COCO performance metrics: AP_{50}, AP_{75}, and AP (averaged over IoU thresholds), evaluated using mask IoU. Tab. 4.3 reports our instance segmentation results. We note that the performance of the Mask-RCNN framework increases consistently as we increase the temporal context for predictions. We show the qualitative results of our instance segmentation experiments in Fig. 4.2 and Fig. 4.3.

Method	AP_{50}	AP_{75}	AP
ResNet50-baseline	0.610	0.305	0.321
spatial G-CRF (Chap. 3)	0.618	0.310	0.329
2-frame st-G-CRF	0.619	0.310	0.331
3-frame st-G-CRF	0.631	0.321	0.330
4-frame st-G-CRF	0.647	0.336	0.349

Table 4.3: Instance Segmentation using ResNet-50 Mask R-CNN on the Davis Person Dataset

Method	mean IoU
Mask Track [Perazzi 2017]	79.7
OSVOS [Caelles 2017]	79.8
Online Adaptation [Voigtlaender 2017]	85.6
Online Adaptation + Spatial G-CRF (Chap. 3)	85.9
Online Adaptation + 2-Frame st-G-CRF	86.3
Online Adaptation + 3-Frame st-G-CRF	86.5

Table 4.4: Instance Tracking on the Davis val Dataset

4.3.2 Instance Tracking on DAVIS Dataset

Here we use the DAVIS dataset described in Sec. 4.3. Instance tracking involves predicting segmentation masks for foreground object instances for each frame in

the video when presented with the ground truth segmentation for the first video frame. Our goal here is to demonstrate that incorporating temporal context helps improve performance in instance tracking methods. To this end, we extend the state-of-the-art approach on the DAVIS benchmark, online adaptation approach from [Voigtlaender 2017] with our spatio-temporal G-CRFs. We use their publicly available software based on the TensorFlow library to generate the unary terms for each of the frames in the video, and keep them fixed. We use a ResNet-50 network to generate spatio-temporal embeddings and use these alongside the unaries computed from [Voigtlaender 2017]. The results are reported in table Tab. 4.4. We compare performance of spatio-temporal G-CRFs against that of just the unaries from [Voigtlaender 2017], and also with spatial G-CRFs from Chap. 3. The evaluation criterion is the mean pixel-IoU. It can be seen that temporal context helps improve the performance. We expect that re-implementing the software from [Voigtlaender 2017] in Caffe2 and back-propagating on the unary branch of the network would yield further performance boosts.

Model	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Cyclist	m-IoU
SegNet [Badrinarayanan 2015]	68.7	52.0	87.0	58.5	13.4	86.2	25.3	17.9	16.0	60.5	24.8	46.4
Bayesian SegNet [Kendall 2015]	–											63.1
DeconvNet [Noh 2015]	–											48.9
Visin et al. [Visin 2016]	–											58.8
FCN8 [Long 2015]	77.8	71.0	88.7	76.1	32.7	91.2	41.7	24.4	19.9	72.7	31.0	57.0
DeepLab-LFOV [Chen 2014a]	81.5	74.6	89.0	82.2	42.3	92.2	48.4	27.2	14.3	75.4	50.1	61.6
Dilation8 [Yu 2016]	82.6	76.2	89.0	84.0	46.9	92.2	56.3	35.8	23.4	75.3	55.5	65.3
Dilation8 + FSO [Kundu 2016]	84.0	77.2	91.3	85.6	49.9	92.5	59.1	37.6	16.9	76.0	57.2	66.1
Tiramisu [Jégou 2017]	83.0	77.3	93.0	77.3	43.9	94.5	59.6	37.1	37.8	82.2	50.5	66.9
Results with our ResNet-101 Implementation												
Basenet ResNet-101 (Ours)	81.2	75.1	90.3	85.2	48.3	93.9	57.7	39.9	15.9	80.5	54.8	65.7
Basenet + Spatial G-CRF (Chap. 3)	81.6	75.7	90.4	86.8	48.1	94.0	59.1	39.2	15.7	80.7	54.7	66.0
Basenet + 2-Frame Video G-CRF	82.0	76.1	91.1	86.2	51.7	93.8	64.2	24.5	25.0	80.1	61.7	66.9
Basenet + 3-Frame Video G-CRF	82.1	76.0	91.1	86.1	52.0	93.7	64.5	24.9	24.4	79.9	61.8	67.0
Results after Cityscapes Pretraining												
Basenet ResNet-101 (Ours)	85.5	77.4	90.9	88.4	62.3	95.4	64.8	62.1	33.3	85.5	60.5	73.3
Basenet + Spatial G-CRF (Chap. 3)	86.0	77.8	91.2	90.8	63.6	95.9	66.5	61.2	35.3	86.9	65.8	74.6
Basenet + 2-Frame Video G-CRF	86.0	78.3	91.2	92.0	63.4	96.3	67.0	62.5	34.4	87.7	66.1	75.0
Basenet + 3-Frame Video G-CRF	86.1	78.3	91.2	92.2	63.7	96.4	67.3	63.0	34.4	87.8	66.4	75.2

Table 4.5: Results on CamVid dataset. We compare our results with some of the previously published methods, as well as our own implementation of the ResNet-101 network which serves as our base network.

4.3.3 Semantic Segmentation on CamVid Dataset

In this set of experiments, we employ our st-G-CRFs for the task of semantic video segmentation. Here we use the CamVid dataset described in Sec. 4.3. Our base network here is our own implementation of ResNet-101 with pyramid spatial pooling as in [Zhao 2016]. Additionally, we pretrain our networks on the Cityscapes dataset [Cordts 2016], and report results both with and without Cityscapes pretraining. Our approach out-performs the baseline approaches both with and with-

out Cityscapes pretraining. We see substantial boosts in performance after the pretraining. Without pretraining, we see an improvement of 1.3% over the basenet, and with pretraining we see an improvement of 1.9%. The qualitative results are shown in Fig. 4.2.

4.4 Summary

In this chapter, we propose efficient, end-to-end trainable G-CRFs for efficient spatio-temporal structured prediction. On a number of benchmarks, we experimentally demonstrate performance boosts when we increase the temporal context of predictions. This additional complexity comes at negligible computational overhead compared to spatial structured prediction owing to our efficient implementation.

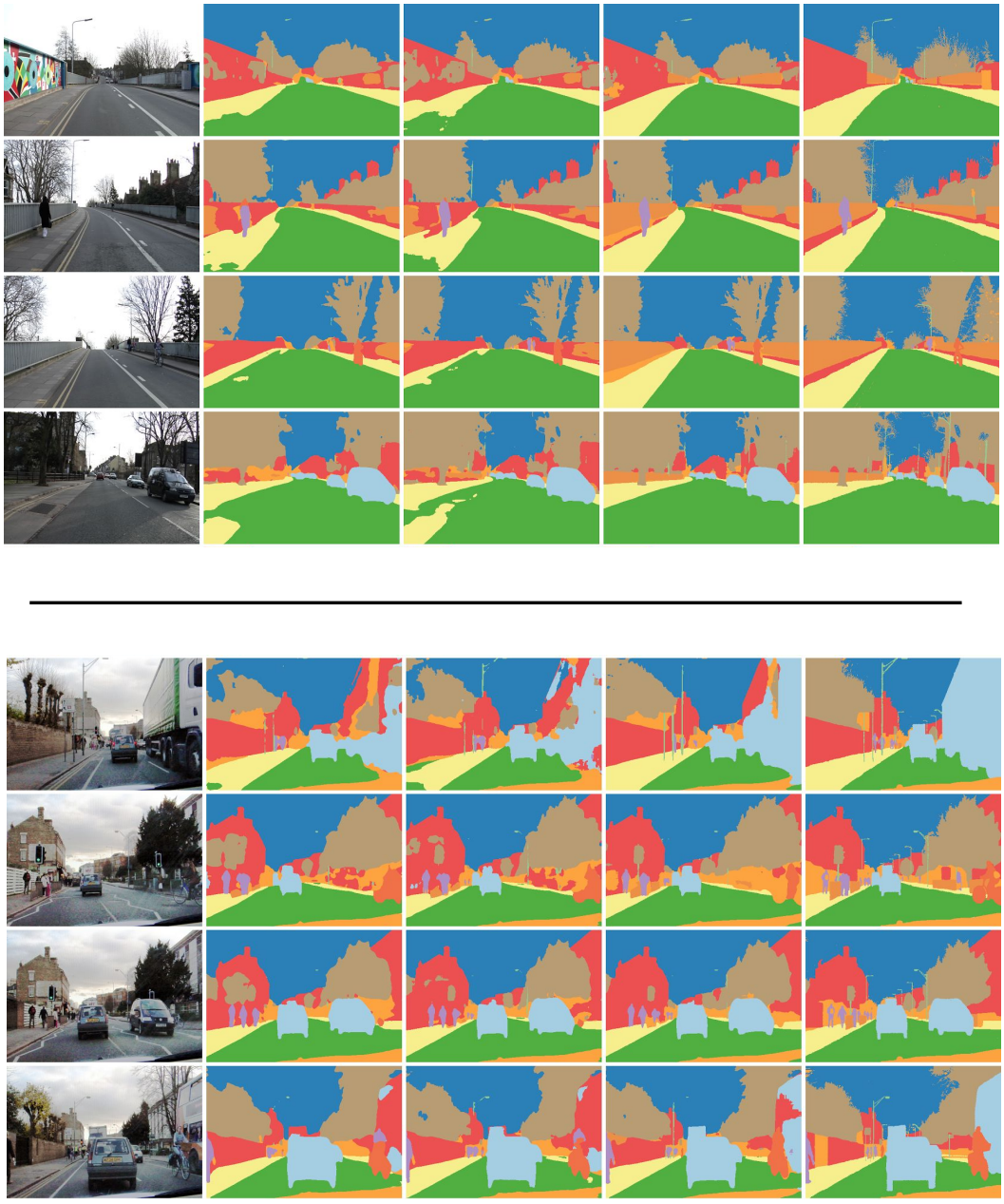


Figure 4.2: Qualitative results on the CamVid dataset. We note that the temporal context from neighbouring frames helps improve the prediction of the truck on the right in the first video, and helps distinguish between the road and the pavement in the second video, overall giving us smoother predictions in both cases.

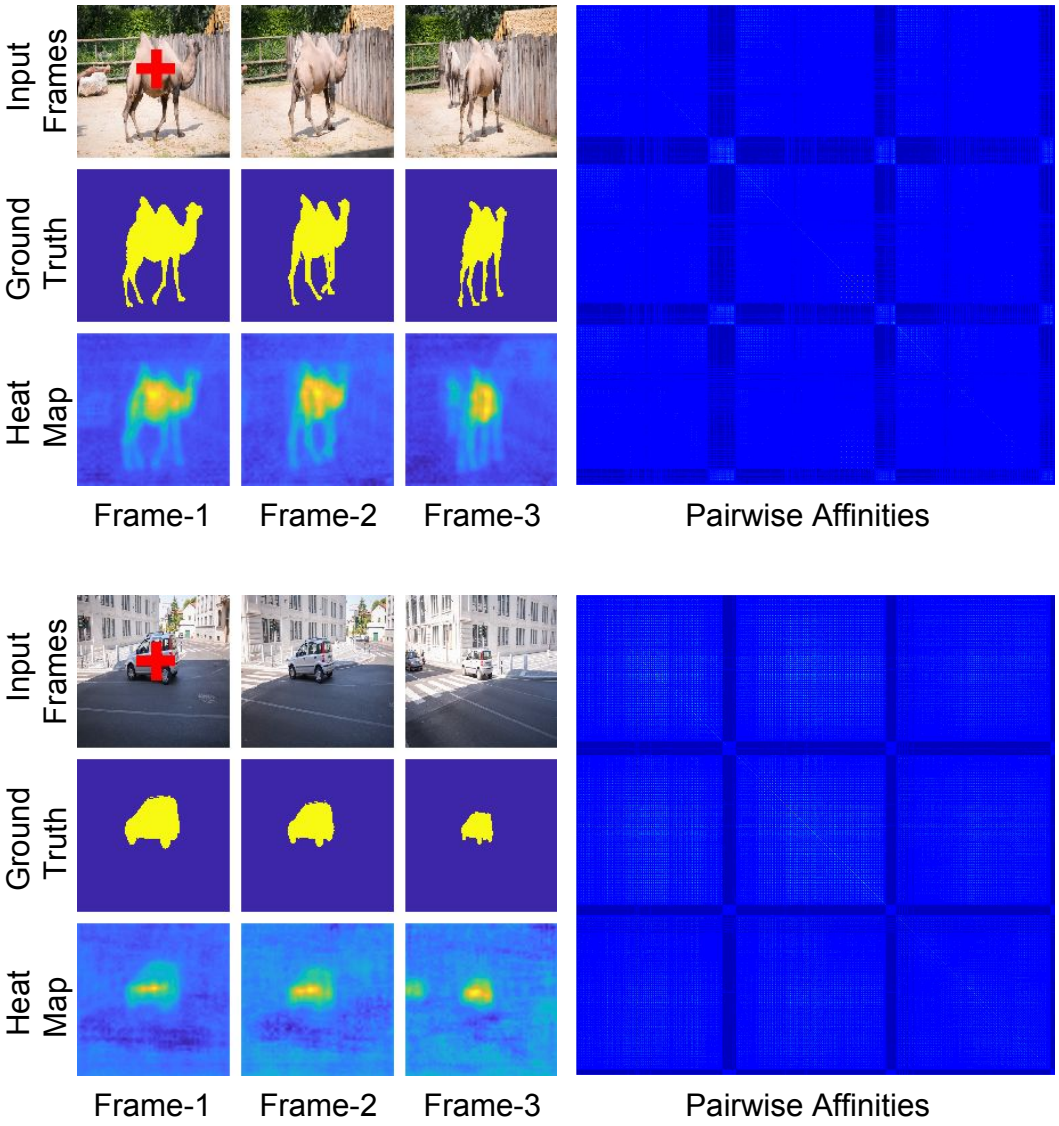


Figure 4.1: Visualization of G-CRF embeddings on 2 videos in the DAVIS dataset. The pairwise affinity between two pixels in the image is given by the dot product of the feature embeddings computed at these two pixels. We choose a *reference* pixel from the Frame-1 (marked by the red cross). We show the heat-map produced by computing the dot-product of the embedding at the reference pixel, with other pixels in the same frame, as well as in two other frames from the video. We also plot the pairwise affinities between every pair of pixels as a heatmap in the 3 frames of the video in the last column of the figure. Here the pixels are ordered according to their class (background followed by object). The bright areas indicate high affinity between pixels belonging to the same class, and the dull areas indicate low affinity between pixels belonging to different classes.

(a) base-net **(b) spatial G-CRF** **(c) st-G-CRF** **(d) GT**

Figure 4.2: Qualitative results for instance segmentation on the DAVIS Person Dataset. We notice that the base-net and the spatial G-CRF in (a),(b) miss the school-girl on the right in the second frame. Temporal context from spatio-temporal G-CRFs in (c) helps recover her.

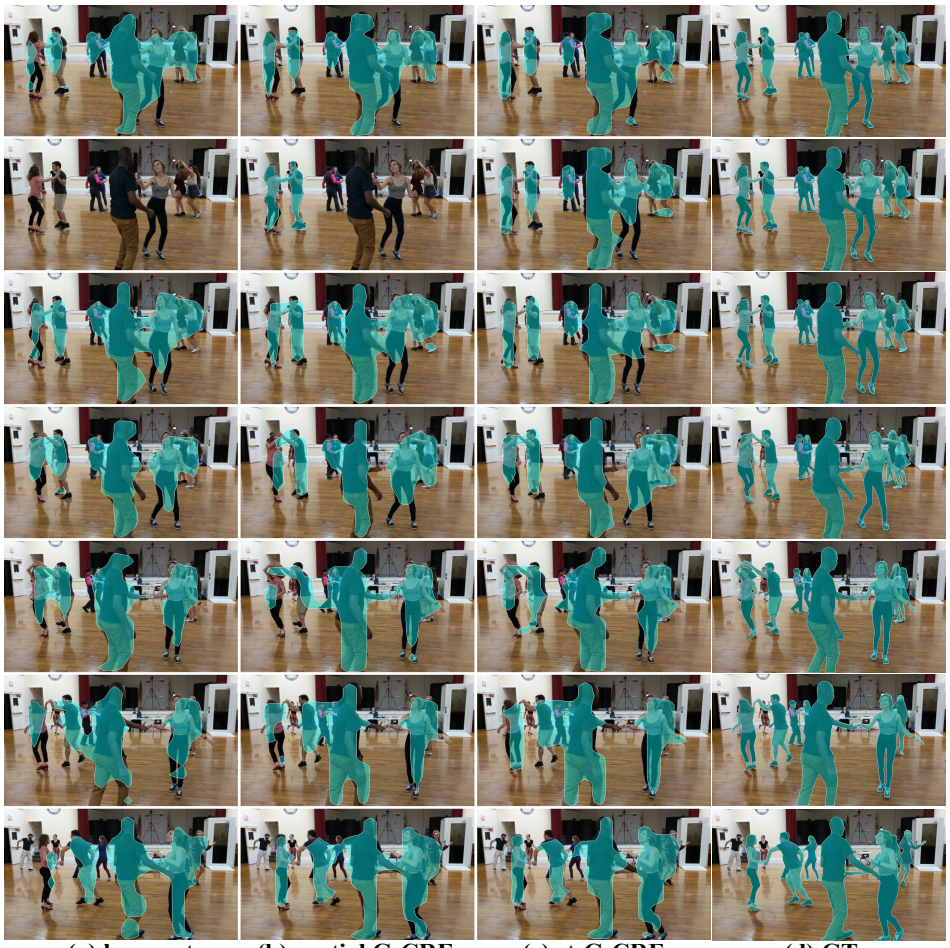


Figure 4.3: Qualitative results for instance segmentation on the DAVIS Person Dataset. We notice that the base-net and the spatial G-CRF in (a),(b) miss instances or parts of instances of dancing persons frequently. Temporal context from spatio-temporal G-CRFs in (c) helps recover missing parts / instances and yield smoother predictions over the video, as seen in row 2.

Appendix

In this appendix, we derive the expressions for weight-update rules for learning for the theory developed in Chap. 4.

4.A Deep Spatio-Temporal Random Fields for Efficient Video Segmentation

As described in Chap. 4, to capture the spatio-temporal context, we propose two kinds of pairwise interactions: (a) pairwise terms between patches in the same frame (spatial pairwise terms), and (b) pairwise terms between patches in different frames (temporal pairwise terms).

Denoting the spatial pairwise terms at frame v by A_v and the temporal pairwise terms between frames u, v as $T_{u,v}$, our inference equation is written as

$$\begin{bmatrix} A_1 + \lambda \mathbf{I} & T_{1,2} & \cdots & T_{1,V} \\ T_{2,1} & A_2 + \lambda \mathbf{I} & \cdots & T_{2,V} \\ & & \vdots & \\ T_{V,1} & T_{V,2} & \cdots & A_V + \lambda \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_V \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_V \end{bmatrix}, \tag{4.12}$$

where we group the variables by frames. Solving this system allows us to couple predictions $\mathbf{x}_v$ across all video frames $v \in \{1, \ldots, V\}$, positions, p and labels l. If furthermore $A_v = A_v^T, \forall v$ and $T_{u,v} = T_{v,u}^T, \forall u, v$ then the resulting system is positive definite for any positive λ.

As in Chap. 4, at frame v we couple the scores for a pair of patches p_i, p_j taking the labels l_m, l_n respectively as follows:

$$A_{v,p_i,p_j}(l_m, l_n) = \langle \mathcal{A}_{v,p_i}^{l_m}, \mathcal{A}_{v,p_j}^{l_n} \rangle, \tag{4.13}$$

where $i, j \in \{1, \ldots, P\}$ and $m, n \in \{1, \ldots, L\}$, $v \in \{1, \ldots, V\}$, and $\mathcal{A}_{v,p_j}^{l_n} \in \mathbb{R}^D$ is the embedding associated to point p_j.

Thus, $\mathcal{A}_v \in \mathbb{R}^{N \times D}$, where $N = P \times L$. Further, to design the *temporal* pairwise terms, we couple patches p_i, p_j coming from different frames u, v taking the labels l_m, l_n respectively as

$$T_{u,v,p_i,p_j}(l_m, l_n) = \langle \mathcal{T}_{u,p_i}^{l_m}, \mathcal{T}_{v,p_j}^{l_n} \rangle, \tag{4.14}$$

where $u, v \in \{1, \ldots, V\}$.

In short, both the spatial pairwise and the temporal pairwise terms are composed as Gram matrices of spatial and temporal embeddings as $A_v = \mathcal{A}_v^\top \mathcal{A}_v$, and $T_{u,v} = \mathcal{T}_u^\top \mathcal{T}_v$.

Using the definitions from Eq. 4.13 and Eq. 4.14, we can rewrite the inference equation as

$$\begin{bmatrix} \mathcal{A}_1^T\mathcal{A}_1 + \lambda\mathbf{I} & \mathcal{T}_1^T\mathcal{T}_2 & \cdots & \mathcal{T}_1^T\mathcal{T}_V \\ \mathcal{T}_2^T\mathcal{T}_1 & \mathcal{A}_2^T\mathcal{A}_2 + \lambda\mathbf{I} & \cdots & \mathcal{T}_2^T\mathcal{T}_V \\ & & \vdots & \\ \mathcal{T}_V^T\mathcal{T}_1 & \mathcal{T}_V^T\mathcal{T}_2 & \cdots & \mathcal{A}_V^T\mathcal{A}_V + \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_V \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_V \end{bmatrix} \tag{4.15}$$

From Eq. 4.15, we can express $\mathbf{b}_v$ as follows:

$$\mathbf{b}_v = \mathcal{A}_v^T\mathcal{A}_v\mathbf{x}_v + \lambda\mathbf{x}_v + \sum_{u \neq v} \mathcal{T}_v^T\mathcal{T}_u\mathbf{x}_u, \tag{4.16}$$

which can be compactly written as

$$\mathbf{b}_v = A_v\mathbf{x}_v + \lambda\mathbf{x}_v + \sum_{u \neq v} T_{v,u}\mathbf{x}_u. \tag{4.17}$$

We will use Eq. 4.17 to derive gradient expressions for $\frac{\partial\mathcal{L}}{\partial\mathcal{A}_v}$ and $\frac{\partial\mathcal{L}}{\partial\mathcal{T}_v}$.

4.B Gradients of the Unary Terms

As in Chap. 2, Chap. 3 the gradients of the unary terms $\frac{\partial\mathcal{L}}{\partial\mathbf{b}_v}$ are obtained from the solution of the following system of linear equations:

$$\begin{bmatrix} \mathcal{A}_1^T\mathcal{A}_1 + \lambda\mathbf{I} & \mathcal{T}_1^T\mathcal{T}_2 & \cdots & \mathcal{T}_1^T\mathcal{T}_V \\ \mathcal{T}_2^T\mathcal{T}_1 & \mathcal{A}_2^T\mathcal{A}_2 + \lambda\mathbf{I} & \cdots & \mathcal{T}_2^T\mathcal{T}_V \\ & & \vdots & \\ \mathcal{T}_V^T\mathcal{T}_1 & \mathcal{T}_V^T\mathcal{T}_2 & \cdots & \mathcal{A}_V^T\mathcal{A}_V + \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\mathbf{b}_1} \\ \frac{\partial\mathcal{L}}{\partial\mathbf{b}_2} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\mathbf{b}_V} \end{bmatrix} = \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\mathbf{x}_1} \\ \frac{\partial\mathcal{L}}{\partial\mathbf{x}_2} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\mathbf{x}_V} \end{bmatrix}, \tag{4.18}$$

where $\mathcal{L}$ is the network loss. Once we have $\frac{\partial\mathcal{L}}{\partial\mathbf{b}_v}$, we use it to compute the gradients of the spatio-temporal embeddings.

4.C Gradients of the Spatial Embeddings

We begin with the observation that computing $\frac{\partial\mathcal{L}}{\partial\mathcal{A}_v}$ requires us to first derive the expression for $\frac{\partial\mathcal{L}}{\partial A_v}$. To this end, we ignore terms from Eq. 4.17 that do not depend on $\mathbf{b}_v$ or A_v and write it as $\mathbf{b}_v = A_v\mathbf{x}_v + c$. We now use the result from Chap. 2 that when

$$A_v\mathbf{x}_v = \mathbf{b}_v,$$

the gradients of A_v are expressed as

$$\frac{\partial \mathcal{L}}{\partial A_v} = -\frac{\partial \mathcal{L}}{\partial \mathbf{b}_v} \otimes \mathbf{x}_v, \tag{4.19}$$

where $\otimes$ denotes the Kronecker product operator.

To compute $\frac{\partial \mathcal{A}_v}{\partial \mathcal{L}}$, we use the chain rule of differentiation as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_v} = \left(\frac{\partial \mathcal{L}}{\partial A_v}\right)\left(\frac{\partial A_v}{\partial \mathcal{A}_v}\right) = \left(\frac{\partial \mathcal{L}}{\partial A_v}\right)\left(\frac{\partial}{\partial \mathcal{A}_v}\mathcal{A}_v^T\mathcal{A}_v\right), \tag{4.20}$$

where $A_v = \mathcal{A}_v^T\mathcal{A}_v$, by definition. We know the expression for $\frac{\partial \mathcal{L}}{\partial A_v}$ from Eq. 4.19, but to obtain the expression for $\frac{\partial}{\partial \mathcal{A}_v}\mathcal{A}_v^T\mathcal{A}_v$ we define a permutation matrix $Q_{m,n}$ of size $mn \times mn$ (as in [Fackler 2005]) as follows:

$$Q_{m,n}\text{vec}(M) = \text{vec}(M^T), \tag{4.21}$$

where $\text{vec}(M)$ is the vectorization operator that vectorizes a matrix M by stacking its columns. Thus, the operator $Q_{m,n}$ is a permutation matrix, composed of 0s and 1s, and has a single 1 in each row and column. When premultiplied with another matrix, $Q_{m,n}$ rearranges the ordering of rows of that matrix, while when postmultiplied with another matrix, $Q_{m,n}$ rearranges its columns. Using this matrix, we can form the following expression [Fackler 2005]:

$$\frac{\partial}{\partial \mathcal{A}_v}\mathcal{A}_v^T\mathcal{A}_v = \left(\mathbf{I} \otimes \mathcal{A}_v^T\right) + \left(\mathcal{A}_v^T \otimes \mathbf{I}\right) Q_{D,N}, \tag{4.22}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix. Substituting Eq. 4.19 and Eq. 4.22 into Eq. 4.20, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_v} = -\left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_v} \otimes \mathbf{x}_v\right)\left(\left(\mathbf{I} \otimes \mathcal{A}_v^\top\right) + \left(\mathcal{A}_v^\top \otimes \mathbf{I}\right) Q_{D,N}\right). \tag{4.23}$$

4.D Gradients of the Temporal Embeddings

As in the last section, from Eq. 4.17, we ignore any terms that do not depend on $\mathbf{b}_v$ or $T_{v,u}$ and write it as $\mathbf{b}_v = c + \sum_{u \neq v} T_{v,u}\mathbf{x}_u$.

Using the strategies in the previous section and the sum rule of differentiation, the gradients of the temporal embeddings are given by the following form:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{T}_v} = -\sum_u \left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_u} \otimes \mathbf{x}_v\right)\left(\left(\mathbf{I} \otimes \mathcal{T}_u^\top\right) + \left(\mathcal{T}_u^\top \otimes \mathbf{I}\right) Q_{D,N}\right) \tag{4.24}$$

Chapter 5

Concluding Remarks

While the majority of approaches in literature have relied on approximate inference for general graphical models, in this thesis we have chosen exactness of inference over expressive power in choosing G-CRFs. Through our systematic experimental evaluations, we have demonstrated that G-CRFs do not lack in expressive power as they out-perform competing approaches which use approximate inference for general graphical models. Further, we demonstrate that G-CRFs can be implemented very efficiently with the inference time orders of magnitude lower than contemporary methods such as the Dense-CRF (see Chap. 2 and 3 for timing comparisons). Additionally, we overcome the computational and memory challenges posed by long-range interactions (Chap. 3) by constructing the pairwise terms as a low-rank Gram matrix of pixel-embeddings. Finally, we derive the gradient expressions and update rules to enable learning of all model parameters in an end-to-end fashion. To sum up, in this thesis, we have proposed a structure prediction method which has (i) exact inference, (ii) long-range connections, (iii) non-parametric pairwise terms, (iv) end-to-end trainable, and is (v) efficient.

In this chapter, we first recapitulate the main contributions of this thesis, and then discuss future avenues that we would like explore.

5.1 Contributions

We now give a formal, more detailed list of our technical contributions in this thesis, organized by the chapters.

5.1.1 G-CRFs for Sparsely Connected Graphical Models

In Chap. 2, we propose a sparse G-CRF method for deep networks which can be trained in an end-to-end fashion. We demonstrate that our inference problem can be solved in closed-form by solving a system of linear equations, and derive the gradients for unary and pairwise terms for back-propagation. We show that back-propagation also involves solving a system of linear equations. Further, we study a number of algorithms for iteratively solving systems of linear equations, and empirically determine that the conjugate gradient algorithm works best for our setting. We also establish parallels between the mean-field iterations that contemporary employ for approximate inference with the Jacobi and Gauss-Seidel approaches that converge slower than conjugate gradient in our experiments.

We also propose a Potts-type variant of our G-CRFs for a simpler, and faster model which uses memory footprint. Further, we propose a multi-scale inference

strategy to capture interactions between image regions at multiple scales. We devise strategies for efficient implementation on the GPU using the CUDA-BLAS library and exploit optimized linear algebra routines for sparse matrices using the CUDA-Sparse library. Our implementation is efficient and exact and can be solved in 0.02 seconds on the GPU for each image in the general setting, and 0.003 seconds for the Potts-type pairwise case using the conjugate gradient method.

Our experimental results indicate that using pairwise terms boosts performance of the network on the task of image segmentation, and our results are competitive with the state of the art methods on the VOC 2012 benchmark, while being substantially simpler.

5.1.2 G-CRFs for Fully-Connected Graphical Models

In Chap. 3, we propose a fully-connected G-CRF model for end-to-end training of deep architectures. To cope with the prohibitive memory and computational demands of a fully-connected graphical model we compose the G-CRF precision matrix as a low-rank Gram matrix of pixel embeddings delivered by a CNN. We derive the gradient expressions and update rules to train all model parameters via back-propagation.

To better exploit the low-rank structure of the G-CRF precision matrix, we propose a *customized* conjugate gradient algorithm which lends itself to very efficient implementation on the GPU. With our customized conjugate gradient algorithm we demonstrate that inference over a fully-connected graph comes with neglegible computational overhead compared to a sparsely connected graph.

Further, we propose a Potts-type variant of our fully-connected G-CRF model which is faster, has a lower memory footprint and contains fewer parameters. We empirically show that the Potts-type variant also outperforms the general model.

Our experimental evaluation indicates consistent improvements over the state of the art approaches on three challenging public benchmarks for semantic segmentation, human part segmentation and saliency estimation.

5.1.3 G-CRFs for Fully-Connected Spatio-Temporal Structured Prediction

In Chap. 4, we extend our fully-connected G-CRF for videos. In particular, we develop G-CRFs for spatio-temporal structured prediction by incorporating pairwise terms between patches in the same and different frames of a video. We derive the gradient expressions and update rules for end-to-end training of all G-CRF parameters via back-propagation.

To allow efficient implementation, we provide a customized conjugate gradient algorithm which eliminates redundant computations. On a number of benchmarks, namely semantic and instance segmentation on videos, and instance tracking in videos, we experimentally demonstrate performance boosts when we increase the temporal context of predictions.

5.2 Future Work

We now discuss some of the novel extensions or domains where we would like to use the methods proposed in this thesis.

Introducing spatial or temporal distance in fully-connected G-CRFs. While the G-CRF models for fully-connected graphs that we have proposed in this thesis (in Chap. 3 and 4) are capable of discovering and learning all the pairwise interactions directly from the data, they do not include a term that captures spatial distance between pixels in the same frame, or even temporal distance between pixels in different frames. As such the long-range connections share the same importance as the short-range connections. While this might not be an issue when we pretrain the embeddings in a supervised fashion, this could be a limitation when we want to train the pairwise terms without any additional supervision. Without any additional supervision, it might make sense to allow the model to place more confidence in the immediate context, i.e. short-range interactions to have more weight than long-range interactions. To this end, we would like to complement the pairwise term coming from the embeddings with spatial or temporal distance. There are a couple of ways to achieving this: (i) either add a 'constant' distance based pairwise matrix to A during inference, or (ii) try to learn embeddings on top of the spatio-temporal coordinates, alongside appearance based embeddings, and train them with additional supervision so they capture the spatio-temporal distance between pixels. In connection with this, we are trying to approximate spatio-temporal distance between two pixels using feature embeddings as in [Vedaldi 2010, Li 2010]. We would like to explore further this direction of research.

Using G-CRFs in other domains. Having developed the theory for efficient inference and learning for sparse and fully-connected graphical models, we would like to apply these techniques to other domains such as regression and detection. We would like to employ our structured prediction models for tasks such as image denoising, prediction of depth and normals, object detection, scene classification and so on. G-CRFs are naturally suitable to all these tasks owing to their continuous nature. Training of these models would require further exploration into initialization or pre-training of the pairwise terms, and this is one avenue we would like to explore.

Semi- and Weakly-Supervised Methods. We would also like to use our models in the semi-supervised and weakly-supervised settings. We believe these problems will become increasingly important as the performance of fully-supervised methods saturates. Annotation is expensive and the amount of un-annotated data is enormous. The next natural step is to use data from all available sources without relying on exhaustive manual labeling. To this end, we can exploit the pairwise affinities coming from a pretrained network to cluster patches in unseen images. This will also allow discovery of novel categories, and could be used to speed up manual annotation. Clustering could allow us to transfer annotations from a la-

beled exemplar image to similar images, followed by a refinement labeling phase if necessary. Furthermore, we could train a classification network containing the G-CRF structured prediction module using image-level labels alone, and use the pairwise affinities to capture patch-level similarities between images.

Extension to Mixture Models. While the G-CRF energy function is unimodal and symmetric by definition, one possible extension is to use a mixture of G-CRFs. However, mixture models are typically learnt iteratively using expectation maximization, and hence computationally expensive. Rather than relying on expectation maximization to learn all G-CRF parameters, we could learn the mixture weights alongside each component via back-propagation. Coming up with strategies to enable efficient training of such models is another research problem which needs to be looked at.

Other Structured Prediction Approaches based on G-CRFs. We have also been working towards simplifying the G-CRF formulation by directly estimating the inverse of the G-CRF precision matrix. In other words, we express the energy function as $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T C^{-1}\mathbf{x} - \mathbf{b}^T\mathbf{x}$. With this simplification, inference becomes $C^{-1}\mathbf{x} = \mathbf{b}$, i.e. $\mathbf{x} = C\mathbf{b}$. Thus, to perform inference for this system, we do not need to solve system of linear equations, as inference and update rules only involve matrix-vector products. This can be interpreted as seeing the output as a linear combination of all inputs. While this structured prediction approach is significantly faster than G-CRF inference (which relies on iterative conjugate gradient computations), it requires us to impose a structure on the inverse of the precision matrix for compact representation (as opposed to imposing structure on the precision matrix). This direction of research is being pursued in several independent works [Wang 2017, Bertasius 2016].

Further, rather than using the aformentioned inverted precision matrix C for mapping inputs $\mathbf{b}$ to outputs $\mathbf{x}$, we could use a non-linear function, such as the Softmax on C. Thus, in this paradigm, the prediction is thus seen as a convex combination of all inputs. While we move away from the G-CRF interpretation in this model, this could be seen as non-linear projection from one space to another, and parallels could be drawn between this approach and others which express a point as a combination of its neighbours (such as the Locally-Linear Embeddings [Saul 2000]). We hypothesize that the outstanding performance of modern architectures such as the residual networks or inception networks is partly due to the presence of multiple paths: these architectures have parallel branches of computation which are merged together, and can thus be seen as ensembles of shallow(er) networks [Veit 2016]. Thus, a non-linear function on C such as the Softmax could be seen as a generalized convolution operation which can discover a latent 'attention' model, allowing an exponential number of receptive fields for each filter, thereby introducing exponential paths in the architecture. We would like to explore further this line of research.

Extension of Spatio-Temporal G-CRFs. With regard to capturing spatio-temporal pairwise terms, we would like to incorporate optical flow techniques in this framework as they provide a natural way of capturing temporal correspondence. This would allow us to exploit both motion-based and appearance-based cues for video understanding. Finally, we would like to exploit this framework for regression and detection tasks, and in the weakly-supervised setting as well.

Sampling from G-CRFs. Papandreou *et al.* [Papandreou 2010,Papandreou 2011] demonstrate that sampling from a Gaussian MRF is equivalent to adding random noise to the MRF parameters and performing inference. With this knowledge, we can sample from G-CRFs by injecting noise into the G-CRF parameters delivered by a CNN and performing inference. We can use this strategy to make the the classifier more robust by using the samples from the G-CRF for data augmentation. Further, by introducing G-CRFs in generative models such as Generative Adversarial Networks [Goodfellow 2014], we can induce diversity in their outputs by carefully designing the distribution of noise. This is another direction of research we would like to explore.

Structure in the Loss. In this thesis, we use two evaluation metrics for the segmentation / saliency estimation tasks we solve: (i) mean Intersection over Union, and (ii) maximal F-measure. Both these metrics are structured i.e. they are defined globally for an image or a dataset and cannot be decomposed over pixels or patches in the image. As such these metrics are non-differentiable, and cannot be directly optimized during training via standard back-propagation. While several recent works propose strategies to directly optimize sctructured losses during training [Yue 2007,Dokania 2014,Ahmed 2015,Berman 2017] by exploiting the structure in the loss, in this thesis we have focussed instead on exploiting structure in the output by modeling interdependencies between output variables. We have used the standard softmax cross-entropy loss during training. In the future, we would like to use structured losses to train our methods.

Bibliography

[Adi 2017] Yossi Adi, Joseph Keshet, Emily Cibelli and Matthew Goldrick. *Sequence segmentation using joint RNN and structured prediction models.* In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on, pages 2422–2426. IEEE, 2017. (Cited on page 81.)

[Ahmed 2015] Faruk Ahmed, Dany Tarlow and Dhruv Batra. *Optimizing expected intersection-over-union with candidate-constrained crfs.* In Proceedings of the IEEE International Conference on Computer Vision, pages 1850–1858, 2015. (Cited on page 107.)

[Arnab 2016] Anurag Arnab, Sadeep Jayasumana, Shuai Zheng and Philip HS Torr. *Higher order conditional random fields in deep neural networks.* In European Conference on Computer Vision, pages 524–540. Springer, 2016. (Cited on pages 19 and 27.)

[Badrinarayanan 2015] V. Badrinarayanan, A. Kendall and R. Cipolla. *Segnet: A deep convolutional encoder-decoder architecture for image segmentation.* In ArXiV *CoRR*, abs/1511.00561, 2015. (Cited on pages 88 and 92.)

[Barron 2016] Jonathan T Barron and Ben Poole. *The fast bilateral solver.* In ECCV, 2016. (Cited on pages 20, 23, 29 and 40.)

[Bengio 2013] Yoshua Bengio, Aaron Courville and Pascal Vincent. *Representation Learning: A Review and New Perspectives.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pages 1798–1828, August 2013. (Cited on page 10.)

[Berman 2017] Maxim Berman and Matthew B. Blaschko. *Optimization of the Jaccard index for image segmentation with the Lovász hinge.* CoRR, vol. abs/1705.08790, 2017. (Cited on page 107.)

[Bertasius 2016] Gedas Bertasius, Lorenzo Torresani, Stella X. Yu and Jianbo Shi. *Convolutional Random Walk Networks for Semantic Image Segmentation.* CoRR, vol. abs/1605.07681, 2016. (Cited on page 106.)

[Boyd 2004] Stephen Boyd and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004. (Cited on page 21.)

[Bratieres 2015] Sébastien Bratieres, Novi Quadrianto and Zoubin Ghahramani. *GPstruct: Bayesian structured prediction using Gaussian processes.* IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 7, pages 1514–1520, 2015. (Cited on pages 23, 29 and 81.)

[Breiman 2001] Leo Breiman. *Random forests.* Machine learning, vol. 45, no. 1, pages 5–32, 2001. (Cited on page 10.)

[Brostow 2017] G. J. Brostow, J. Shotton, J. Fauqueur and R. Cipolla. *Segmentation and recognition using structure from motion point clouds.* In ECCV, 2017. (Cited on page 88.)

[Caelles 2017] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers and L. Van Gool. *One-Shot Video Object Segmentation.* In Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on pages 81 and 91.)

[Campbell 2013] Neill D. F. Campbell, Kartic Subr and Jan Kautz. *Fully-Connected CRFs with Non-Parametric Pairwise Potentials.* In CVPR, 2013. (Cited on page 26.)

[Chatfield 2011] Ken Chatfield, Victor Lempitsky, Andrea Voedaldi and Andrew Zisserman. *The devil is in the details: an evaluation of recent feature encoding methods.* In BMVC, 2011. (Cited on page 10.)

[Chatfield 2014] Ken Chatfield, K. Simonyan, A. Vedaldi and A. Zisserman. *Return of the Devil in the Details: Delving Deep into Convolutional Nets.* In British Machine Vision Conference, 2014. (Cited on page 10.)

[Chen 2014a] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy and Alan L Yuille. *Semantic image segmentation with deep convolutional nets and fully connected crfs.* arXiv preprint arXiv:1412.7062, 2014. (Cited on pages 1, 2, 13, 14, 17, 18, 20, 22, 23, 24, 29, 37, 40, 46, 47, 48, 50, 62, 71 and 92.)

[Chen 2014b] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun and A. Yuille. *Detect what you can: Detecting and representing objects using holistic models and body parts.* In CVPR, 2014. (Cited on pages 72 and 73.)

[Chen 2015a] L.-C. Chen, A. G. Schwing, A. L. Yuille and R. Urtasun. *Learning Deep Structured Models.* In ICML, 2015. (Cited on pages 20, 24 and 28.)

[Chen 2015b] Liang-Chieh Chen, George Papandreou, Kevin Murphy and Alan L Yuille. *Weakly- and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation.* ICCV, 2015. (Cited on pages 14, 20, 23, 24, 29, 50 and 51.)

[Chen 2016a] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy and Alan L Yuille. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.* arXiv:1606.00915, 2016. (Cited on pages 51, 62, 70, 71, 72, 73 and 89.)

[Chen 2016b] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu and Alan L. Yuille. *Attention to Scale: Scale-aware Semantic Image Segmentation.* CVPR, 2016. (Cited on pages 46, 72 and 73.)

[Chen 2017] Liang-Chieh Chen, George Papandreou, Florian Schroff and Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation.* CoRR, vol. abs/1706.05587, 2017. (Cited on page 72.)

[Cooper 1990] Gregory F Cooper. *The computational complexity of probabilistic inference using Bayesian belief networks.* Artificial intelligence, vol. 42, no. 2-3, pages 393–405, 1990. (Cited on page 20.)

[Cordts 2016] M. Cordts, M. Omran, S. Ramos, T. Scharwachter, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele. *The Cityscapes dataset for semantic urban scene understanding.* CVPR, 2016. (Cited on page 92.)

[Cortes 1995] Corinna Cortes and Vladimir Vapnik. *Support-vector networks.* Machine learning, vol. 20, no. 3, pages 273–297, 1995. (Cited on page 10.)

[Dalal 2005] Navneet Dalal and Bill Triggs. *Histograms of oriented gradients for human detection.* In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005. (Cited on page 10.)

[Deng 2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. *ImageNet: A Large-Scale Hierarchical Image Database.* In CVPR09, 2009. (Cited on pages 10 and 14.)

[Desmaison 2016] Alban Desmaison, Rudy Bunel, Pushmeet Kohli, Philip HS Torr and M Pawan Kumar. *Efficient continuous relaxations for dense CRF.* In European Conference on Computer Vision, pages 818–833. Springer, 2016. (Cited on page 26.)

[Dokania 2014] P. K. Dokania, A. Behl, C. V. Jawahar and P. K. Kumar. *Learning to Rank using High-Order Information.* ECCV, 2014. (Cited on page 107.)

[Donahue 2015] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko and Trevor Darrell. *Long-term recurrent convolutional networks for visual recognition and description.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2625–2634, 2015. (Cited on page 81.)

[Fackler 2005] Paul L. Fackler. *Notes on Matrix Calculus.* 2005. (Cited on pages 57, 65 and 101.)

[Farabet 2012] Clement Farabet, Camille Couprie, Laurent Najman and Yann Lecun. *Scene parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers.* In ICML, 2012. (Cited on page 12.)

[Farabet 2013] Clement Farabet, Camille Couprie, Laurent Najman and Yann LeCun. *Learning Hierarchical Features for Scene Labeling.* PAMI, 2013. (Cited on page 12.)

[Fathi 2017] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama and Kevin P. Murphy. *Semantic Instance Segmentation via Deep Metric Learning.* CoRR, vol. abs/1703.10277, 2017. (Cited on page 60.)

[Fu 2017] Jun Fu, Jing Liu, Yuhang Wang and Hanqing Lu. *Stacked Deconvolutional Network for Semantic Segmentation.* CoRR, vol. abs/1708.04943, 2017. (Cited on page 14.)

[Gadde 2017] Raghudeep Gadde, Varun Jampani and Peter V. Gehler. *Semantic Video CNNs through Representation Warping.* In ICCV, 2017. (Cited on pages 81 and 82.)

[Ganin 2014] Y. Ganin and V. Lempitsky. *N^4-fields: Neural network nearest neighbor fields for image transforms.* In ACCV, 2014. (Cited on page 12.)

[Girshick 2014] B. Hariharan P. Arbeláez R. Girshick and J. Malik. *Simultaneous detection and segmentation.* In ECCV, 2014. (Cited on page 12.)

[Golub 1996] Gene H. Golub, Van Loan and Charles F. *Matrix Computations.* vol. 3, no. 1-2, page 510, January 1996. (Cited on pages 44 and 46.)

[Goodfellow 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. *Generative adversarial nets.* In Advances in neural information processing systems, pages 2672–2680, 2014. (Cited on page 107.)

[Grady 2006] Leo Grady. *Random walks for image segmentation.* In PAMI, 2006. (Cited on page 45.)

[Gupta 2014] S. Gupta, R. Girshick, P. Arbelaez and J. Malik. *Learning rich features from RGB-D images for object detection and segmentation.* In ECCV, 2014. (Cited on page 12.)

[Hahnloser 2000] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas and H Sebastian Seung. *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit.* Nature, vol. 405, no. 6789, page 947, 2000. (Cited on pages 11 and 12.)

[Hariharan 2015] Bharath Hariharan, Pablo Arbeláez, Ross Girshick and Jitendra Malik. *Hypercolumns for Object Segmentation and Fine-grained Localization.* In CVPR, 2015. (Cited on pages 12, 48 and 70.)

[Harley 2015] Adam Harley, Konstantinos Derpanis and Iasonas Kokkinos. *Deep networks for saliency detection via local estimation and global search.* In CVPR, 2015. (Cited on page 60.)

[Harley 2017] Adam W. Harley, Konstantinos G. Derpanis and Iasonas Kokkinos. *Learning Dense Convolutional Embeddings for Semantic Segmentation.* In ICCV, 2017. (Cited on page 60.)

[He 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Deep Residual Learning for Image Recognition.* In CVPR, 2016. (Cited on pages 1, 10, 13, 15 and 51.)

[He 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick. *Mask R-CNN.* ICCV, 2017. (Cited on page 90.)

[Huang 2017] Gao Huang, Zhuang Liu, Kilian Q Weinberger and Laurens van der Maaten. *Densely connected convolutional networks.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. (Cited on page 14.)

[Ilg 2017] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and T. Brox. *FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul 2017. (Cited on page 81.)

[Ioffe 2015] Sergey Ioffe and Christian Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift.* In International conference on machine learning, pages 448–456, 2015. (Cited on page 48.)

[Jain 2017] Suyog Jain, Bo Xiong and Kristen Grauman. *FusionSeg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos.* arXiv preprint arXiv:1701.05384, 2017. (Cited on page 81.)

[Jampani 2016] Varun Jampani, Martin Kiefel and Peter V. Gehler. *Learning Sparse High Dimensional Filters: Image Filtering, Dense CRFs and Bilateral Neural Networks.* In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 4452–4461, 2016. (Cited on pages 22, 23, 24, 26, 29 and 39.)

[Jampani 2017] Varun Jampani, Raghudeep Gadde and Peter V. Gehler. *Video Propagation Networks.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. (Cited on page 81.)

[Jancsary 2012] Jeremy Jancsary, Sebastian Nowozin, Toby Sharp and Carsten Rother. *Regression Tree Fields - An Efficient, Non-parametric Approach to Image Labeling Problems.* In CVPR, 2012. (Cited on pages 10, 22, 34, 37, 40 and 41.)

[Jégou 2017] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero and Yoshua Bengio. *The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation.* In Computer Vision and Pattern

Recognition Workshops (CVPRW), 2017 IEEE Conference on, pages 1175–1183. IEEE, 2017. (Cited on pages 88 and 92.)

[Jia 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama and Trevor Darrell. *Caffe: Convolutional Architecture for Fast Feature Embedding.* arXiv preprint arXiv:1408.5093, 2014. (Cited on page 67.)

[Jin 2016] Xiaojie Jin, Xin Li, Huaxin Xiao, Xiaohui Shen, Zhe Lin, Jimei Yang, Yunpeng Chen, Jian Dong, Luoqi Liu, Zequn Jie, Jiashi Feng and Shuicheng Yan. *Video Scene Parsing with Predictive Feature Learning.* CoRR, vol. abs/1612.00119, 2016. (Cited on page 81.)

[Karpathy 2014] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar and Li Fei-Fei. *Large-scale video classification with convolutional neural networks.* In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 1725–1732, 2014. (Cited on page 81.)

[Kendall 2015] A. Kendall, V. Badrinarayanan, and R. Cipolla. *Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding.* In ArXiV *CoRR*, abs/1511.02680, 2015. (Cited on page 92.)

[Khoreva 2017] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox and Bernt Schiele. *Lucid Data Dreaming for Object Tracking.* arXiv preprint arXiv:1703.09554, 2017. (Cited on page 81.)

[Kokkinos 2016] Iasonas Kokkinos. *Pushing the Boundaries of Boundary Detection using Deep Learning.* In ICLR, 2016. (Cited on pages 71 and 72.)

[Kokkinos 2017] Iasonas Kokkinos. *UberNet: A Universal CNN for the joint treatment of Low-, Mid-, and High- Level Vision Problems.* In CVPR, 2017. (Cited on pages 73 and 74.)

[Koller 2007] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques.* In MIT Press, 2007. (Cited on page 19.)

[Kolmogorov 2006] Vladimir Kolmogorov. *Convergent tree-reweighted message passing for energy minimization.* IEEE transactions on pattern analysis and machine intelligence, vol. 28, no. 10, pages 1568–1583, 2006. (Cited on page 28.)

[Kolmogorov 2007] Vladimir Kolmogorov and Carsten Rother. *Minimizing nonsubmodular functions with graph cuts-a review.* IEEE transactions on pattern analysis and machine intelligence, vol. 29, no. 7, 2007. (Cited on page 28.)

[Krähenbühl 2011] Philipp Krähenbühl and Vladlen Koltun. *Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials.* In NIPS, 2011. (Cited on pages 19, 20, 24, 25, 26 and 87.)

[Krizhevsky 2012] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks.* In NIPS, 2012. (Cited on pages 10 and 13.)

[Kundu 2016] Abhijit Kundu, Vibhav Vineet and Vladlen Koltun. *Feature space optimization for semantic video segmentation.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3168–3175, 2016. (Cited on pages 23, 29, 80, 81, 88 and 92.)

[Lafferty 2001] John Lafferty, Andrew McCallum and Fernando CN Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data.* 2001. (Cited on page 18.)

[Laina 2016] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari and Nassir Navab. *Deeper depth prediction with fully convolutional residual networks.* In 3D Vision, pages 239–248. IEEE, 2016. (Cited on page 14.)

[Lecun 1998] Yann Lecun, Leon Bottou, Yoshua Bengio and Patrick Haffner. *Gradient-based learning applied to document recognition.* In Proceedings of the IEEE, 1998. (Cited on pages 1, 11 and 12.)

[Len 2014] Len. *Microsoft COCO: Common objects in context.* In ECCV, 2014. (Cited on pages 10 and 48.)

[Li 2010] Fuxin Li, Catalin Ionescu and Cristian Sminchisescu. *Random Fourier approximations for skewed multiplicative histogram kernels.* In Joint Pattern Recognition Symposium, pages 262–271. Springer, 2010. (Cited on page 105.)

[Li 2014] Y. Li, X. Hou, C. Koch, J. M. Rehg and A. L. Yuille. *The secrets of salient object segmentation.* In CVPR, 2014. (Cited on pages 73 and 74.)

[Li 2015] G. Li and Y. Yu. *Visual saliency based on multiscale deep features.* In CVPR, 2015. (Cited on pages 73 and 74.)

[Li 2016] G. Li and Y. Yu. *Deep contrast learning for salient object detection.* In CVPR, 2016. (Cited on pages 73 and 74.)

[Li 2017] Xiaoxiao Li, Yuankai Qi, Zhe Wang, Kai Chen, Ziwei Liu, Jianping Shi, Ping Luo, Xiaoou Tang and Chen Change Loy. *Video Object Segmentation with Re-identification.* arXiv preprint arXiv:1708.00197, 2017. (Cited on page 81.)

[Liang 2016a] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin and Shuicheng Yan. *Semantic object parsing with graph lstm.* In European Conference on Computer Vision, pages 125–143. Springer, 2016. (Cited on pages 72 and 73.)

[Liang 2016b] Xiaodan Liang, Xiaohui Shen, Donglai Xiang, Jiashi Feng, Liang Lin and Shuicheng Yan. *Semantic Object Parsing With Local-Global Long Short-Term Memory.* In CVPR, 2016. (Cited on pages 72 and 73.)

[Lin 2016] Guosheng Lin, Chunhua Shen, Ian D. Reid and Anton van den Hengel. *Efficient piecewise training of deep structured models for semantic segmentation.* CVPR, 2016. (Cited on pages 22, 23, 24, 27, 29, 40, 71 and 72.)

[Liu 2015a] F. Liu, C. Shen, and G. Lin. *Deep Convolutional Neural Fields for Depth Estimation from a Single Image.* In CVPR, 2015. (Cited on pages 22, 23, 29 and 40.)

[Liu 2015b] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy and Xiaoou Tang. *Semantic image segmentation via deep parsing network.* In CVPR, pages 1377–1385, 2015. (Cited on pages 19, 23, 24, 27, 29, 40, 71 and 72.)

[Liu 2016] Fayao Liu, Chunhua Shen, Guosheng Lin and Ian Reid. *Learning depth from single monocular images using deep convolutional neural fields.* PAMI, vol. 38, no. 10, pages 2024–2039, 2016. (Cited on page 12.)

[Long 2015] Jonathan Long, Evan Shelhamer and Trevor Darrell. *Fully convolutional networks for semantic segmentation.* In CVPR, pages 3431–3440, 2015. (Cited on pages 1, 12, 13, 71 and 92.)

[Lowe 2004] David G Lowe. *Distinctive image features from scale-invariant keypoints.* International journal of computer vision, vol. 60, no. 2, pages 91–110, 2004. (Cited on page 10.)

[Matan 1992] Ofer Matan, Christopher JC Burges, Yann LeCun and John S Denker. *Multi-digit recognition using a space displacement neural network.* In Advances in neural information processing systems, pages 488–495, 1992. (Cited on page 12.)

[Mostajabi 2015] M. Mostajabi, P. Yadollahpour and G. Shakhnarovich. *Feedforward semantic segmentation with zoom-out features.* In CVPR, 2015. (Cited on page 12.)

[Newell 2016] Alejandro Newell and Jia Deng. *Associative Embedding: End-to-End Learning for Joint Detection and Grouping.* CoRR, vol. abs/1611.05424, 2016. (Cited on page 60.)

[Nilsson 2016] David Nilsson and Cristian Sminchisescu. *Semantic Video Segmentation by Gated Recurrent Flow Propagation.* CoRR, vol. abs/1612.08871, 2016. (Cited on page 81.)

[Noh 2015] H. Noh, S. Hong, and B. Han. *Learning deconvolution network for semantic segmentation.* In *arXiv preprint arXiv:1505.04366*, 2015. (Cited on page 92.)

[Nowozin 2011] Sebastian Nowozin, Christoph H Lampert*et al. Structured learning and prediction in computer vision.* Foundations and Trends® in Computer Graphics and Vision, vol. 6, no. 3–4, pages 185–365, 2011. (Cited on page 16.)

[Papandreou 2010] G. Papandreou and A. Yuille. *Gaussian Sampling by Local Perturbations.* In Proc. Int. Conf. on Neural Information Processing Systems (NIPS), pages 1858–1866, Vancouver, B.C., Canada, December 2010. (Cited on page 107.)

[Papandreou 2011] G. Papandreou and A. Yuille. *Perturb-and-MAP Random Fields: Using Discrete Optimization to Learn and Sample from Energy Models.* In Proc. IEEE Int. Conf. on Computer Vision (ICCV), pages 193–200, Barcelona, Spain, November 2011. (Cited on page 107.)

[Perazzi 2016] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross and A. Sorkine-Hornung. *A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation.* In Computer Vision and Pattern Recognition, 2016. (Cited on page 88.)

[Perazzi 2017] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele and A.Sorkine-Hornung. *Learning Video Object Segmentation from Static Images.* In Computer Vision and Pattern Recognition, 2017. (Cited on pages 81 and 91.)

[Perronnin 2010] Florent Perronnin, Jorge Sánchez and Thomas Mensink. *Improving the fisher kernel for large-scale image classification.* In European conference on computer vision, pages 143–156. Springer, 2010. (Cited on page 10.)

[Pinheiro 2014] P. H. Pinheiro and R. Collobert. *Recurrent convolutional neural networks for scene labeling.* In ICML, 2014. (Cited on page 12.)

[Pont-Tuset 2017] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung and Luc Van Gool. *The 2017 DAVIS Challenge on Video Object Segmentation.* arXiv:1704.00675, 2017. (Cited on page 88.)

[Press 1992] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery. Numerical recipes in c, 2nd edition. Cambridge University Press, 1992. (Cited on pages 44 and 45.)

[Roth 1996] Dan Roth. *On the hardness of approximate reasoning.* Artificial Intelligence, vol. 82, no. 1-2, pages 273–302, 1996. (Cited on page 20.)

[Rue 2005] H. Rue and L. Held. Gaussian Markov random fields: Theory and applications, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 2005. (Cited on pages 20 and 46.)

[Saul 2000] Lawrence K Saul and Sam T Roweis. *An introduction to locally linear embedding*. 2000. (Cited on page 106.)

[Schmidhuber 2015] J. Schmidhuber. *Deep Learning in Neural Networks: An Overview*. Neural Networks, vol. 61, pages 85–117, 2015. (Cited on page 10.)

[Schwing 2015] Alexander G Schwing and Raquel Urtasun. *Fully connected deep structured networks*. arXiv preprint arXiv:1503.02351, 2015. (Cited on page 26.)

[Sermanet 2013] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala and Yann LeCun. *Pedestrian detection with unsupervised multi-stage feature learning*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3626–3633, 2013. (Cited on page 12.)

[Shelhamer 2016] Evan Shelhamer, Kate Rakelly, Judy Hoffman and Trevor Darrell. *Clockwork Convnets for Video Semantic Segmentation*. CoRR, vol. abs/1608.03609, 2016. (Cited on page 81.)

[Shewchuk 1994] Jonathan Richard Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. In `https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`, 1994. (Cited on pages 42, 55, 62 and 84.)

[Shotton 2009] Jamie Shotton, John Winn, Carsten Rother and Antonio Criminisi. *Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context*. International Journal of Computer Vision, vol. 81, no. 1, pages 2–23, 2009. (Cited on page 10.)

[Simonyan 2015] Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. ICLR, 2015. (Cited on pages 10, 13 and 50.)

[Sutton 2012a] Charles Sutton and Andrew McCallum. *Piecewise training for undirected models*. arXiv preprint arXiv:1207.1409, 2012. (Cited on page 27.)

[Sutton 2012b] Charles Sutton, Andrew McCallum*et al. An introduction to conditional random fields*. Foundations and Trends® in Machine Learning, vol. 4, no. 4, pages 267–373, 2012. (Cited on pages 16 and 20.)

[Tappen 2007] Marshall F. Tappen, Ce Liu, Edward H. Adelson and William T. Freeman. *Learning Gaussian Conditional Random Fields for Low-Level Vision*. In CVPR, 2007. (Cited on pages 20, 37 and 40.)

[Tappen 2008] Marshall F Tappen, Kegan GG Samuel, Craig V Dean and David M Lyle. *The logistic random field - A convenient graphical model for learning parameters for MRF-based labeling.* In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008. (Cited on page 22.)

[Taskar 2004] Ben Taskar, Vassil Chatalbashev and Daphne Koller. *Learning associative Markov networks.* In Proceedings of the twenty-first international conference on Machine learning, page 102. ACM, 2004. (Cited on page 22.)

[Vedaldi 2010] A. Vedaldi and A. Zisserman. *Efficient Additive Kernels via Explicit Feature Maps.* In Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2010. (Cited on page 105.)

[Veit 2016] Andreas Veit, Michael J Wilber and Serge Belongie. *Residual networks behave like ensembles of relatively shallow networks.* In Advances in Neural Information Processing Systems, pages 550–558, 2016. (Cited on page 106.)

[Vemulapalli 2016a] Raviteja Vemulapalli, Oncel Tuzel and Ming-Yu Liu. *Deep Gaussian Conditional Random Field Network: A Model-based Deep Network for Discriminative Denoising.* In CVPR, 2016. (Cited on page 40.)

[Vemulapalli 2016b] Raviteja Vemulapalli, Oncel Tuzel, Ming-Yu Liu and Rama Chellapa. *Gaussian Conditional Random Field Network for Semantic Segmentation.* In CVPR, June 2016. (Cited on pages 22, 23, 24, 27, 29, 37, 40 and 46.)

[Vineet 2013] Vibhav Vineet, Jonathan Warrell, Paul Sturgess and Philip HS Torr. *Improved Initialization and Gaussian Mixture Pairwise Terms for Dense Random Fields with Mean-field Inference.* 2013. (Cited on page 26.)

[Visin 2016] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci and A. Courville. *Reseg: A recurrent neural network-based model for semantic segmentation.* In CVPR workshop, 2016. (Cited on page 92.)

[Voigtlaender 2017] Paul Voigtlaender and Bastian Leibe. *Online Adaptation of Convolutional Neural Networks for Video Object Segmentation.* In BMVC, 2017. (Cited on pages 81, 88, 91 and 92.)

[Vu 2015] Tuan-Hung Vu, Anton Osokin and Ivan Laptev. *Context-aware CNNs for person head detection.* In ICCV, pages 2893–2901, 2015. (Cited on pages 2, 23, 24, 28, 29 and 40.)

[Wainwright 2008] Martin J. Wainwright and Michael I. Jordan. *Graphical Models, Exponential Families, and Variational Inference.* Found. Trends Mach. Learn., vol. 1, no. 1-2, pages 136–138, January 2008. (Cited on page 46.)

[Wang 2015a] K. Wang, L. Lin, J. Lu, C. Li and K. Shi. *PISA: pixelwise image saliency by aggregating complementary appearance contrast measures with edge-preserving coherence.* 2015. (Cited on page 74.)

[Wang 2015b] L. Wang, H. Lu, X. Ruan and M. Yang. *Deep networks for saliency detection via local estimation and global search.* In CVPR, 2015. (Cited on pages 73 and 74.)

[Wang 2016] Shenlong Wang, Sanja Fidler and Raquel Urtasun. *Proximal deep structured models.* In Advances in Neural Information Processing Systems, pages 865–873, 2016. (Cited on page 26.)

[Wang 2017] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta and Kaiming He. *Non-local Neural Networks.* CoRR, vol. abs/1711.07971, 2017. (Cited on page 106.)

[Wolf 1994] Ralph Wolf and John C Platt. *Postal address block location using a convolutional locator network.* In Advances in Neural Information Processing Systems, pages 745–752, 1994. (Cited on page 12.)

[Xia 2016] F. Xia, P. Wang, L. Chen and A. L. Yuille. *Zoom better to see clearer: Human part segmentation with auto zoom net.* In ECCV, 2016. (Cited on pages 72 and 73.)

[Xie 2015] Saining Xie and Zhuowen Tu. *Holistically-nested edge detection.* In ICCV, pages 1395–1403, 2015. (Cited on page 12.)

[Yu 2016] Fisher Yu and Vladlen Koltun. *Multi-scale context aggregation by dilated convolutions.* ICLR, 2016. (Cited on pages 13 and 92.)

[Yu 2017] Zhiding Yu, Chen Feng, Ming-Yu Liu and Srikumar Ramalingam. *CASENet: Deep Category-Aware Semantic Edge Detection.* arXiv preprint arXiv:1705.09759, 2017. (Cited on page 14.)

[Yue 2007] Yisong Yue, Thomas Finley, Filip Radlinski and Thorsten Joachims. *A support vector method for optimizing average precision.* In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pages 271–278. ACM, 2007. (Cited on page 107.)

[Zagoruyko 2016] Sergey Zagoruyko and Nikos Komodakis. *Wide Residual Networks.* In BMVC, 2016. (Cited on page 14.)

[Zhao 2015] R. Zhao, W. Ouyang, H. Li and X. Wang. *Saliency detection by multi-context deep learning.* In CVPR, 2015. (Cited on pages 73 and 74.)

[Zhao 2016] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang and Jiaya Jia. *Pyramid Scene Parsing Network.* CoRR, vol. abs/1612.01105, 2016. (Cited on pages 14, 72, 82 and 92.)

[Zheng 2015] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang and Philip Torr. *Conditional Random Fields as Recurrent Neural Networks*. In ICCV, 2015. (Cited on pages 2, 19, 22, 23, 24, 26, 27, 29, 37, 40, 46, 50, 51, 71, 72 and 86.)

[Zhu 2016] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan and Yichen Wei. *Deep Feature Flow for Video Recognition*. CoRR, vol. abs/1611.07715, 2016. (Cited on page 81.)