Thesis for the Degree of Licentiate of Engineering

End-to-End Learning of Deep Structured Models for Semantic Segmentation

Måns Larsson

Department of Electrical Engineering
Chalmers University of Technology
Göteborg, Sweden 2018

End-to-End Learning of Deep Structured Models for Semantic Segmentation
MÅNS LARSSON

Technical report no R002/2018 ISSN 1403-266X
Computer Vision and Image Analysis group
Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
SE–412 96 Göteborg, Sweden

Typeset by the author using LaTeX.

Chalmers Reproservice
Göteborg, Sweden 2018

End-to-End Learning of Deep Structured Models for Semantic Segmentation
Måns Larsson
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The task of semantic segmentation aims at understanding an image at a pixel level. This means assigning a label to each pixel of an image, describing the object it is depicting. Due to its applicability in many areas, such as autonomous vehicles, robotics and medical surgery assistance, semantic segmentation has become an essential task in image analysis. During the last few years a lot of progress have been made for image segmentation algorithms, mainly due to the introduction of deep learning methods, in particular the use of Convolutional Neural Networks (CNNs). CNNs are powerful for modeling complex connections between input and output data but lack the ability to directly model dependent output structures, for instance, enforcing properties such as label smoothness and coherence. This drawback motivates the use of Conditional Random Fields (CRFs), widely applied as a post-processing step in semantic segmentation.

This thesis summarizes the content of three papers, all of them presenting solutions to semantic segmentation problems. The applications have varied widely and several different types of data have been considered, ranging from 3D CT images to RGB images of horses. The main focus has been on developing robust and accurate models to solve these problems. The models consist of a CNN capable of learning complex image features coupled with a CRF capable of learning dependencies between output variables. Emphasis has been on creating models that are possible to train end-to-end, as well as developing corresponding optimization methods needed to enable efficient training. End-to-end training gives the CNN and the CRF a chance to learn how to interact and exploit complementary information to achieve better performance.

Keywords: Semantic segmentation, supervised learning, convolutional neural networks, conditional random fields, deep structured models.

Preface

I started my PhD with an attitude of "How am I going to come up with stuff that someone else have not already thought about?". The idea of actually adding something to the overwhelming mountain of knowledge already present in the field seemed absurd, and almost three years later it still kind of does. Despite this I somehow managed to arrive at the half-way milestone of a Licentiate thesis. Something that would not have been possible if not for all the help and support from my colleagues, friends and family.

I would like to start off by thanking my supervisor Fredrik Kahl for introducing me to the field of Computer Vision as well as guiding me through my PhD while constantly having to convince me that what I'm doing is actually worth publishing. Your input, encouragement and ideas have been invaluable during this time.

I am also grateful for my current and former colleagues in the Computer Vision and Image Analysis Group and the department of Electrical Engineering at Chalmers. Thank you Carl Toft, Olof Enqvist, Carl Olsson, Erik Stenborg, Lars Hammarstrand, Eskil Jörgensen, Mikaela Åhlén, Lucas Brynte, Yuhang Zhang and Jesús Briales García for you good company, great coffee break discussions and the occasional after work. A special thanks to Jennifer Alvén for starting her PhD a few months before me and hence constantly having to guide me through mine. In addition I would like to extend my thanks to my collaborators at the Torr Vision Group in Oxford, especially Anurag Arnab and Shuai Zheng who have been involved in the development of a big part of this thesis and introduced me to the wonderful yet frightening world of large-scale deep learning experiments.

Lastly, I would like to thank my friends and family. My family, for continuing to support me in what I'm doing, even though they stopped understanding what it is a long time ago. My friends, for brightening my spare time and constantly reminding me that there are more important and enjoyable things than work. My wonderful girlfriend, Maria, for all her support and constant encouragement in everything I do. Thank you all!

Included publications

Paper I M. Larsson, A. Arnab, F. Kahl, S. Zheng, and P. Torr. ”Revisiting Deep Structured Models in Semantic Segmentation with Gradient-Based Inference”. *Submitted to SIAM Journal on Imaging Sciences.* Extended version of paper (a).

Paper II M. Larsson, J. Alvén and F. Kahl. ”Max-Margin Learning of Deep Structured Models for Semantic Segmentation”.*Scandinavian Conference on Image Analysis (SCIA),* 28–40, 2017.

Paper III M. Larsson, Y. Zhang and F. Kahl ”Robust Abdominal Organ Segmentation Using Regional Convolutional Neural Networks”. *Submitted to Applied Soft Computing.* Extended version of paper (b).

Subsidiary publications

(a) M. Larsson, A. Arnab, F. Kahl, S, Zheng, and P. Torr. ”A Projected Gradient Descent Method for CRF Inference allowing End To End Training of Arbitrary Pairwise Potentials”. *11th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)* 2017.

(b) M. Larsson, Y. Zhang and F. Kahl ”Robust Abdominal Organ Segmentation Using Regional Convolutional Neural Networks”. *Scandinavian Conference on Image Analysis (SCIA),* 41–52, 2017.

(c) A. Arnab, S. Zheng, S. Jayasumana, B. Romera-Paredes, M. Larsson, A. Kirillov, B. Savchynskyy, C. Rother, F. Kahl, P. Torr. Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction". *IEEE Signal Processing Magazines Special Issue on: Deep Learning for Visual Understanding*, Volume: 35, Issue: 1, Jan. 2018, Pages 37-52 .

Abbreviations

ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
CRF	**C**onditional **R**andom **F**ield
DSM	**D**eep **S**tructured **M**odel
IoU	**I**ntersection **o**ver **U**nion
mIoU	**m**ean **I**ntersection **o**ver **U**nion
MRF	**M**arkov **R**andom **F**ield
ReLU	**Re**ctified **L**inear **U**nit
PGM	**P**robabilistic **G**raphical **M**odel
RNN	**R**ecurrent **N**eural **N**etwork
ROI	**R**egion **O**f **I**nterest

Contents

I Introductory Chapters

II Included Publications

Part I

Introductory Chapters

Chapter 1

Introduction

Understanding the content of an image is something that humans excel at. If I were to ask you to describe the objects present in an image you would in almost all cases manage that task effortlessly. However, if I ask you to state a set of rules to decide if an image contains a cat or a dog, you might have difficulties. Humans are so good at parsing and understanding visual scenes that we do not reflect on how we do it. Designing methods that do this automatically has however been proven to be a challenging problem and the field of Computer Vision is still very active.

Given an image, information can be extracted on different levels. This is illustrated in Figure 1.1 where a few examples of image analysis tasks of different detail are shown. The focus of this thesis is semantic segmentation, which aims at understanding an image on a pixel level. This means that we want to assign a label to each pixel, describing the object it is depicting. For example, going back to Figure 1.1 we have assigned the label "person" to the pixels colored pink and the label "dining table" to the pixels colored yellow.

Semantic segmentation has numerous of applications. In robotics, agents are usually needed to extract useful information and understand their environment to perform tasks such as navigation and manipulation of objects. This is something that can be achieved with a camera and a semantic segmentation algorithm. Also, autonomous vehicles require a precise understanding of their surrounding to be able to make safe decisions in traffic. Semantic segmentation algorithms are also useful for numerous applications in medical research and clinical care, such as computer aided diagnosis and surgery assistance. Since many of the images handled in medical applications are three dimensional manual segmentation is time consuming. Having an automatic method will in these cases save medical personnel a lot of time and be very helpful for time-critical tasks such as surgery planning.

Traditionally, semantic segmentation algorithms have been approached by extracting some type of hand-crafted image features from the image. These features could be something as simple as color gradient or a more complex function of the

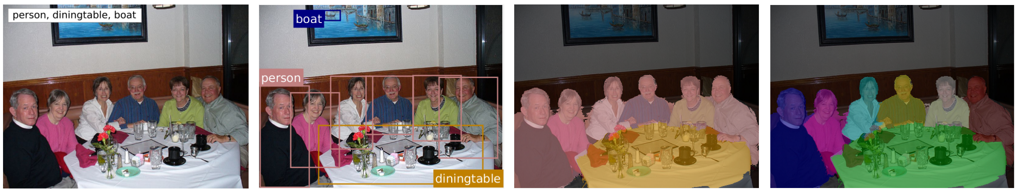

Figure 1.1: *Example of scene understanding tasks with increasing detail from left to right. From left: image captioning, object detection, semantic segmentation and instance segmentation. This thesis focuses on semantic segmentation. Image modified from [1].*

pixel values. A model relating these features to semantic classes is then created, or learnt from annotated examples, *i.e.* a set of images paired with their "true" semantic segmentations. During recent years most methods have moved from hand-crafted feature to using Convolutional Neural Networks (CNNs), capable of learning complex images features from data.

The introduction of CNNs for semantic segmentation meant a large improvement in performance and we are now able to create models that are fairly good at understanding the content of an image (given that it is similar to the images it has been trained on). A drawback with a CNN is however that they cannot explicitly take the dependencies between output variables, *i.e.* how the label of one pixel depends on the label of the output pixels, into account. This can however be done using Conditional Random Fields (CRFs), which have been used extensively for semantic segmentation. Because of this, many state-of-the-art methods combine a CNN and a CRF creating a Deep Structured Model (DSM) capable learning complex image features while still taking output dependencies into account.

The parameters of these DSMs are usually learnt from data. This learning can be easily achieved by using traditional deep learning methods to train the CNN. Then, using the output of the CNN to form the CRF, learning the weight of the CRF. This approach, commonly referred to as piece-wise training, is suboptimal since the parameters of the CNN is learnt while ignoring output dependencies. A better approach is to train the CNN and CRF jointly, or end-to-end. This gives the CNN and the CRF a chance to learn how to interact to achieve better results, a sketch of piece-wise and end-to-end training of a DSM is shown in Figure 1.2.

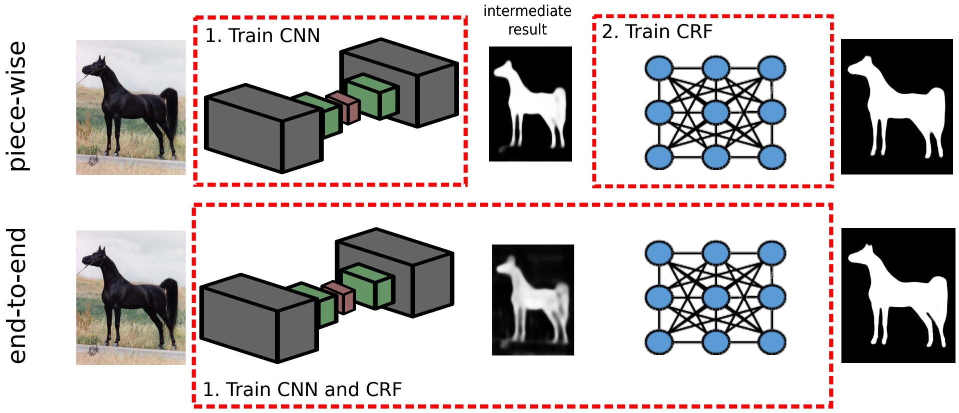

Figure 1.2: *Comparison of piece-wise and end-to-end training of a deep structured model (DSM). For the piece-wise training (above) the CNN is trained first, as a second step the parameters of the CRF is trained keeping the weights of the CNN fixed. During end-to-end training (below) the weights of the CNN and CRF are jointly trained, giving them a chance to learn how to interact to achieve better results.*

1.1 Thesis Scope

This thesis consists of three papers, all of them presenting solutions to semantic segmentation problems. The applications have varied widely and several different types of data have been considered, from 3D CT images to RGB images of horses to indoor scene understanding. The main focus has been on developing robust and accurate models to solve these problems. These models consist of a CNN capable of learning complex image features coupled with a CRF capable of learning dependencies between output variable, in our case pixel or voxel labels. Emphasis have been put on creating these type of models that also are possible to train end-to-end as well as the methods needed to enable this type of training.

1.2 Thesis Outline

The first part of this thesis consists of this introductory chapter, followed by Chapter 2 that provides some background knowledge needed to understand the papers included in this thesis as well as placing them in an academic context. Chapter 3 summarizes the work and contributions of this thesis as well as each paper separately. A brief discussion of future work is given in Chapter 4. Finally, the papers forming this thesis are appended in part II.

Chapter 2

Background

The focus of this thesis is to develop methods for semantic segmentation. Hence the background chapter will start off with a brief introduction to the problem of semantic segmentation. Afterwards a brief introduction to Convolutional Neural Networks (CNNs) as well as Conditional Random Fields (CRFs) will be given. Lastly, we will touch on the subject of end-to-end training of Deep Structured Models (DSMs), *i.e.* a combination of a CNN and a CRF. The sections in this chapter are by no means exhaustive but aim at giving the reader background knowledge enough to understand the included papers as well as place them in an academic context.

2.1 Semantic Segmentation

Semantic segmentation or scene labeling is the process of assigning each pixel of an image to the semantic class that it is depicting. The semantic class should depend on the surrounding information, or context, of the pixel. That means that we want to understand what the image is containing on a pixel level. What classes we are interested in dividing the image pixel in depends on the task and what information about our surrounding we are interested in. Given a set of images from a camera mounted on the front of a car we might want to classify each pixel as being one of ("driveable surface", "sidewalk", "pedestrian" *etc.*) while given a medical CT image of the abdomen we might want to classify pixels into different organs, or perhaps "tumour" and "not tumour". An example of visualizations of semantic segmentations is shown in Figure 2.1.

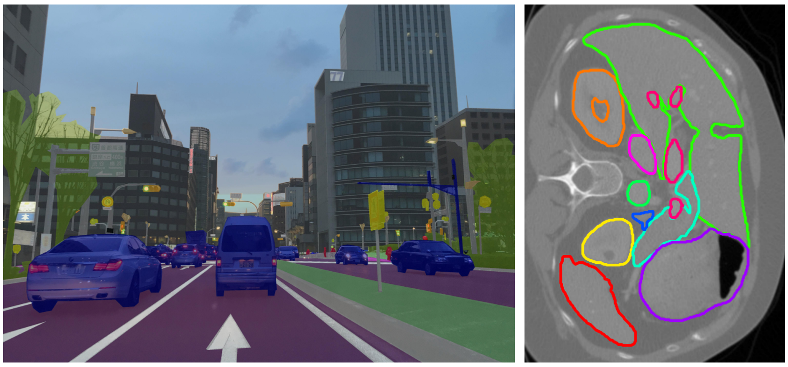

Figure 2.1: *Two examples of semantic segmentations. To the left is an image from the Mapillary Vistas dataset [2], a street-level image dataset with 66 semantic classes. The semantic class of each pixel is visualized by overlaying the original pixel with the class color. To the right is a slice of a CT image from the MICCAI 2015 challenge "Multi-Atlas Labeling Beyond the Cranial Vault" [3] for organ segmentation in the abdomen. Here the voxels of a class are visualized by delineating them with the class color. Note that this is only one slice of the original 3D CT volume.*

2.1.1 Evaluation

Given an image paired with a semantic segmentation it is quite easy for a human to visually evaluate the segmentation as good or bad. It is however important to quantify how good a segmentation is, both to be able to quickly evaluate a method applied to a big set of images and also to be able to compare between different methods. A straightforward metric to use is the pixel accuracy which is defined as the ratio between correctly classified pixels and total number of pixels. However, for some datasets, the per-pixel accuracy can be quite misleading. Given, for example, an image with a lot of pixels labeled as "background". A segmentation method simply assigning the "background" label to all pixels will get a high pixel accuracy even though it obviously performs poorly.

An alternative metric is the commonly used Intersection over Union (IoU) or "Jaccard" index. Given the set of pixels A segmented as a class and the set of pixels B belonging to the same class according to the annotation the IoU is

$$\mathrm{IoU} = \frac{|A \cap B|}{|A \cup B|}. \tag{2.1}$$

In terms of true/false positives/negatives we get

$$\mathrm{IoU} = \frac{\#tp}{\#tp + \#fn + \#fp}, \tag{2.2}$$

where $\#tp$ denotes number of true positives, $\#fn$ denotes number of false negatives and so on. The IoU can value between zero and one where a value of one means

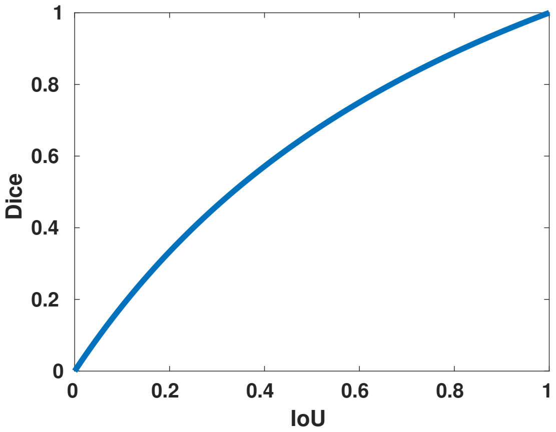

Figure 2.2: *The Dice coefficient plotted as a function of the intersection over union. The Dice coefficient and intersection over union are two commonly used measure to quantify segmentation results.*

a perfect overlap of the segmentation and the ground truth while a value of zero means no overlap at all. For multi-label problems the mean IoU (mIoU) over all classes is usually measured as an overall performance indicator of a segmentation method. A segmentation method simply assigning the "background" label to all pixels will get a quite low mIoU.

For medical image segmentation tasks the Sørensen-Dice coefficient, or simply Dice coefficient, is a common metric. Using the previous notation the Dice coefficient is defined as

$$\text{Dice} = \frac{2|A \cap B|}{|A| + |B|}, \tag{2.3}$$

which in terms of true/false positives/negatives can be written as

$$\text{Dice} = \frac{2\#tp}{2\#tp + \#fn + \#fp}. \tag{2.4}$$

Similar to the IoU the Dice score can vary between zero and one. The relation between the Dice score and the IoU is Dice $=$ 2 IoU/(1+IoU) which is visualized in Figure 2.2. As can be seen from the figure the Dice coefficient always corresponds to a lower intersection over union.

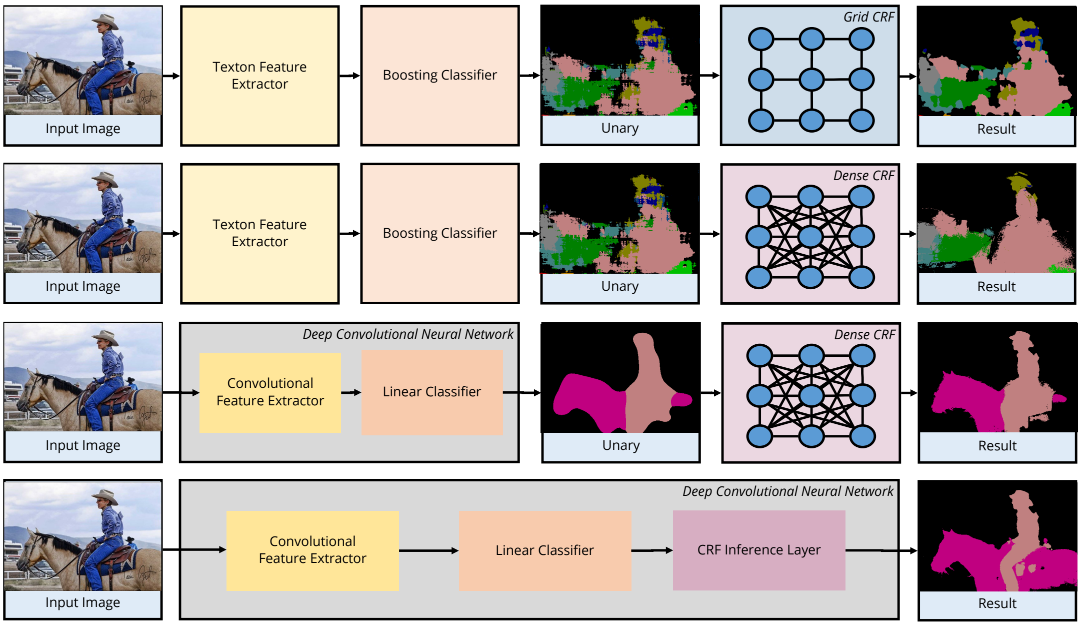

Figure 2.3: *Evolution of Semantic Segmentation systems. Initially, most approaches relied on hand-crafted image features and a fairly simple CRF model, this is represented in the first row showing the "Textonboost" work [4]. The second row uses a more sophisticated CRF model, DenseCRF, presented in [5]. Later on, most works have replaced the hand-crafted features with features learned from data with a Convolutional Neural Network. An early example of this is [6]. Currently, several state-of-the-art method follows an approach that first appeared in [7] where the CRF inference is incorporated as a part of the neural network. Allowing learning of the CNN and CRF weight simultaneously. This image is taken from [1] and result for this figure were obtained using the publicly available code of [6–10]*

2.1.2 Development of Approaches

Semantic segmentation methods date back to the 1970s, *e.g.* [11, 12]. Many of the early approaches tried to divide the image into semantic areas and then relate these areas to each other using a fixed rule-based system. It was in most cases hard to get these kind of rule-based or grammar-based methods to generalize well and performance was quite poor for general images.

From the early 2000s up until now the popularity and performance of semantic segmentation methods has increased tremendously. The early methods utilized powerful tools such as image descriptor and machine learning [13]. The majority of these methods are data driven and require manually annotated images to be able to train the models. However, the models can of course be applied to unseen images and segment them into the semantic classes that were present in the manually annotated images. A lot of the state-of-the-art methods used a CRF to be able to model interactions between the input images and output labels but also interactions between output labels. Given a CRF model most early approaches used the following pipeline

1. Extract features from the image. The features extracted could be the RGB color of the pixel and its surrounding pixel or some more advanced features such as Textons [14] or SIFT [4].
2. Use the extracted features and the annotated image to train an appearance model, *i.e.* a local classifier.
3. Use the output of the appearance model to form the unary term, *i.e.* the part of the CRF that models interactions between input and output.
4. Define, or learn from data, how the CRF should model interactions between output labels. Most commonly the type of interactions were pairwise, *i.e.* between the classes of two pixels.
5. Perform inference on the CRF model to segment an image.

This is of course a rough pipeline which a lot of methods will not fit into. In addition, a lot of extensions and variants exists for basically each step of the pipeline. Regarding, the first point of extracting features most work has moved from the carefully designed features to learning features from annotated data, usually with a CNN. This will be discussed thoroughly in Chapter 2.2. Also, several works have done data driven approaches to learn the pairwise interactions described by the CRF. In addition, several different types of CRF models have been proposed. A notable example is the DenseCrf presented in [5] where every pair of pixels is connected by a pairwise term in the CRF. Finally, during the last few years methods that learn the parameters of the CRF as well as the weights of the feature extracting CNN jointly have appeared. Two of them are the papers included in this thesis but a few more examples exist. Figure 2.3 provides a summary of this development.

2.2 Learning Features

As mentioned in Section 2.1.2 most methods for semantic segmentation nowadays use image features learnt from annotated data. The dominating approach is to use a CNN to learn these features and looking on most popular semantic segmentation benchmarks all top entries on the leaderboard use a CNN. In this section an introduction to Artificial Neural Networks (ANNs) and CNNs is given.

The idea behind ANNs and CNNs is not new. Already in the 1960s the biologically inspired Perceptron was introduced [15] which resembles the commonly used ANNs of today. Also the idea of introducing spatial invariance in ANNs were presented already in 1980, when K Fukushima *et al.* introduced the "Neocognitron" [16]. During the 1980s and 1990s there were some progression in the field of neural networks but it wasn't until a few years back that these types of methods got their breakthrough. In 2012 Krizhevsky *et al.* [17] presented "Alexnet", a CNN for classifying images of the ImageNet [18] dataset that achieved considerably better than the previous state-of-the-art. Since then, approaches using CNNs have become dominant in most detection and classification problems [19]. For the task of semantic segmentation a defining paper was J Longs *et al.* "Fully convolutional networks for semantic segmentation" [10] which introduced a method of transforming CNNs previously used for classification to efficiently segment an image. These types of "Fully Convolutional" networks are the standard for semantic segmentation nowadays.

2.2.1 Multilayer Neural Networks

The most common ANNs have a feed-forward neural network architecture. In these networks computations are done layer-wise, and the values of the data at one layer of the network depend only on computations in previous layers. In one layers of the network, the input to the layers is multiplied by a weight vector $\boldsymbol{W_i}$ and a bias vector is added $\boldsymbol{b_i}$ according to

$$\boldsymbol{g}_i = \boldsymbol{W}_i \boldsymbol{h}_{i-1} + \boldsymbol{b}_i, \tag{2.5}$$

here $\boldsymbol{h}_{i-1}$ is the output of the previous layer (or the input data if i is the first layer). The vectors $\boldsymbol{h}_i$ are commonly referred to as hidden units (except for the inputs $\boldsymbol{h}_0$ and the output $\boldsymbol{h}_L$) and their size depends on the size of the weight matrices $\boldsymbol{W}$. A weight matrix with less rows than columns will decrease the size of the hidden units vector. After this computation the output $\boldsymbol{g}_i$ is passed through a non-linear activation function

$$\boldsymbol{h}_i = \sigma(\boldsymbol{g}_i). \tag{2.6}$$

These computation layers are stacked on top of each other and the output of the last layer L is the output of the neural network $\boldsymbol{y} = \boldsymbol{h}_L$. In modern literature, these types of neural network layers are often referred to as fully connected layers.

2.2.2 Activation Functions

The activation functions are a crucial part of the neural network. If these were to be omitted the computations of the entire network would consist of only linear functions and could be replaced with an equivalent single matrix multiplication. In contrast, with non-linear activation function, it has been shown that a feed-forward neural network is a universal function approximator [20]. This means that, in theory, they can learn any function.

In the early days of neural networks, smooth non-linear activation functions were commonly used. Two examples of these are the Sigmoid, defined as

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.7}$$

and the hyperbolicus tangen function $\sigma(x) = \tanh x$. These functions are quite similar but differs in range, the output of a sigmoid lie within $[0,1]$ while the output of $\tanh(x)$ lie within $[-1,1]$. The most popular activation function at the moment is the rectified linear unit (ReLU) [21] which is defined as $\sigma(x) = \max(0,x)$. Using ReLU activation functions in a neural network makes the training faster in general as well as allowing training of networks with more layers [19].

The activation function of the final layer is usually chosen differently to the intermediate activation functions. The choice usually depends on what task we are training the network to solve. For example, if we want to use the network to solve a regression problem we might not use any final activation function at all, allowing for unbounded output values of the network. If we instead are interested in image classification we could use a softmax activation function defined as

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{N} e^{x_k}} \quad \text{for} \quad j = 1, ..., N. \tag{2.8}$$

The softmax function outputs a set of values all between zero and one and which sum to one. The value of $\sigma(x)_j$ can hence be used as an estimation of the probability of the current input belonging to class j.

2.2.3 Convolutional Neural Networks

Convolutional Neural Networks [22] are designed to process data that has an inherent grid-like structure, such as 2D RGB images, 3D videos or medical images such as CT scans. Since many of these data types have a lot of input parameters, an RGB image of standard size can for example be represented with $512 \times 512 \times 3 = 786432$ values, the weight matrix $\boldsymbol{W}$ of a fully connected layer would become very large. This would make the computations very demanding while also giving the neural network an extremely large amount of weights, something that might cause overfitting.

CNNs circumvent this problem by using a biologically inspired spatial weight sharing scheme [23]. Instead of learning full weight matrices for each layer a CNN learns a bank of filters for each convolutional layer. The intermediate values between layers are referred as feature maps and keep their spatial grid-like structure throughout the network. Learning filter, instead of full weight matrices, means that the same weight values will be applied at every spatial position for each layer, greatly reducing the number of weights needed to be learnt. This would be the equivalent of restricting the weight matrix of equation (2.5) to be a Toeplitz matrix. In addition, the convolutional layers are spatially invariant, meaning that input patterns found in different parts of an image will be processed similarly regardless of spatial position.

Another common component of a CNN are pooling layers. A pooling layer applies a rectangular window to each feature map forwarding for example the maximum number present in the window (for max-pooling layers). Pooling layers introduces an invariance to small shifts in input data while also reducing the spatial size of the feature maps, controlling the capacity of the neural network [19]. Adding pooling layers also enlarges the receptive field of higher level features. The receptive field of a feature is the part of the input image that might influence the value of the feature, a larger receptive field enables learning of more complex features. A common approach to building an CNN for image classification is to stack a couple of convolutional and pooling layers, adding activation functions (typically ReLU) after the convolutional layers. This enables the CNN to learn more complex and high-level features for each stacked layer. Ideally the first few layers learn to extract low level image features such as edges, lines and blobs while later layers extracts complex features such as faces, legs or wheels. Finally one or several fully connected layer can be applied to transform the features from spatially structured maps to for example a vector of estimated class probabilities.

Convolutional Layers

As mentioned the convolutional layers of a CNN learn a bank of filters. Given an input feature map $\boldsymbol{X}$ of size $W^{in} \times H^{in} \times F^{in}$, where W is the width, H the height and F the number of feature maps. A trained bank of filters is applied according to

$$\boldsymbol{Y}_j = \boldsymbol{B}_j + \sum_i \boldsymbol{W}_{ij} * \boldsymbol{X}_i, \tag{2.9}$$

where $\boldsymbol{X}_i$ denotes the i-th feature map of $\boldsymbol{X}$. The output feature map $\boldsymbol{Y}$ has a size of $W^{out} \times H^{out} \times F^{out}$, padding can be used to keep the same width and height as the input feature map. The filter bank has a size of $K_1 \times K_2 \times F^{in} \times F^{out}$, where $K_1 \times K_2$ is the size of each filter. For each output, optionally, a bias weight is learnt. $\boldsymbol{B}_j$ denotes this bias resized to the width and height of the output feature map. The size of the filters differs from application to application but a size of 3×3 is most commonly used [24–26].

A variant of the convolutional layer designed to provide a greater increase of the receptive field, *i.e.* the region of the input that affects a particular unit of the network, between subsequent layers is the *à-trous* or dilated convolutions [27]. These convolutions uses a set of upsampled filters where only weights at every l-th index is non-zero. Here, l is usually referred to as the dilation factor, note that dilated convolution with $l = 1$ is just standard convolution.

Pooling Layers

Pooling layers perform down-sampling of the image features [28]. Several types of pooling layers exist, the most common ones are max pooling and average pooling. These layers applies a fixed size window to the input feature map in strides. It then outputs the max (or average) of the values in this window. Choosing a stride equal to the window size results in non-overlapping regions that forwards information to the next feature map. Choosing a window size of 2×2 and a stride of 2 would result in a down-sampling of the spatial size of the feature map by a factor of 2.

2.2.4 Learning

Once the architecture of our CNN is set we can view it as a function approximator $f(\boldsymbol{x}, \boldsymbol{\theta})$, where $\boldsymbol{x}$ is the input and $\boldsymbol{\theta}$ the learnable weights of each layer. This section will give a brief introduction to the most important parts needed for the learning process. Note that this is only applicable for supervised learning, where an annotated dataset is available.

Loss Function

A loss function is a way to quantify how well the CNN is performing, it hence measure the compatibility of the CNNs output, or prediction, to the ground truth label. The loss is generally defined for one sample of the dataset, and during learning the weights of the CNN, $\boldsymbol{\theta}$ are adjusted to minimize the mean of the losses

$$L(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \frac{1}{N} \sum_i L_i(y_i, f(\boldsymbol{x}_i, \boldsymbol{\theta})). \tag{2.10}$$

Here $\boldsymbol{X}, \boldsymbol{Y}$ are the set of input and labels of a given dataset with N samples and $\boldsymbol{x}_i, y_i$ denotes the data/label pair of one sample.

A commonly used loss function for classification tasks is the cross-entropy loss. For a CNN with a softmax activation function as a last layer, meaning that it outputs an estimation of the probabilities of input $\boldsymbol{x}_i$ belonging to each class, the cross-entropy loss may be defined as

$$L_i(y_i, f(\boldsymbol{x}_i, \boldsymbol{\theta})) = -\log(f_{y_i}(\boldsymbol{x}_i, \boldsymbol{\theta})). \tag{2.11}$$

Here, $f_{y_i}(\boldsymbol{x}_i, \boldsymbol{\theta})$ is the estimate of the probability from the CNN that the input belongs to the ground truth class y_i. The name cross-entropy loss comes from the fact that this loss minimizes the cross entropy between the distribution of the ground truth labels and the label distribution generated by the CNN, given that the samples are independent and identically distibuted random variables.

Loss Minimization

As previously mentioned the learning is achieved by minimizing a defined loss function over the given dataset. Since there generally is not any closed-form solution to the learning problem, $\theta^* = \arg\max(L(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}))$, local optimization methods are often used. Most commonly a variant of gradient descent is used which updates the parameters of the CNN according to

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \nabla_{\boldsymbol{\theta}} L(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}), \tag{2.12}$$

where η is the step size or learning rate. For large datasets this is however inefficient and often a stochastic approximation of the gradient is used instead. This is called mini-batch gradient descent and is defined as

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}} L_i(y_i, f(\boldsymbol{x}_i, \boldsymbol{\theta})), \tag{2.13}$$

where $\mathcal{B}$ is the batch which in turn is a subset of the complete dataset. Using a mini-batch of size one is referred to as stochastic gradient descent. There are several variants of this update rule, designed to reduce noise of the estimated gradient and accelerate convergence. Some examples are gradient descent with momentum [29], with Nesterov momentum [30], AdaGrad [31] and Adam [32]. All of these are first-order methods which means that they only require the calculation of the gradient with regards to the weights.

The gradient of the loss function with respect to all the weight of the network can be efficiently computed using the back-propagation algorithm [29]. The back-propagation algorithm is just a practical application of the chain rule for derivatives. Given the loss derivative $\frac{\partial L}{\partial y}$ with respect to the output of a simple layer described by $y = f(x, \theta)$, where x is the input, y the output and θ the weights. The loss derivative with respect to the input can calculated by simply applying the chain rule $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial f}{\partial x}$, similarly for the weight we get $\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial y}\frac{\partial f}{\partial \theta}$. Using this back-propagation we can start from the final layer of the network and calculate the loss with respect to the input of each layer, propagating the loss gradient all through the network until we have calculated it with respect to every weight of the network.

Since the learning problem is non-convex and almost all methods are based on local optimization it is a possibility for the learning to get stuck in a poor local minimum. In practice, this is generally not a big problem, even for different initial conditions many networks reach a solution of very similar quality [19]. Recent work points towards the existence of a lot of saddle points in the loss surface that the learning algorithm might get stuck in [33, 34]. However, all of them have very similar and low values of the loss function and hence give a good enough solution.

Regularization

Overfitting is a term used for a model that performs well on the training data but poorly on unseen input data. Since a typical CNN contains a very large number of free parameters it is prone to overfitting. A large enough CNN could basically learn to "memorize" the training data instead of learning good rules that generalize to unseen data. Because of this, several regularization methods meant to prevent the CNN to overfit have been developed.

Ideally, we would like to just add training data until the CNN is incapable of overfitting. Annotating new data is however a timely process which we generally want to avoid. An alternative to adding new training data is to perform data augmentation on the already available data. This means changing the samples of the data in an slightly randomized way during training. Common augmentation operations used for image tasks are, random cropping of the image, random rotation or simply adding noise to the pixel values.

Another fairly simple regularization technique is weight decay. This means adding an extra term to the loss function that penalize large values of the parameters of the CNN. For the case of L^2 weight decay the term $\lambda\frac{1}{2}\sum_i \theta_i^2$, where λ is the weight decay strength, is added to the loss function in (2.10). For L^1 regularization we instead add the term $\lambda\sum_i |\theta_i|$, favoring sparse solutions.

Two additional, very popular, regularization techniques are Dropout [35] and Batch Normalization [36]. These are added as separate layers to the CNN and have different functionalities during learning and during inference. Dropout works by only keeping the values of each neuron non-zero with set probability p, the others are set to zero. During inference, all neuron values are kept but scaled with a factor p. Batch Normalization shift the values of the features to have zero mean and unit variance for each mini-batch during training. In addition to avoid overfitting to some extent this also allows the use of a higher learning rate during training [36].

CNNs for Semantic Segmentation

The task of annotating data is considerably harder and more time-consuming for semantic segmentation where each pixel need to be annotated compared to image classification where only one label per image is needed. This restricts the size of available datasets for semantic segmentation and there are no available datasets of the same size as for example Imagenet [37]. Due to this, many popular CNNs for semantic segmentation have a classification counterpart that has been trained on the million images of Imagenet. The architecture of the classification CNN is changed to enable dense output maps, transforming it to a segmentation CNN. However, the weights of the first few layers are kept with the motivation that these layers have learnt to extract meaningful image features during the extensive classification training. This approach have shown to be preferable to training from scratch for many segmentation tasks, even for images fairly different from the Imagenet data [38–40].

Repurposing a classification network for semantic segmentation is not entirely straight-forward. As previously mentioned a defining paper for ths was "Fully convolutional networks for semantic segmentation" by J Long *et al.* [10] where they presented segmentation version of several classification network that were state-of-the-art on Imagenet at that time. The fully connected layers of these networks were transformed to convolutional layers with filter size 1×1, which changes the previous classification scores to spatial feature maps. These feature maps, together with feature maps earlier in the CNN were upsampled using deconvolution layers [41] and merged providing dense output predictions for images of arbitrary size. These CNNs can then be trained for segmentation end-to-end using a pixel-wise version of the cross-entropy loss presented in Section 2.2.4.

Despite the success of fully convolutional networks of [10] this architecture has several drawbacks. Pooling layers are great for image classification, enabling the CNN to learn complex high-level image feature. It is however not ideal since performing pooling operations means loosing spatial information of where the image features came from in the image. Some works have tried to get rid of the pooling layers entirely [42] and other types of layers has been introduced in an effort to keep spatial information while still achieving large receptive field. An example of this is the dilated convolutions mentioned in Section 2.2.3 [27].

Many recent works considering CNNs for semantic segmentation try to design networks that are able to learn high-level image features while not losing spatial information. Some examples include encoder-decoder networks such as Segnet [43] and U-Net [44] as well as PSP-Net [45] that processes features on different resolution in separate paths.

2.3 Learning Structure

As mentioned in Section 2.2.3 CNNs are good at modeling complex relations between input data and output data. They cannot however explicitly take dependencies between output variables into account. In addition, they are often trained with a pixel-wise loss function, disregarding the fact that the output data is actually structured.

A way of taking output structure into account which also allows for explicit modeling of dependencies between output variables is using Probabilistic Graphical Models (PGMs). A PGM models a probability distribution over a set of random variables whose structure is defined via a graph. In this thesis we will focus on Conditional Random Fields (CRFs) that are commonly used for semantic segmentation.

2.3.1 Conditional Random Fields

Conditonal Random Fields (CRFs) models the conditional probability, $P(\mathcal{Y}|\boldsymbol{X})$ of an output set $\mathcal{Y} = \{Y_1, ..., Y_N\}$ given and input $\boldsymbol{X}$. Working with images, $\boldsymbol{X}$ denotes the image values and we generally associate one input and one output variable with each pixel. For semantic segmentation each output Y_i is assigned a value from a finite set of possible states $\mathcal{L} = \{l_1, l_2, ..., l_L\}$, where each state represent a class label. The dependencies between output variables are described by an undirected graph whose vertices are the random variables $\{Y_1, ..., Y_N\}$. The conditional probability for the CRF can be written as

$$P(\mathcal{Y} = \boldsymbol{y}|\boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} \exp(-E(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{w})), \tag{2.14}$$

where $E(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{w})$ denotes the Gibbs energy function with respect to the assignment of labels to the output variables $\boldsymbol{y} \in \mathcal{L}^N$. The parameters, $\boldsymbol{w}$, of the CRF can either be hand-crafted from prior knowledge or learnt from data. $Z(\boldsymbol{X})$ is the partition function given by

$$Z(\boldsymbol{X}) = \sum_{\boldsymbol{y} \in \mathcal{L}^N} \exp(-E(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{w})). \tag{2.15}$$

It is hence a normalization constant making the conditional probabilities sum to one. Note that the sum is over all possible combination of label assignments available, it is hence computationally costly to evaluate the value of the partition function.

For most image applications the Gibbs energy function decomposes over unary and pairwise terms, *i.e.* terms depending on only one and two variable respectively.

The energy can be written as

$$E(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{w}) = \sum_{u \in \mathcal{V}} \psi_u(y_u, \boldsymbol{X}; \boldsymbol{w}) + \sum_{(u,v) \in \mathcal{E}} \psi_{uv}(y_u, y_v, \boldsymbol{X}; \boldsymbol{w}), \tag{2.16}$$

note that the graph structure defines what terms are present in this energy. For $\psi_{uv}(y_u, y_v, \boldsymbol{X}; \boldsymbol{w})$ to be non-zero node u and v must share an edge. The terms $\psi_u(y_u)$ and $\psi_{uv}(y_u, y_v, \boldsymbol{X}; \boldsymbol{w})$ are commonly referred as unary and pairwise potentials respectively.

Potential Types

The unary potentials, also known as the data cost, of the CRF energy are often obtained from a pixelwise classifier estimating the class probabilities of each pixel. Commonly the term for each pixel is set to

$$\psi_u(y_u = l_p) = -w_1 \log(P(y_u = l_p | \boldsymbol{X})), \tag{2.17}$$

where $P(y_u = l_p|\boldsymbol{X})$ is an estimate of the probability of pixel u belonging to class l_p.

For the pairwise potential,as a first step, the connectivity or structure of our graph needs to be defined. This specifies what pairwise terms should be included in the energy, and also which output variables should depend on each other. A simple and commonly used structure is the nearest neighbour connectivity where pixels are connected through an edge to its neighbours only. The size of the neighbourhood might vary but for 2D images a size of four or eight is common. An example of this structure can be seen in Figure 2.4.

The pairwise potentials, $\psi_{uv}(y_u = l_p, y_v = l_q, I; \boldsymbol{w})$, defines the cost of assigning label l_p to pixel u and label l_q to pixel v . It can hence be used to enforce consistency and structure in the output. As an example, for semantic segmentation, we generally want neighbouring pixels to have the same labels. A type of pairwise term that enforces this is the Potts model given by

$$\psi_{uv}(y_u = l_p, y_v = l_q, I; \boldsymbol{w}) = w_2 \mathbb{1}_{l_p \neq l_q}, \tag{2.18}$$

where $\mathbb{1}_{l_p \neq l_q}$ denotes the indicator function equaling one if $l_p \neq l_q$ and zero otherwise. This pairwise term can be generalized in several ways, for example we might want to weight the cost of assigning different labels to neighbouring pixel differently depending on if they have similar color or not. This can be achieved by adding a weighting term according to

$$\psi_{uv}(y_u = l_p, y_v = l_q, I; \boldsymbol{w}) = w_2 \mathbb{1}_{l_p \neq l_q} e^{-(x_u - x_v)^2}, \tag{2.19}$$

where x_u and x_v are the pixel values of pixel u and v. This type of pairwise terms adds a lower energy if two neighbouring pixels differ a lot in color.

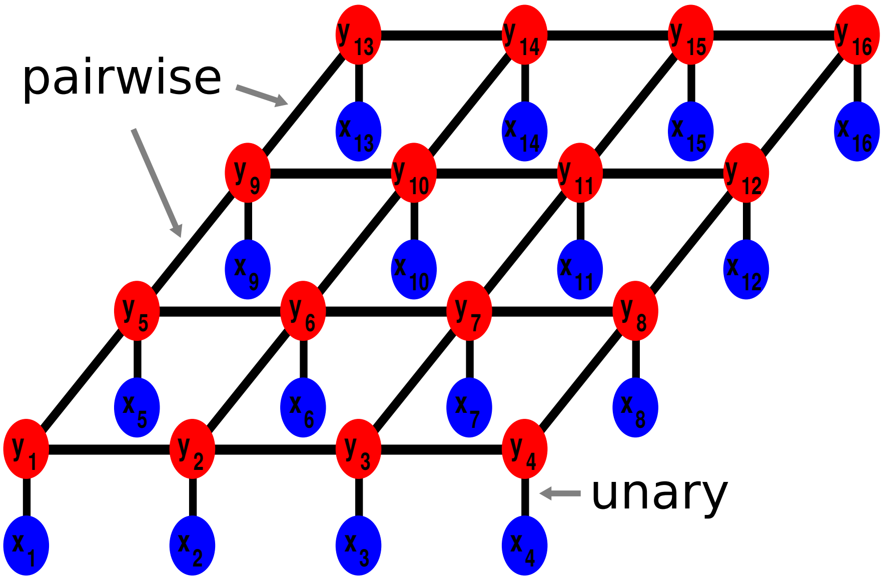

Figure 2.4: *CRF with a simple nearest neighbour connectivity, neighbourhood size four. The variables y_u are assigned class labels while the variables x_u represents the pixel values.*

Both of these pairwise terms are constructed using prior knowledge, such that neighbouring pixel often have the same label unless there is a change in contrast. This is of course not true in all cases and several works have instead tried to learn the pairwise term from data [46, 47]. In Paper I we present a CRF model with more general pairwise potentials that can be learnt from data.

Using a neighbourhood only consisting of neighbouring pixels limits the extent on how far across the image information can propagate. A natural way to increase this limit is to increase the size of the neighbourhood, for example connecting all pixels closer than d pixels apart. The extreme of this would be to connect all pairs of pixels which is done for the denseCrf model. The denseCrf model were popularized by [5], that presented a method to perform efficient inference for these types of CRFs. The pairwise terms for dense CRFs also include a weighting on the distance between two pixels, hence the strength of the pairwise term decays exponentially with the distance between the pixels.

It is also possible to include potentials that depend on more than two pixel labels, *i.e.* higher order potentials. Higher order potential can for example be used to enforce consistency within superpixels or utilize object detection results for semantic segmentation [48–51].

Inference

Given an image $\boldsymbol{X}$ the inference problem, giving a semantic segmentation, equates to finding the maximum a posteriori labeling of the model in equation 2.14. Finding the minimizer to the Gibbs energy, *i.e.*

$$\boldsymbol{y}^* = \arg\min_{\boldsymbol{y}} E(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{w}), \tag{2.20}$$

is an equivalent problem. This problem is in general NP-hard [52], typical approaches to solving it can hence be divided into two categories, exact algorithms that only applies to special cases of the energy and approximate solutions. We will provide a few examples here but for an extensive overview of approaches we refer to [53, 54].

If we deal with a binary segmentation problem, *i.e.* only are interested in two classes, and if the energy is submodular the globally optimal solution can be found using the graph cuts method [54]. This approach can be extended to multi-label problems using the α-expansion [55], however we lose the guarantee of finding the global optimum.

Several popular methods are based on a relaxation of the original problem, these are usually the most efficient ones for performing inference in denser CRFs. One example is the mean-field method where the original distribution is $P(\boldsymbol{y}|I)$ is approximated with a fully factorized one $Q(\boldsymbol{y})$. The optimization is then done by minimizing the Kullback–Leibler divergence between the two distributions. Other approaches rely on a continuous relaxation of the Gibbs energy, and then using local search methods to find a local minimum of the energy. This type of methods have been shown to outperform mean-field on several tasks [56] and is the approach used in Paper I.

Parameter Learning

The learning problem consists of estimating the parameters of the CRF, $\boldsymbol{w}$, based on a training set $(\boldsymbol{Y}^{(k)}, \boldsymbol{X}^{(k)})_{k=1}^{N}$. The goal with the training is that if inference is performed for an input image from the data set, we want a solution close to the ground truth labeling. An intuitive approach to the learning problem is based on the maximum likelihood principle. This means finding the set of parameters that maximizes the probability of the training set.

A major difficulty when performing maximum likelihood training for CRFs is that it requires computation of the partition function for each training instance and for each iteration of a numerical optimization algorithm. This is of course computationally expensive and makes learning infeasible for CRF models used for semantic segmentation. Most popular learning methods hence makes approximations that simplifies the computation of the partition function. Mean field is an example of this where the fully factorized distribution simplifies computation of the partition function [5]. Piece-wise training is also an option which only requires computation of local normalization factor over fewer variables [57,58]. Other methods instead try to estimate the partition function using sampling [59].

Another approach is to use a learning method that avoids the computation of the partition function, for example learning a model that maximizes the margin between the energy of the ground truth and any other output configuration [60,61]. This can be formulated as

$$\begin{aligned} &\max_{\boldsymbol{w}} \quad \zeta \\ s.t. \quad &E(\boldsymbol{Y}, \boldsymbol{X}^{(k)}; \boldsymbol{w}) - E(\boldsymbol{Y}^{(k)}, \boldsymbol{X}^{(k)}; \boldsymbol{w}) \geq \zeta \quad \forall\, k \text{ and } \boldsymbol{Y} \neq \boldsymbol{Y}^{(k)}. \end{aligned} \tag{2.21}$$

Since, there is an exponential amount of constraint in this optimization problem it is not feasible to solve it as is. A solution to this is to iteratively add the constraints that currently is furthest away from being satisfied [62]. This learning method is utilized in Paper II.

2.4 End-to-End Learning

Combining CNNs and CRFs is a powerful approach for dense classification tasks such as semantic segmentation. The CNNs ability to learn complex high-level image features paired with the CRFs ability to model output dependencies generally yields impressive results. However, many existing approaches use a two step training process to learn the weights of the CNN and CRF. Firstly, the CNN is trained to perform pixel-wise segmentation on the available data set. Secondly, the CRF is trained keeping the unary potentials fixed (although based on the output of the CNN). This is often referred to as piece-wise training and is non-ideal since the CNN is learnt while ignoring dependencies between output variables.

Instead, a better solution would be to perform end-to-end training. This means jointly training the CNN and the CRF at the same time. In this way the CNN and the CRF get the chance to learn how to interact and exploit complementary information to achieve as good of a result as possible. During recent years several examples of these deep structured model trained end-to-end have been proposed in the literature [7,47,63–65]. This section gives a very brief introduction to some of these methods, for more details we refer to the recently published tutorial paper on the subject [1].

2.4.1 CRF Inference as a Neural Network Layer

Given a CRF inference method only consisting of differentiable operations, these operations can be implemented as neural network layers. By implementing the back-propagation routines for this layer, which amounts to applying the chain rule for derivative, the error derivatives with respect to the parameters of the CRF can be computed during training. In addition, the error derivative with respect to the output of the CNN can be computed and the error can be propagated all the way back through the CNN. This enables the parameters of both the CRF and the CNN to be updated simultaneously using for example stochastic gradient descent. This was shown to be possible for the mean-field inference algorithm [7]. In Paper I we show that this is possible for gradient-based CRF inference as well.

2.4.2 Back-propagating CRF Learning Objective

Many of the approaches for CRF parameter learning presented in Section 2.3.1 can be abstracted to minimizing a global objective L. This global objective depends on the samples of the data set, the parameters of the CRF as well as the output of the CNN, denoted $\boldsymbol{z}$, used to create the CRF potentials. If we are able to calculate the gradient of this global objective with respect to the CNN output, $\nabla_{\boldsymbol{z}} L$, we can back-propagate this gradient back through the CNN to calculate $\nabla_{\boldsymbol{\theta}} L$, where $\boldsymbol{\theta}$ are the weights of the CNN. The weights can then be updated using local search methods. Examples of methods in this category are [47, 59, 63]. In Paper II we present a method for doing this utilizing the max margin training approach for CRFs introduced in Section 2.3.1.

Chapter 3

Summary

The focus of this thesis is to develop DSMs for semantic segmentation. Emphasis have been put on creating models that are possible to train end-to-end as well as the methods needed for training. Paper I and Paper II are obvious examples of this. In Paper III a robust segmentation method for abdominal organs using CNNs is developed. The original idea was to use the DSM and training routines developed in paper II and add it to this framework as well. However, this did not actually improve the results much and was hence discarded. This brings up an important question, what types of CRFs are needed to improve on the results of a CNN? This is something that will be discussed in Chapter 4.

Regardless of using a DSM or not Paper III presents a robust method for abdominal organ segmentation. The paper combines a robust organ localization with the use of specialized organ CNNs for segmentation. Since segmentation is a key problem in medical image analysis, a method for organ segmentation can be crucial for numerous applications in medical research and clinical care such as computer aided diagnosis and surgery assistance. In addition, robustness is something that is generally highly valuable for medical applications. At the time of writing this thesis the development of deep learning methods for 3D medical application is a popular research subject and many large scale research projects exist on this topic [66].

Paper II introduces a method of training a DSM end-to-end using a max-margin objective. However, several restrictions to the CRF is needed to be able to do the training efficiently. The CRF used has a pairwise term where each pixel is only connected to its closest neighbours. This can easily be extended, however the method uses graph-cut inference of the CRF during training which becomes slow for densely connected CRFs.

In Paper I a framework for training DSMs with more expressive CRFs is presented, here a newly developed approximate inference method of the CRF is used. However, we would like to point out that this should not be seen as a strictly better approach than the one used in Paper II. The max-margin approach of Paper II has the advantage of using a fast and exact inference algorithm of the CRF. In addition the learning approach used have been shown to generalize well, even for smaller datasets. Something that suited the experiments on smaller medical data sets presented in the paper well.

The more expressive DSM presented in Paper I is however suitable for larger datasets with more semantic classes. This enabled us to do experiments on for example the PASCAL VOC 2012 segmentation benchmark [67] consisting of several thousands of annotated images and 21 semantic classes.

At the moment segmentation is all about the evaluation numbers. The methods achieving the highest mean Intersection over Union on the major benchmarks get the most attention. This is of course positive in the manner of encouraging researchers to develop practically useful and accurate methods. The major benchmarks are also an invaluable tool for comparing different methods. However, the chase for better numbers combined with the use of very data-hungry deep learning brings on a research pipeline that usually is 10% method development and 90% parameter tuning and learning hacks. Something that also makes reproducing previous work harder.

Unfortunately, the methods presented in Paper II and Paper I fail to achieve state-of-the-art results on the major benchmarks. We have however successfully shown the usefulness of both approaches in each paper respectively. Perhaps most notable for Paper II is the performance of the method on smaller medical datasets. For Paper I the model has the ability to improve on very strong CNN baseline trained on a lot of additional data, even though the DSM were only trained end-to-end on a subset of the data.

The remainder of this chapter will contain short summaries of the papers included in this thesis. In addition, the thesis author's contribution to each paper will be stated.

3.1 Paper I

M. Larsson, A. Arnab, F. Kahl, S. Zheng, and P. Torr. ”Revisiting Deep Structured Models in Semantic Segmentation with Gradient-Based Inference”. *Submitted to SIAM Journal on Imaging Sciences.*

In this paper we move from binary label CRFs with short spatial pairwise interactions to CRFs being able to handle multiple labels and learn pairwise interactions on larger distances. We present an inference technique based on gradient descent on the Gibbs energy of the CRF. This inference method consists only of differentiable operations which enables us to unroll the CRF inference as a number of update steps of a Recurrent Neural Network (RNN). During learning, we can also back-propagate through the RNN and do end-to-end training of the entire model.

Two different types of CRF models are presented in the paper. The first one consists of a spatial pairwise term as well as a high-dimensional bilateral kernels. In contrast to many previous works we don’t restrict these two kernels to have Gaussian shape but allow for arbitrary shape of the spatial and bilateral kernels. In addition, we introduce a new type of potential function which is image-dependent like the bilateral kernel, but an order of magnitude faster to compute since only spatial convolutions are employed. The major contributions of the paper are

- A new model for a pairwise CRF potential which is image-dependent like the bilateral kernel, but does not require high-dimensional filtering. It is based on a learned 2D filter bank which makes both inference and learning an order of magnitude faster than high-dimensional filtering approaches.
- A new optimization method for CRF inference based on gradient descent that enables end-to-end training.
- We show that our inference method supports learning pairwise kernels of arbitrary shape. The learned kernels are empirically analyzed and it is demonstrated that in many cases non-Gaussian potentials are preferred.

Author contribution. I did the method development and implementation with support from F. Kahl. Experiments were run by me, A. Arnab and S. Zheng. The writing was done by me and A. Arnab while the initial idea was proposed by F. Kahl.

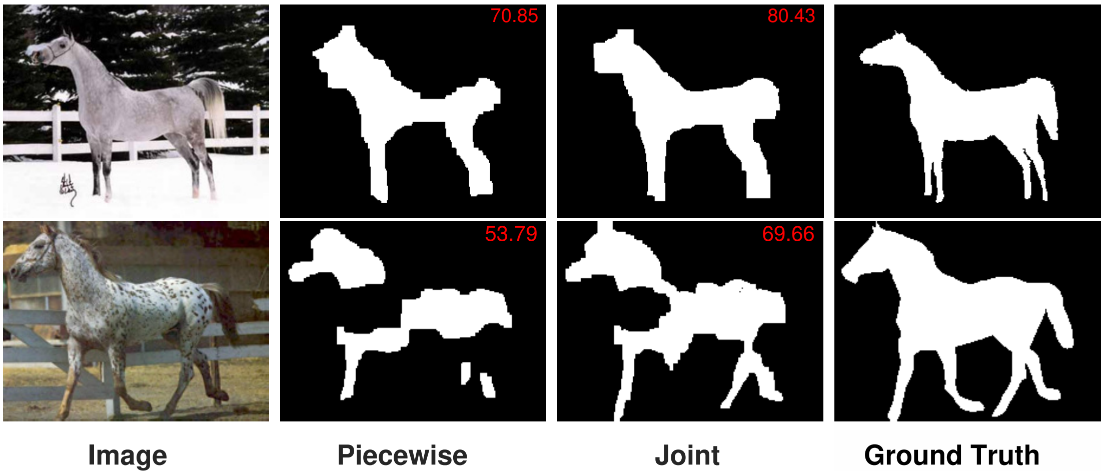

Figure 3.1: *Comparison of piecewise versus joint training of a deep structured model for some hand-picked example. The number shown in the upper right corner is the Jaccard index (%).*

3.2 Paper II

M. Larsson, J. Alvén and F. Kahl. ”Max-Margin Learning of Deep Structured Models for Semantic Segmentation”. *Scandinavian Conference on Image Analysis (SCIA)*, 28–40, 2017.

This paper presents a method for learning the parameters of a Deep Structured Model which in this case refers to a CNN paired with a CRF. The learning problem is formulated as a Structured Support Vector Machine (SSVM) and we show that it is possible to calculate the derivative of the objective with respect to the output of the CNN. This enables us to back-propagate all through the layers of the CNN and learn the weights of the CNN and the CRF at the same time. Since the SSVM uses a max-margin loss function that generally gives good generalization capabilities of the trained model this method is especially suitable for application where labelled data is limited.

Figure 3.1 shows a comparison of the piecewise and jointly trained models. As can be seen the model where the CNN and CRF have been trained jointly performs for better, avoiding error such as cutting of the legs of the horse. This is because the CNN has learnt to compensate for the slight shrinking effect of the CRF.

Author contribution. I did the method development and implementation with support from F. Kahl. The writing was done by me and J. Alvén while the initial idea was proposed by F. Kahl.

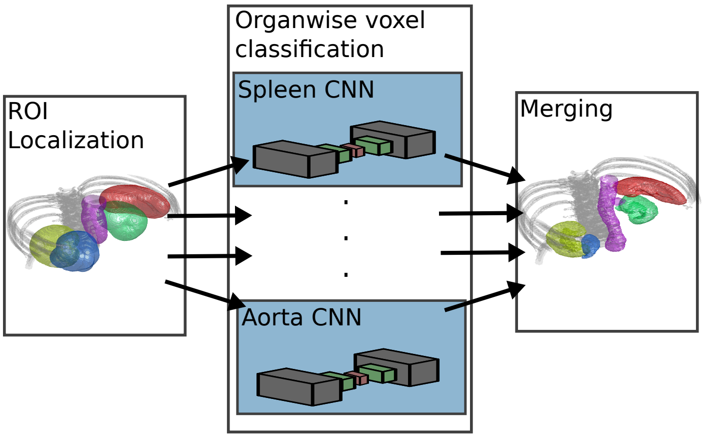

Figure 3.2: *Graphical representation of the method presented in Paper I.*

3.3 Paper III

M. Larsson, Y. Zhang and F. Kahl ”Robust Abdominal Organ Segmentation Using Regional Convolutional Neural Networks”. *Submitted to Applied Soft Computing.*

This paper presents a method for segmenting 13 different abdominal organs utilizing CNNs. The method can be divided into two main steps. Firstly, an efficient and robust feature registration method is applied estimating the center-point of each organ. Secondly, a convolutional neural network performing voxelwise classification is applied to a region, defined by a prediction mask placed at the estimated organ center-point. The prediction mask is created using the ground truths of each organ in the training set. The approach of first localizing a region of interest for each organ transforms the problem the CNN has to solve from a large multi-label problem to 13 smaller binary-labels problems. We can hence train smaller CNNs and more specialized, or regional, networks that only need to differentiate between a certain organ and the background.

During the development of this method and at the writing of the first draft of this paper there were very few examples of deep learning methods applied to medical 3D segmentation tasks and none for this specific tasks. Since then, there has been a lot of development in this area and several papers have been published further showing that deep learning methods can perform really well on these types of tasks.

Author contribution. I did the method development and implementation with support from Y. Zhang and F. Kahl. The writing was done by me while all authors contributed to the main idea.

Chapter 4

Outlook

The field of semantic segmentation has moved at a high pace during the last few years, especially when it comes to methods based on CNNs. Every month there is a new CNN with a different architecture pushing state-of-the-art further. During 2015 and 2016 the results of CNNs presented for semantic segmentation could be greatly increased by adding a CRF [7, 26]. This is mainly due to the fact that the architectures of the networks used during this time did not allow the CNN to learn interactions over long ranges. In addition the downsampling of the pooling layers resulted in a loss of spatial information that prohibited the accurate segmentation of fine edges between classes, hence the output usually became "blobby". Both of these errors are something that the most commonly used CRF models excel at. However, during late 2016 and 2017 several new CNNs have been proposed, raising state-of-the-art, that do not use a CRF for post-processing [45, 68]. This indicates that the type of CRFs commonly used in semantic segmentation might not be necessary for these large scale problems with a lot of annotated data.

An additional detail with the CRFs is that inference for most of the commonly used models are still fairly slow, especially for the powerful dense models with edge-aware pairwise potentials. Some work has been done on creating alternatives to these CRFs that are less computationally demanding but still has the ability to refine segmentations near edges [69, 70]. This is also adressed as part of Paper I.

So, for CRFs and DSMs to be really useful for these large scale segmentation problem in the future there are two improvements needed. Firstly, the inference needs to be faster, adding a CRF should not speed down the inference or training considerably. Unless, of course the gain in performance is worth it. Secondly, there might be a need to rethink the type of models we are using and try to create CRFs that are better suited to correct the errors that the new state-of-the-art CNNs do.

Moving away from the large scale segmentation benchmarks there are still a lot of applications where adding a CRF gives a big increase in performance. Looking at datasets with slightly smaller training set, DSMs tend to perform better in general. CRFs are also a good way to include prior knowledge in you segmentation pipeline.

Previous work has shown that geometric constraints, such as convex or star shaped for-ground objects only can be enforced by a CRF [71, 72].

4.1 Future Work

4.1.1 Output Structure

At the moment many of the top entries of the major segmentation benchmarks train CNNs with a pixel-wise loss function, disregarding the fact that the output is structured. Modern CNNs have the capability to, and probably do, implicitly learn that there is structure in the output. However, actually taking the output structure into account, whether by using a CRF or in some other way, could be beneficial. There is hence interest in continuing the work on end-to-end training of DSMs, both trying to improve computation speed and to design more expressive models. In addition it would be interesting to do more work on DSMs for medical image segmentation where more application-specific types of CRFs might be needed.

Another interesting approach to taking output structure into account was introduced in [73], which used a Generative Adversarial Network (GAN) for semantic segmentation. The idea was that the discriminator would be able to learn how an ground truth segmentation should look like. Hence during training, the generator, that also performs the actual segmentation, would have to output realistic segmentation to trick the discriminator. In this way the output of the segmentation CNN would have to follow, and hence learn, the output structure of the segmentations. This introduces an interesting opportunity to learn output structure without having to perform CRF inference.

4.1.2 Weak Supervision

Annotating data for semantic segmentation is a tedious task, especially for 3D medical images. There has hence been an increase in approaches that utilize weakly annotated data, for example image tags or bounding boxes of objects [74–77]. Having powerful weak supervision learning methods would mean less needed annotated data for application specific segmentation tasks, which would be crucial for many situations. An interesting idea of utilizing GANs for this task was presented last year [78], something that might be interesting to build upon.

One of our current projects is addressing the task of semantic localization, *i.e.* utilizing semantic cues to estimate the pose of a camera given the image taken and an 3D map. One step of the pipeline requires accurate semantic segmentations for road-scenarios. What we noticed, when trying some state-of-the-art models trained on the cityscapes dataset [79], was that these performed very poorly on our images. This despite the fact that the environments were fairly similar. We are now working on improving segmentation results on these images by utilizing a dataset consisting of a large 3D map. Using this we can extract correspondences where we have two images of the same place, we can hence train the CNN to be consistent for the part of the images that show the same object or objects. The hope is that this would give us a CNN that generalizes better to new conditions, without having to manually annotate new images.

Bibliography

[1] A. Arnab, S. Zheng, S. Jayasumana, B. Romera-Paredes, M. Larsson, A. Kirillov, B. Savchynskyy, C. Rother, F. Kahl, and P. H. S. Torr, "Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 37–52, Jan 2018.

[2] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *International Conference on Computer Vision (ICCV)*, 2017. [Online]. Available: https://www.mapillary.com/dataset/vistas

[3] Z. Xu, "Multi-atlas labeling beyond the cranial vault - workshop and challenge," 2016, [Online; accessed 10-January-2017]. [Online]. Available: https://www.synapse.org/#!Synapse:syn3193805/wiki/217752

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157 vol.2.

[5] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with gaussian edge potentials," in *Adv. Neural. Inf. Process. Syst.*, 2011.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *Int. Conf. on Learning Representations*, 2015.

[7] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional random fields as recurrent neural networks," in *Int. Conf. on Computer Vision*, 2015.

[8] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr, "Associative hierarchical crfs for object class image segmentation," in *International Conference on Computer Vision*, 2009.

[9] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Neural Information Processing Systems*, 2011.

[10] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[11] A. Hanson, “Visions: A computer system for interpreting scenes,” *Computer vision systems*, 1978.

[12] Y.-i. Ohta, T. Kanade, and T. Sakai, “An analysis system for scenes containing objects with substructures,” in *Proceedings of the Fourth International Joint Conference on Pattern Recognitions*, 1978, pp. 752–754.

[13] H. Zhu, F. Meng, J. Cai, and S. Lu, “Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation,” *Journal of Visual Communication and Image Representation*, vol. 34, pp. 12–27, 2016.

[14] J. Shotton, J. Winn, C. Rother, and A. 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, pp. 2–23, Jan 2009. [Online]. Available: https://doi.org/10.1007/s11263-007-0109-1

[15] F. Rosenblatt, “Principles of neurodynamics. perceptrons and the theory of brain mechanisms,” Cornell Aeronautical Lab, Tech. Rep., 1961.

[16] K. Fukushima and S. Miyake, “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition,” in *Competition and cooperation in neural nets.* Springer, 1982, pp. 267–285.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Adv. Neural. Inf. Process. Syst.*, 2012.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[19] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[20] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[21] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, “Subject independent facial expression recognition with robust face detection using a convolutional neural network,” *Neural Networks*, vol. 16, no. 5, pp. 555–559, 2003.

[24] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[26] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.” *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[27] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in *International Conference on Learning Representations*, 2016.

[28] M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.

[29] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[30] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th international conference on machine learning (ICML-13)*, 2013, pp. 1139–1147.

[31] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[32] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[33] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization,” in *Advances in neural information processing systems*, 2014, pp. 2933–2941.

[34] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” in *Artificial Intelligence and Statistics*, 2015, pp. 192–204.

[35] N. Sivastava, G., A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[36] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Conf. on Computer Vision and Pattern Recognition*, 2009.

[38] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing, “Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks,” *Computer Vision–ECCV 2008*, pp. 69–82, 2008.

[39] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[40] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[41] M. D. Zeiler, G. W. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2018–2025.

[42] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” *arXiv preprint arXiv:1412.6806*, 2014.

[43] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[44] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9351. Springer, 2015, pp. 234–241, (available on arXiv:1505.04597 [cs.CV]). [Online]. Available: http://lmb.informatik.uni-freiburg.de//Publications/2015/RFB15a

[45] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.

[46] S. Chandra, N. Usunier, and I. Kokkinos, “Dense and low-rank gaussian crfs using deep embeddings,” in *ICCV 2017-International Conference on Computer Vision*, 2017.

[47] G. Lin, C. Shen, A. van den Hengel, and I. Reid, “Efficient piecewise training of deep structured models for semantic segmentation,” in *Conf. on Computer Vision and Pattern Recognition*, 2016.

[48] V. Vineet, J. Warrell, and P. H. S. Torr, “Filter-based mean-field inference for random fields with higher-order terms and product label-spaces,” *International Journal of Computer Vision*, vol. 110, no. 3, pp. 290–307, Dec 2014. [Online]. Available: https://doi.org/10.1007/s11263-014-0708-6

[49] C. Wojek and B. Schiele, “A dynamic conditional random field model for joint labeling of object and scene classes,” *Computer Vision–ECCV 2008*, pp. 733–747, 2008.

[50] P. Kohli, P. H. Torr *et al.*, “Robust higher order potentials for enforcing label consistency,” *International Journal of Computer Vision*, vol. 82, no. 3, pp. 302–324, 2009.

[51] A. Arnab, S. Jayasumana, S. Zheng, and P. H. Torr, “Higher order conditional random fields in deep neural networks,” in *European Conference on Computer Vision*. Springer, 2016, pp. 524–540.

[52] M. Li, A. Shekhovtsov, and D. Huber, “Complexity of discrete energy minimization problems,” in *European Conference on Computer Vision*. Springer, 2016, pp. 834–852.

[53] J. Kappes, B. Andres, F. Hamprecht, C. Schnorr, S. Nowozin, D. Batra, S. Kim, B. Kausler, J. Lellmann, N. Komodakis *et al.*, “A comparative study

of modern inference techniques for discrete energy minimization problems," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1328–1335.

[54] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 147–159, 2004.

[55] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[56] A. Desmaison, R. Bunel, P. Kohli, P. H. Torr, and M. P. Kumar, "Efficient continuous relaxations for dense crf," in *European Conference on Computer Vision.* Springer, 2016, pp. 818–833.

[57] C. Sutton and A. McCallum, "Piecewise training for undirected models," *arXiv preprint arXiv:1207.1409*, 2012.

[58] A. Kolesnikov, M. Guillaumin, V. Ferrari, and C. H. Lampert, "Closed-form training of conditional random fields for large scale image segmentation," *arXiv preprint arXiv:1403.7057*, 2014.

[59] A. Kirillov, D. Schlesinger, S. Zheng, B. Savchynskyy, P. H. Torr, and C. Rother, "Joint training of generic cnn-crf models with stochastic optimization," in *Asian Conference on Computer Vision.* Springer, 2016, pp. 221–236.

[60] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin, "Learning structured prediction models: A large margin approach," in *Proceedings of the 22nd international conference on Machine learning.* ACM, 2005, pp. 896–903.

[61] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of machine learning research*, vol. 6, no. Sep, pp. 1453–1484, 2005.

[62] M. Szummer, P. Kohli, and D. Hoiem, "Learning crfs using graph cuts," *Computer Vision–ECCV 2008*, pp. 582–595, 2008.

[63] L.-C. Chen, A. Schwing, A. Yuille, and R. Urtasun, "Learning deep structured models," in *Int. Conf. on Machine Learning*, 2015.

[64] M. Larsson, A. Arnab, F. Kahl, S. Zheng, and P. H. S. Torr, "A projected gradient descent method for crf inference allowing end-to-end training of arbitrary

pairwise potentials," in *11th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, (EMMCVPR)*. Springer, 2017.

[65] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Adv. Neural. Inf. Process. Syst.*, 2014.

[66] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual Review of Biomedical Engineering*, vol. 19, no. 1, pp. 221–248, 2017, pMID: 28301734. [Online]. Available: https://doi.org/10.1146/annurev-bioeng-071516-044442

[67] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan 2015. [Online]. Available: https://doi.org/10.1007/s11263-014-0733-5

[68] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[69] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4545–4554.

[70] G. Bertasius, L. Torresani, S. X. Yu, and J. Shi, "Convolutional random walk networks for semantic image segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[71] O. Veksler, "Star shape prior for graph-cut image segmentation," *Computer Vision–ECCV 2008*, pp. 454–467, 2008.

[72] L. Gorelick, O. Veksler, Y. Boykov, and C. Nieuwenhuis, "Convexity shape prior for segmentation," in *European Conference on Computer Vision*. Springer, 2014, pp. 675–690.

[73] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," in *NIPS Workshop on Adversarial Training*, 2016.

[74] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1796–1804.

[75] F. Saleh, M. S. A. Akbarian, M. Salzmann, L. Petersson, S. Gould, and J. M. Alvarez, “Built-in foreground/background prior for weakly-supervised semantic segmentation,” in *European Conference on Computer Vision.* Springer, 2016, pp. 413–432.

[76] Y. Wei, X. Liang, Y. Chen, Z. Jie, Y. Xiao, Y. Zhao, and S. Yan, “Learning to segment with image-level annotations,” *Pattern Recognition*, vol. 59, pp. 234–244, 2016.

[77] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille, “Weakly-and semi-supervised learning of a dcnn for semantic image segmentation,” *arXiv preprint arXiv:1502.02734*, 2015.

[78] N. Souly, C. Spampinato, and M. Shah, “Semi supervised semantic segmentation using generative adversarial network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5688–5696.

[79] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Part II

Included Publications

Paper I

Revisiting Deep Structured Models in Semantic Segmentation with Gradient-Based Inference

M. Larsson, A. Arnab, F. Kahl, S. Zheng, and P. Torr

Submitted to SIAM Journal on Imaging Sciences

Comment: The layout of this paper has been reformatted in order to comply with the rest of the thesis.

Revisiting Deep Structured Models in Semantic Segmentation with Gradient-Based Inference

M. Larsson, A. Arnab, F. Kahl, S. Zheng, and P. Torr

Abstract

Semantic segmentation and other pixel-level labelling tasks have made significant progress recently due to the deep learning paradigm. Many state-of-the-art structured prediction methods also include a random field model with a hand-crafted Gaussian potential to model spatial priors, label consistencies and feature-based image conditioning. These random field models with image conditioning typically require computationally demanding filtering techniques during inference. In this paper, we present a new inference and learning framework which can learn arbitrary pairwise CRF potentials. Both standard spatial and high-dimensional bilateral kernels are considered. In addition, we introduce a new type of potential function which is image-dependent like the bilateral kernel, but an order of magnitude faster to compute since only spatial convolutions are employed. It is empirically demonstrated that such learned potentials can improve segmentation accuracy and that certain label-class interactions are indeed better modelled by a non-Gaussian potential. Our framework is evaluated on several public benchmarks for semantic segmentation with improved performance compared to previous state-of-the-art CNN+CRF models.

1 Introduction

Markov Random Fields (MRFs), Conditional Random Fields (CRFs) and more generally, probabilistic graphical models are a ubiquitous tool used in a variety of domains spanning Computer Vision, Computer Graphics and Image Processing [1–3]. In this paper, we focus on the application of MRFs for Computer Vision problems involving per-pixel labelling such as image segmentation. There are many successful approaches in this line of research, such as the interactive segmentation of [4] using graph cuts and the semantic segmentation works of [5, 6] where the parallel mean-field algorithm was applied for fast inference. Recently, Convolutional Neural Networks (CNNs) have dominated the field in a variety of recognition tasks [7–9]. However, we observe that several leading segmentation approaches still include CRFs, either as a post-processing step [10–13], or as part of the deep neural network itself [14–19].

We also leverage this idea of embedding inference of graphical models into a neural network. An early example of this idea was presented in [20] where the authors back propagated through the Viterbi algorithm when designing a document recognition system. Similar to [14, 16, 19, 21], we use a recurrent neural network to unroll the iterative inference steps of a CRF. This was first used in [14] and [22] to imitate mean-field inference and to train a fully convolutional network [10, 23] along with a CRF end-to-end via back propagation. In contrast to mean field, we do not optimize the KL-divergence between the true probabiity distribution and a fully-factorised approximation. Instead, we use a gradient descent approach for the inference that directly minimizes the Gibbs energy of the random field and hence avoids the approximations of mean-field. A similar framework was recently suggested in [21] and the followup work [24] for multi-label classification problems in machine learning with impressive results. Moreover, [25] have recently shown that one can obtain lower energies compared to mean-field inference using gradient descent based optimization schemes.

In many works, the pairwise potentials consist of parameterized Gaussians [14, 16, 26] and it is only the parameters of this Gaussian which are learned. Our framework can learn arbitrary pairwise potentials which need not be Gaussian. In [27], a general framework for learning arbitrary potentials in deep structured model was proposed based on approximate ML learning. One of the advantages with that framework is that data likelihood is maximized in the learning process. However, this involves approximating the partition function which is otherwise intractable. This hinders the handling of large structured output spaces like in our case.

Another approach to learning arbitrary pairwise potentials was presented in [18] which uses Gibbs sampling. Again they struggle with the difficulty of computing the partition function. In the end, only experiments on synthetic data restricted to learned 2D potentials are presented.

The authors of [15] and [13] also learn arbitrary pairwise potentials to model contextual relations between parts of the image. However, their approaches still perform post-processing with a CRF model with parametric Gaussian potentials. In [28], a pairwise potential is learned based on sparse bilateral filtering. Applying such a filter can be regarded as one iteration in the CRF inference step. In [28], the bilateral filter is applied twice, mimicking the first two iterations of inference. Our method is not restricted to a limited number of iterations. Perhaps more importantly is that we not only learn sparse high-dimensional bilateral filters, but also learn arbitrary spatial filters. Such spatial 2D potentials are computationally much more efficient and easier to analyze and interpret compared to their high-dimensional counterparts. We also note that [29] proposed back propagating through mean-field inference to learn parameters. However, this was not in the context of neural networks as in the aforementioned approaches and our work. For pixel-labelling tasks, we focus on discrete random fields. We note that learning arbitrary pairwise potentials for deep structured models with continuous valued output variables has recently been explored by [19].

A major drawback with using image dependent dense CRFs is the relatively high computation cost. Calculating the contribution of a bilateral kernel requires a filtering operation in 5D-space. Something that is very computationally expensive, even utilizing sophisticated approximate filtering techniques such as the permutohedral lattice filtering technique [30]. Since the image dependent CRF usually performs very well, especially when it comes to aligning object boundaries in segmentation tasks, it is still used for these tasks. In this paper we also propose an alternative CRF model which is also image dependent but only requires 2D convolutions during inference. The image dependence of the model comes from a output map of the base CNN that acts as a "filter selection" map. This enables the model to, for example, use one filter representing pairwise interaction between pixel labels at semantic edges and another filter far away from semantic edges.

Previous approaches trying to find alternatives to the computation heavy bilateral CRF include [31] where they use discriminatively trained domain transform as an edge-preserving filtering method. The authors show that the domain transform can be applied as a Recurrent Neural Network (RNN) applied across the image across all directions. Another example is [32] where they add a final layer that performs random graph walk across the image refining the segmentation.

In summary, our contributions are as follows.

- We present a new model for a pairwise CRF potential which is image-dependent like the bilateral kernel, but does not require high-dimensional filtering. It is based on a learned 2D filter bank which makes both inference and learning an order of magnitude faster than high-dimensional filtering approaches.
- We introduce a new optimization method for CRF inference based on gradient descent that enables end-to-end training.
- We show that our inference method supports learning pairwise kernels of arbitrary shape. The learned kernels are empirically analyzed and it is demonstrated that in many cases non-Gaussian potentials are preferred.

Our framework has been implemented in Caffe [33] and all source code will be made publicly available to facilitate further research.

2 CRF Formulation

Consider a Conditional Random Field over N discrete random variables $\mathcal{X} = \{X_1, ..., X_N\}$ conditioned on an observation $\boldsymbol{I}$ and let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an undirected graph whose vertices are the random variables $\{X_1, ..., X_N\}$. Each random variable corresponds to a pixel in the image and takes values from a predefined set of L labels $\mathcal{L} = \{0, ..., L-1\}$. The pair $(\mathcal{X}, \boldsymbol{I})$ is modelled as a CRF characterized by the Gibbs distribution

$$P(\mathcal{X} = \boldsymbol{x}|\boldsymbol{I}) = \frac{1}{Z(\boldsymbol{I})} \exp(-E(\boldsymbol{x}|\boldsymbol{I})), \tag{1}$$

where $E(\boldsymbol{x}|\boldsymbol{I})$ denotes the Gibbs energy function with respect to the labeling $\boldsymbol{x} \in \mathcal{L}^N$ and $Z(\boldsymbol{I})$ is the partition function. To simplify notation the conditioning on $\boldsymbol{I}$ will from now on be dropped. The MAP inference problem for the CRF model is equivalent to the problem of minimizing the energy $E(\boldsymbol{x})$. In this paper, we only consider energies containing unary and pairwise terms. The energy function can hence be written as

$$E(\boldsymbol{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \tag{2}$$

where $\psi_i : \mathcal{L} \to \mathbb{R}$ and $\psi_{ij} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ are the unary and pairwise potentials, respectively. We now describe these potentials before discussing inference in Sec. 3.

2.1 Potentials

The unary potential $\psi_i(x_i)$ specifies the energy cost of assigning label x_i to pixel i. In this work we obtain our unary potentials from a CNN. Roughly speaking, the CNN outputs a probability estimate of each pixel containing each class. Denoting the output of the CNN for pixel i and class x_i as $z_{i:x_i}$, the unary potential is

$$\psi_i(x_i) = -w_u \log(z_{i:x_i} + \epsilon) \tag{3}$$

where w_u is a parameter controlling the impact of the unary potentials, and ϵ is introduced to avoid numerical problems.

The pairwise potential $\psi_{ij}(x_i, x_j)$ specifies the energy cost of assigning label x_i to pixel i while pixel j is assigned label x_j. Introducing pairwise terms in our model enables us to take dependencies between output variables into account. We consider two alternative types, the *combined* and the *filterbank* versions.

Combined

The *combined* version has pairwise potentials that consist of a sum of one spatial term and one bilateral term. It has the following form

$$\psi_{ij}(x_i, x_j) = k^{spatial}_{x_i,x_j}(\boldsymbol{p}_i - \boldsymbol{p}_j) + k^{bilateral}_{x_i,x_j}(\boldsymbol{f}_i - \boldsymbol{f}_j) \tag{4}$$

Here $k^{spatial}_{x_i,x_j}$ denote a spatial kernel with compact support. Its value depends on the relative position coordinates $\boldsymbol{p}_i - \boldsymbol{p}_j$ between pixels i and j. We do not restrict these spatial terms to any specific shape. However we restrict the support of the potential meaning that if pixels i and j are far apart, then the value of $k^{spatial}_{x_i,x_j}(\boldsymbol{p}_i - \boldsymbol{p}_j)$ will be zero. We choose to use spatial kernels with compact support in contrast to the commonly used dense Gaussian potential since this allows the inference calculations to be performed using standard 2D convolutions. The CRFs with Gaussian potentials do not in theory have compact support, and therefore, they are often referred to as dense. However, in practice, the exponential function in the kernel drops off quickly and effectively, the interactions between pixels far apart are negligible.

The term $k^{bilateral}_{x_i,x_j}$ is a bilateral kernel which depends on the feature vectors $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ for pixels i and j, respectively. Following several previous works on random fields, we let the vector depend on pixel coordinates $\boldsymbol{p}_i$ and RGB values associated to the pixel, hence $\boldsymbol{f}_i$ is a 5-dimensional vector. Note that for both the spatial and the bilateral kernels, there is one kernel for each label-to-label (x_i and x_j) interaction to enable the model learn differently shaped kernels for each of these interactions.

Filterbank

The pairwise potentials of the *filterbank* version has the following form

$$\psi_{ij}(x_i, x_j) = \sum_{f=1}^{F} g_f(\boldsymbol{p}_i, I) k^{spatial}_{x_i,x_j,f}(\boldsymbol{p}_i - \boldsymbol{p}_j), \tag{5}$$

where $k^{spatial}_{x_i,x_j,f}$ denote a spatial kernel with compact support similar to the case of the *combined* version. The weights g_f depends both on the position of the pixel as well as the image I. These weights are taken as the output of a CNN. Hence, this gives rise to an image-dependent potential, but one only needs convolve with a bank of F 2D filters to evaluate it during inference. For example, the CNN outputting the weights can learn to detect semantic edges meaning that we would apply a different spatial filter close to a semantic edge than at the center of an semantic object. Setting the last layer of the CNN as a softmax the features $\boldsymbol{g_f}$ act as "filter selectors" deciding how the several 2d-filters describing the pairwise term should b weighted for each pixel individually.

2.2 Multi-label Graph Expansion and Relaxation

To be able to explain our inference method we reformulate the original minimization of $E(\boldsymbol{x})$ as a real-valued optimization problem. To facilitate a continuous relaxation of the energy minimisation problem we start off by expanding our original graph in the following manner. Each vertex in the original graph $\mathcal{G}$ will now be represented by L vertices $X_{i:\lambda}$, $\lambda \in \mathcal{L}$. In this way, an assignment of labels in $\mathcal{L}$ to each variable X_i is equivalent to an assignment of boolean labels 0 or 1 to each node $X_{i:\lambda}$, whereby an assignment of label 1 to $X_{i:\lambda}$ means that in the multi-label assignment, X_i receives label λ. To ensure that only one label is assigned to each node, an additional constraint is needed saying that, for each i, only one of $X_{i:\lambda}$ are allowed to be labeled 1. This enables to rewrite the energy minimization problem $\min E(\boldsymbol{x})$ as the following equivalent integer program

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{V}, \lambda \in \mathcal{L}} \psi_i(\lambda) x_{i:\lambda} + \sum_{\substack{(i,j) \in \mathcal{E} \\ \lambda, \mu \in \mathcal{L}}} \psi_{ij}(\lambda, \mu) x_{i:\lambda} x_{j:\mu} \\ \text{s.t.} \quad & x_{i:\lambda} \in \{0, 1\} \qquad \forall i \in \mathcal{V}, \lambda \in \mathcal{L} \\ & \sum_{\lambda \in L} x_{i:\lambda} = 1 \qquad \forall i \in \mathcal{V}. \end{aligned} \tag{6}$$

As a next step, we relax the integer program by allowing real values on the unit interval $[0, 1]$ instead of booleans only. We denote the relaxed variables $q_{i:\lambda} \in [0, 1]$. We can now write our problem as a quadratic program

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{V}, \lambda \in \mathcal{L}} \psi_i(\lambda) q_{i:\lambda} + \sum_{\substack{(i,j) \in \mathcal{E} \\ \lambda, \mu \in \mathcal{L}}} \psi_{ij}(\lambda, \mu) q_{i:\lambda} q_{j:\mu} \\ \text{s.t.} \quad & q_{i:\lambda} \geq 0 \qquad \forall i \in \mathcal{V}, \lambda \in \mathcal{L} \\ & \sum_{\lambda \in \mathcal{L}} q_{i:\lambda} = 1 \qquad \forall i \in \mathcal{V}. \end{aligned} \tag{7}$$

The two constraints can by summarized as $\boldsymbol{q}_i \in \triangle^L$, $\forall i \in \mathcal{V}$ where $\triangle^L$ is the probability simplex and L is the number of classes. A natural question is what happens when the domain is enlarged. Somewhat surprisingly, the relaxation is tight [34].

Proposition 2.1. *Let $E(\boldsymbol{x}^*)$ and $E(\boldsymbol{q}^*)$ denote the optimal values of* (6) *and* (7)*, respectively. Then,*

$$E(\boldsymbol{x}^*) = E(\boldsymbol{q}^*).$$

In the supplementary material, we show that for *any* real $\boldsymbol{q}$, one can obtain a binary $\boldsymbol{x}$ such that $E(\boldsymbol{x}) \leq E(\boldsymbol{q})$. In particular, it will be true for $\boldsymbol{x}^*$ and $\boldsymbol{q}^*$, which implies $E(\boldsymbol{x}^*) = E(\boldsymbol{q}^*)$. Note that the proof is constructive.

3 MAP Inference via Gradient Descent Minimization

To solve the program stated in (7) we propose an optimization scheme based on projected gradient descent, see Algorithm 1. It was designed with an extra condition in mind, that all operations should be differentiable to enable back propagation during training.

```
Initialize $\boldsymbol{q}^0$
for $t$ from 0 to $T-1$ do
    Compute the gradient $\nabla_{\boldsymbol{q}} E(\boldsymbol{q}^t)$.
    Take a step in the negative direction, $\tilde{\boldsymbol{q}}^{\boldsymbol{t+1}} = \mathbf{q^t} - \gamma \nabla_{\mathbf{q}} \boldsymbol{E}$.
    Project $\tilde{q}_{i:\lambda}^{t+1}$ to the probability simplex $\triangle^L$. $\boldsymbol{q}^{t+1} = \mathrm{Proj}_{\triangle^L}(\tilde{\boldsymbol{q}})$.
end for
return $\boldsymbol{q}^{T-1}$
```

Algorithm 1: Algorithm 1. Projected gradient descent algorithm.

3.1 Gradient Computations

The gradient $\nabla_{\boldsymbol{q}} E$ of the objective function $E(\boldsymbol{q})$ in (7) has the following elements

$$\frac{\partial E}{\partial q_{i:\lambda}} = \psi_i(\lambda) + \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} \psi_{ij}(\lambda,\mu) q_{j:\mu}. \tag{8}$$

The contribution from the spatial kernel in ψ_{ij}, cf. (4), can be written as

$$v_{i:\lambda}^{spatial} = \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} k_{\lambda,\mu}^{spatial}(\boldsymbol{p}_i - \boldsymbol{p}_j) q_{j:\mu}. \tag{9}$$

Since the value of the kernel $v_{i:\lambda}^{spatial}$ only depends on the relative position of pixels i and j, the contribution for all pixels and classes can be calculated by passing $q_{j:\mu}$ through a standard convolution layer consisting of $L \times L$ filters of size $(2s+1)\times(2s+1)$ where L is the number of labels and s the number of neighbours each pixel interacts with in each dimension.

The contribution from the bilateral term is

$$v_{i:\lambda}^{bilateral} = \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} k_{\lambda,\mu}^{bilateral}(\boldsymbol{f}_i - \boldsymbol{f}_j) q_{j:\mu}. \tag{10}$$

For this computation we utilize the method presented by Jampani *et al.* [28] which is based on the permutohedral lattice introduced by Adams *et al.* [30]. Efficient computations are obtained by using the fact that the feature space is generally sparsely populated. Similar to the spatial filter we get $L \times L$ filters, each having size of $(s+1)^{d+1} - s^{d+1}$ where s is the number of neighbours each pixel interacts with in each dimension in the sparse feature space.

For the *filter bank* version the contribution of the pairwise term can be calculated as

$$v_{i:\lambda}^{bank} = \sum_{f=1}^{F} g_f(\boldsymbol{p}_i, I) \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} k_{x_i,x_j,f}^{spatial}(\boldsymbol{p}_i - \boldsymbol{p}_j) q_{j:\mu}, \tag{11}$$

which, similar to the other spatial kernel can be efficiently calculated using a standard convolution layer. The number of filters needed is $L \times FL$.

3.2 Update Step and Projection to Feasible Set

Given the energy gradient and a previous estimate of the solution we want to improve our solution by taking a step in that decreases the energy while still keeping the solution feasible. A straightforward approach of doing this would be to start by taking a step in the negative direction of the gradient according to

$$\tilde{\boldsymbol{q}}^{\boldsymbol{t+1}} = \mathbf{q}^{\mathbf{t}} - \gamma \, \nabla_{\mathbf{q}} \boldsymbol{E}, \tag{12}$$

where γ is the the step size. After taking the step the values are projected onto the simplex $\triangle^L$ satisfying $\sum_{\lambda\in\mathcal{L}} q_{i:\lambda} = 1$ and $0 \leq q_{i:\lambda} \leq 1$ by following the method by Chen *et al.* [35]. This method is used by by Larsson *et al.* in [36]. A drawback with this approach is that, if $\tilde{\boldsymbol{q}}^{\boldsymbol{t+1}}$ is outside of the simplex, backpropagation through the projection method will give zero gradients.

An alternative method is to use the entropic descent algorithm proposed by Beck *et al.* [37]. In this method, the distance measure for the projection is the Kullback-Leibler divergence in contrast to the Euclidean distance. Beck *et al.* showed that the update step can be written on the following closed form

$$q_{ij}^{k+1} = \frac{q_{ij}^k \exp\left(-t_k \nabla_{q^k} E\right)}{\sum q_{ij}^k \exp\left(-t_k \nabla_{q^k} E\right)}, \quad t_k = \frac{\sqrt{2 \ln n}}{L_f} \frac{1}{\sqrt{k}} \tag{13}$$

where n is the number of dimensions (the number of classes in our case), k is the iteration number and L_f is a tunable parameter. Note that this projection is done individually for each pixel i.

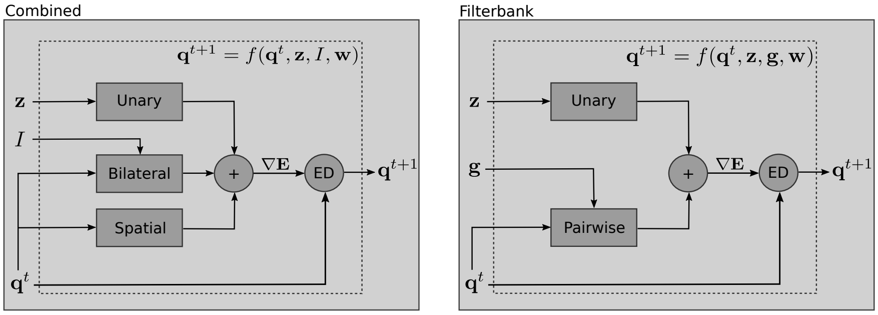

Figure 1: *The data flow of one iteration of the projected gradient descent algorithm. Each rectangle or circle represent an operation that can be performed within a deep learning framework, the ED component performs an entropic descent update step according to equation* (13). *Left: Combined version of CRF, Right: Filterbank version of CRF.*

3.3 Comparison to Mean-Field.

In recent years, a popular choice for CRF inference is to apply the mean-field algorithm. One reason is that the kernel evaluations can be computed with fast bilateral filtering [5]. As we have seen in this section, it can be accomplished with our framework as well, with formulas that are less involved. The main difference is that our framework directly optimizes the Gibbs energy which corresponds to MAP while mean-field optimizes KL-divergence which does not.

4 Integration in a Deep Neural Network

In this section we will describe how the steps of Algorithm 1 can be formulated as layers in a neural network. For this, we need to be able to calculate error derivatives with respect to the input given error derivatives with respect to the output. In addition we need to be able to calculate the error derivatives with respect to the network parameters, i.e. the filter weights for the pairwise kernels as well as the unary weight. This will enable us to unroll the entire gradient descent process as a Recurrent Neural Network (RNN) making it possible to train both the parameters of the CRF as well as the parameters of the CNN that gives the unary potentials as well as $\boldsymbol{g}$, the filter weighting function. A schematic of the data flow for one step is shown in Fig. 1. In the supplementary material, all derivative formulas are written out in detail.

4.1 Initialization.

The variables $\boldsymbol{q}^0$ are set as the output of the CNN, which has been pretrained to estimate the probability of each pixel containing each class and has a softmax layers as the last layer to ensure that the variables lies within zero and one.

4.2 Gradient Computations.

We have previously explained the gradient computations in Section 3 for the forward pass. To describe the calculation of the error derivatives we first notice that the gradient is calculated by summing the unary term and the pairwise term. We can hence treat these separately and combine them using an element-wise summing operation.

Unary Term. The unary term in (3) is an elementwise operation with the CNN output as input and the unary weight w_u as parameter. The operation is obviously differentiable with respect to both the layer input as well as its parameter. Note that for w_u we get a summation over all class and pixel indexes for the error derivatives while for the input the error derivatives are calculated elementwise.

Pairwise Term - Combined version. The spatial pairwise term of the gradient can be calculated efficiently using standard 2D convolution. In addition to giving us an efficient way of performing the forward pass we can also utilize the 2D convolution layer to perform the backward pass, calculating the error derivatives with respect to the input and parameters. Similar to the spatial term, the bilateral term is also calculated utilizing a bilateral filtering technique. Jampani *et al.* [28] also presented a way to calculate the error derivatives with respect to the parameters for an arbitrary shaped bilateral filter.

Pairwise Term - Filterbank version. For the Filterbank version we also use standard 2D convolution operations to calculate the pairwise part of the gradient. This makes the process of propagating the error derivatives similar as for the spatial term of the Combined version. Interpreting the calculations as two separate steps, one convolution with $L \times FL$ filters and one weighted summation over the feature weights, the error derivative can be calculated with standard network layers. Note that the error derivatives with respect to the feature weights g_f are also calculated and propagated further back through the pairwise CNN.

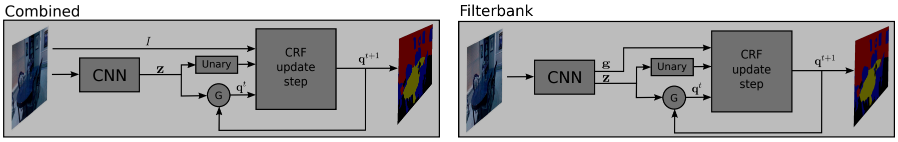

Figure 2: *The data flow of the deep structure model. Each rectangle or circle represent an operation that can be performed within a deep learning framework. Note that the CNN outputs both class probabilities $\boldsymbol{z}$ and filterbank features $\boldsymbol{g}$ for the filterbank version.*

4.3 Entropic Descent Update

The entropic descent step is done individually for each pixel. Since we have the update step on closed form we can easily implement it as a layer in a deep learning framework. Regarding the error derivative we are required to calculate both the error derivatives with respect to the values of the previous iteration, $\boldsymbol{q}^t$ and with respect to gradient, $\nabla_{q^t} E$. The error derivatives with respect to the values of the previous iteration are given according to

$$\frac{\partial L}{\partial q_{ij}^k} = \frac{q_{ij}^{k+1}}{q_{ij}^k} \left(\frac{\partial L}{\partial q_{ij}^{k+1}} - \sum_{l=1}^{n} \frac{\partial L}{\partial q_{il}^{k+1}} q_{il}^{k+1} \right), \tag{14}$$

where the index i is over all pixels and j is over all the n number of classes. Note that the error derivatives with respect to q_{ij}^{k+1} are given by the previous iteration. The error derivatives with respect to the gradient are given according to

$$\frac{\partial L}{\partial y_{ij}} = -t_k q_{ij}^{k+1} \left(\frac{\partial L}{\partial q_{ij}^{k+1}} - \sum_{l=1}^{n} \frac{\partial L}{\partial q_{il}^{k+1}} q_{il}^{k+1} \right). \tag{15}$$

Note that, for ease of notation, we have used y_{ij} as the energy derivative of pixel i and class j.

5 Recurrent Formulation as Deep Structured Model

Our iterative solution to the CRF energy minimisation problem by projected gradient descent, as described in the previous sections, is formulated as a Recurrent Neural Network (RNN). The input to the RNN is the image, and the outputs of the CNN, as shown in Fig. 2. The Unary CNN's output, **z**, are the unary potentials and obtained after the final softmax layer (since the CNN is initially trained for classification). For the filterbank version the CNN also outputs image-dependent features, **g**, which are "selecting" which filters to use to compose the pairwise term at each pixel location.

Each iteration of the RNN performs one projected gradient descent step to approximately solve (7). Thus, one update step can be represented by:

$$\boldsymbol{q}^{t+1} = f(\boldsymbol{q}^t, \boldsymbol{z}, I, \boldsymbol{w}). \tag{16}$$

As illustrated in Fig. 2, the gating function G sets $\boldsymbol{q}^t$ to $\mathbf{z}$ at the first time step, and to $\boldsymbol{q}^{t-1}$ at all other time steps. In our iterative energy minimisation, the output of one step is the input to the next step. We initialise at $t = 0$ with the output of the unary CNN.

The output of the RNN can be read off $\boldsymbol{q}^T$ where T is the total number of steps taken. In practice, we perform a set number of T steps where T is a hyperparameter. It is possible to run the RNN until convergence for each image (thus a variable number of iterations per image), but we observed minimal benefit in the final Intersection over Union (IoU) from doing so, as opposed to fixing the number of iterations to $T = 5$.

The parameters of the RNN are the filter weights for the pairwise kernels, and also the weight for the unary terms. Since we are able to compute error derivatives with respect to the parameters, and input of the RNN, we can backpropagate error derivates through our RNN to the preceding CNN and train our entire network end-to-end. Furthermore, since the operations of the RNN are formulated as filtering, training and inference can be performed in a fully-convolutional manner.

The CNN part of our network allows us to leverage the ability of CNNs to learn rich feature represenations from data, whilst the RNN part of the network utilises the CRF's ability to model output structure. As we learn the parameters of our pairwise terms, we are not restricted to Gaussian potentials as in [14, 26], and we show the benefits of this in our experiments (Section 7).

6 Implementation Details

Our proposed CRF model has been implemented in the Caffe [33] library. The Unary CNN part of our model is initialized form a pre-trained segmentation network. For all experiments we use the Deeplab-LargeFOV proposed by Chen *et al.* [38]. For the *combined* version of our model the unary CNN is pre-trained for pixel-wise classification.

For the *filterbank* version we use a modified version of the Deeplab-LargeFOV where a second head is added to the the network as in [31]. This head is formed by upsampling and concatenating several intermediate layers of the original network, a final convolution are applied to the concatenated features and lastly a softmax layer is added. The second head outputs the filter choosing features $\boldsymbol{g}_f$ and is pre-trained to classify each pixel as horizontal semantic edge, vertical semantic edge or no edge. This part of the base network can also be trained during the final end-to-end training.

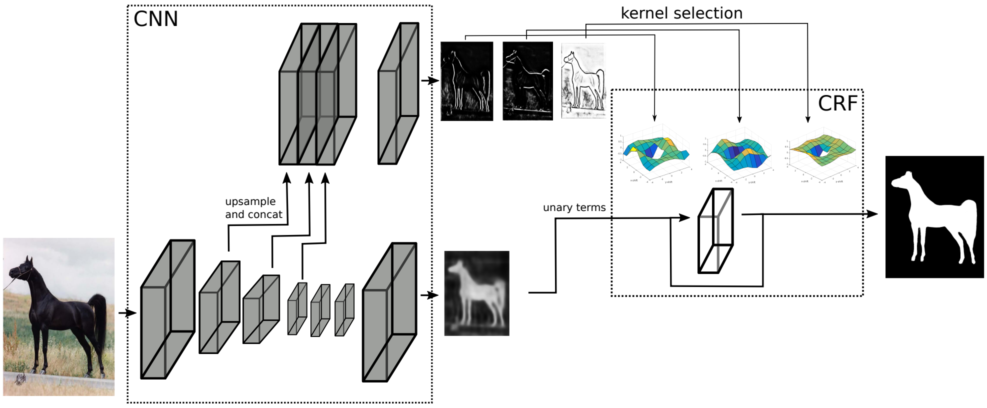

Figure 3: *Schematic of the filterbank version of our model. The CNN part outputs initial class probability maps as well as filter selection maps. The structure used for the CNN is a modified version of the Deeplab-LargeFOV [38] with an extra added head.*

The CRF model has several tunable hyperparameters. The parameter L_f and the number of iterations T specify the properties of the gradient decent algorithm. L_f influences the step size (larger L_f gives a smaller step size), too high a step size might make the algorithm not end up in a minimum while setting a low step size and a low number of iterations might not give the algorithm a chance to converge. The kernel sizes for the pairwise kernels also need to be set. Choosing the value of these parameters gives a trade-off between model expression ability and number of parameters, which may cause (or hinder) over-fitting.

The spatial weights of the CRF model are all initialized as zero with the motivation that we did not want to impose a shape for these filters, but instead see what was learned during training. The bilateral filters were initialized as Gaussians with the common Potts class interaction (the filters corresponding to interactions between the same class were set to zero) [5, 10, 14].

7 Experiments

We evaluate the proposed approach on three datasets: Weizmann Horse dataset [39], NYU V2 geometric dataset [40] and Pascal VOC 2012 [41]. In these experiments, we show that the proposed approach, has advantages over similar approaches such as CRF-RNN [14]. In addition we show that adding a CRF-model as proposed in this paper improves the results on strong unary CNN networks, even for cases where the CNN has been trained on large amounts of extra data.

Method	mIoU (%)
unary CNN - Deeplab [10]	90.89
CRF-RNN [14]	91.47
Gaussian-ED	92.64
Combined-ED	92.99
Combined-MF	92.73
Combined-PGD	92.79
Filterbank-ED	93.22

Table 1: *Quantitative results on the* Weizmann Horse *dataset comparing our method to baselines as well as comparison of different inference methods. Mean intersection over union for the test set is shown.*

7.1 Weizmann Horse

The Weizmann Horse dataset is widely used for benchmarking object segmentation algorithms. It contains 328 images of horses in different environments. We divide these images into a training set of 150 images, a validation set of 50 images and a test set of 128 images. Our purpose is to verify our ability to learn reasonable kernels and study the effects of different settings on a relatively small dataset. In addition we use this dataset to evaluate our proposed inference method as well as the different types of CRF-models. To compare the different types of inference methods train our *combined* model with three types of inference methods: Entropic Descent (ED), Projected Gradient Descent (PGD) and Mean Field (MF). We also train a version with only Gaussian potentials (using the same potentials as for CRF-RNN [14]). We also trained and evaluated the *filter-bank* version. The results are summarized in Table 1 and some example segmentations are shown in Fig. 4. As can be seen from the results, our proposed inference method using entropic descent achieves slightly better results on the test set for the *combined* CRF model. However, the increase over mean field and projected gradient descent inference is minor. For the case where we used Gaussian CRF potentials we get better results with entropic descent inference compared to mean field. Comparing entropic descent inference and projected gradient descent the two methods achieve similar results. Training a model with projected gradient descent is however problematic due to the zeroing of gradients, to solve this we train with a "leaky" version of projected gradient descent. This means that the intermediate states and final results might not lie on the probability simplex, something that is guaranteed for entropic descent inference.

In Fig. 5 the mean intersection over union on the test set is plotted as a function of the number of CRF inference iterations. During training the number

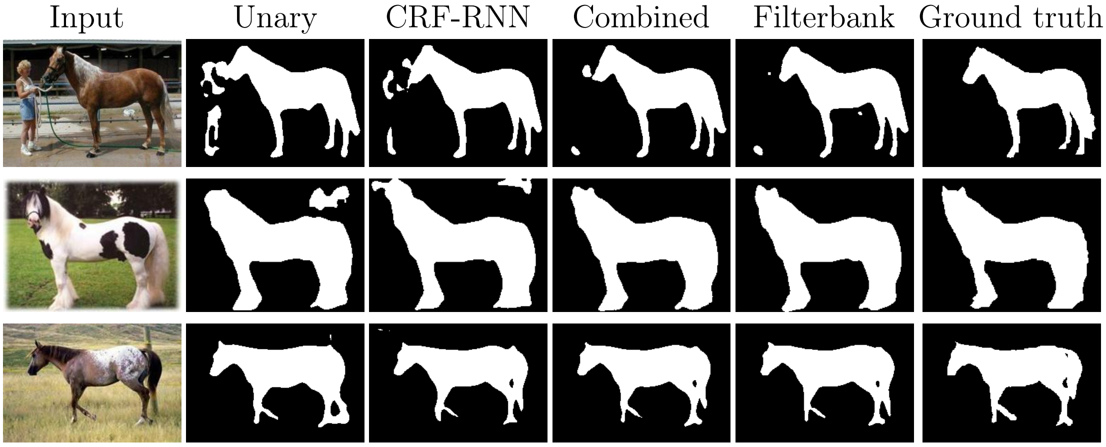

Figure 4: *Qualitative results on the* Weizmann Horse *dataset. Note that the proposed methods capture the shape of the horses better than the baselines, especially compared to the unary netwok.*

of iterations were set to five. As can be seen in the figure, increasing the number of inference step will only slightly increase the segmentation result. In Fig. 6 the pairwise weights of the *filterbank* version is visualized.

7.2 NYU V2

The NYU V2 dataset contains images taken by Microsoft Kinect V-1 camera in 464 indoor scenes. We use the official training and validation splits consisting of 795 and 654 images, respectively. Following the setting described in Wang *et al.* [40], we also include additional images for training. These are the images from the NYU V1 dataset that do not overlap with the images in the official validation set. This gives a total of 894 images with semantic label annotations for training. As in [40] we consider 5 classes conveying strong geometric properties: ground, vertical, ceiling, furniture and objects.

As shown in Table 2, we achieved superior results for semantic image segmentation on the NYU V2 dataset. Some example segmentations are shown in Fig. 8.

Method	mIoU (%)
R-CNN [42]	40.3
Semantic HCRF [40]	42.7
Joint HCRF [40]	44.2
Modular CNN [43]	54.3
unary CNN - Deeplab [10]	62.8
CRF-RNN [14]	64.4
Combined	65.4
Filterbank	65.4

Table 2: *Quantitative results comparing our method to baselines as well as state-of-the-art methods. Mean intersection over union for the validation set is shown for the* NYU V2 *dataset. The CRF-RNN baseline was initialized with the same unary network as the proposed models.*

Method	mIoU (%)
unary CNN - Deeplab [10]	68.5
CRF-RNN [14]	71.7
Combined	72.0
Filterbank	70.1

Table 3: *Quantitative results on the PASCAL VOC 2012 validation set. The CRF-RNN baseline was initialized with the same unary network as the proposed models. The unary model was pretrained on the MS-COCO 2014 dataset [44].*

Method	mIoU (%)
unary CNN - Deeplab [10]	68.9
DT-EdgeNet [31]	71.7
CRF-RNN [14]	72.2
Combined	72.5
Filterbank	69.5
PSPNet [46]	85.4
Multipath-RefineNet [47]	84.2

Table 4: *Quantitative results on the PASCAL VOC 2012 test set. The three top entries use the same base network as our models. The unary model was pretrained on the MS-COCO 2014 dataset [44], but note that our models were not trained using MS-COCO.*

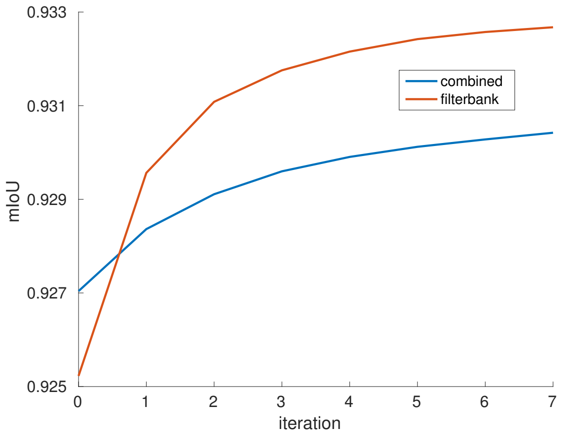

Figure 5: Weizmann Horse *test set results in terms of mean Intersection over Union plotted as a function of the number of iterations for the CRF inference method. During training the number of iterations were set to five.*

7.3 PASCAL VOC

The PASCAL VOC 2012 segmentation benchmark [45] consists of 20 foreground and one background class. The unary network used for these experiments is again the Deeplab-LargeFOV network [38], this network has been pretrained on the MS-COCO 2014 dataset [44] and then trained on the PASCAL VOC training data as well as a training set created from annotations of the semantic boundaries dataset [48]. We add our CRF-models to this baseline network and train only on the PASCAL VOC training data. This to show that we can improve upon really strong baselines, even though we finetune the complete models on only a fraction of the training data used for the baseline. The results for the PASCAL VOC 2012 validation set is shown in Table 3. In addition we evaluate our model on the test set, for this the results are shown in Table 4. As can be seen, our models perform similar to models trained with the same base network. Note that our models are only trained on the training data during end-to-end training. Recently there have several CNNs with different base architectures presented that perform well, even without a CRF. We include the three top entries in the table as well. We leave it to future work to explore whether these architectures can be improved using our proposed methodology.

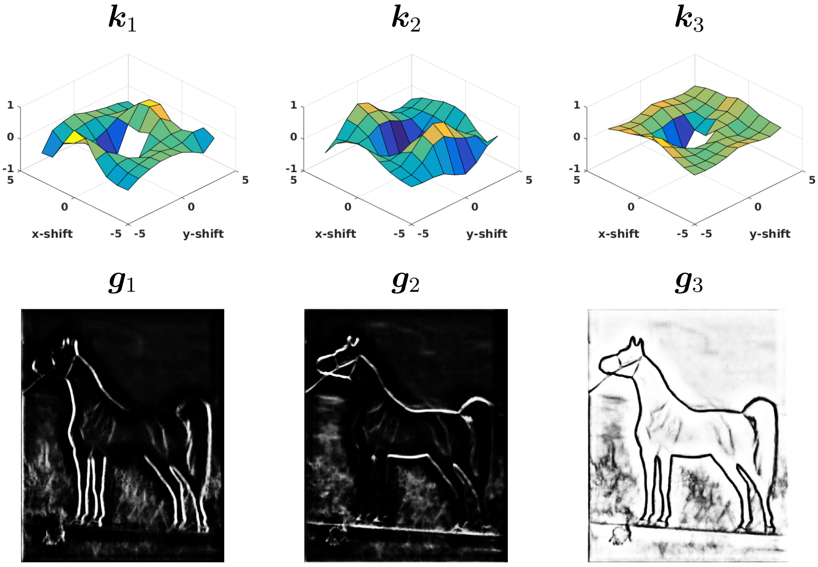

Figure 6: *Visualization of the pairwise kernel weights for the filterbank version trained on the* Weizmann Horse *data set. These weights are for the classes "background" and "horse", the plots can be understood as the energy added when assigning the pixels with the relative positions (x,y) and (x+x-shift,y+y-shift) as background and horse. This energy is then multiplied by the "filter choosing"-map* $\boldsymbol{g}$ *for each pixel and then summed. The first map of* $\boldsymbol{g}$ *has high values at edges in the horizontal direction, looking at* $\boldsymbol{k}_1$ *we see that changing classes in this direction does not add as much energy as changing classes in the vertical direction. Similar behaviour can be seen for the second map.*

7.4 Execution time

We also investigated the difference in running time between the two proposed models. This was done on a computer with a Nvidia Titan X GPU with Pascal architecture and an Intel i7-5930K processor. The implementation used for the bilateral filtering used was the one from Jampani *et al.* [28] where most of the computations are done on the GPU. The initialization of the permutohedral lattice is however done on the CPU. The runtimes were tested by performing the forward step for a randomized RGB image of size 640 × 640 with 21 classes. The numbers presented are the average of 100 runs. For the *combined* model the forward runtime was 12 seconds while for the *filterbank* it was 0.37 second.

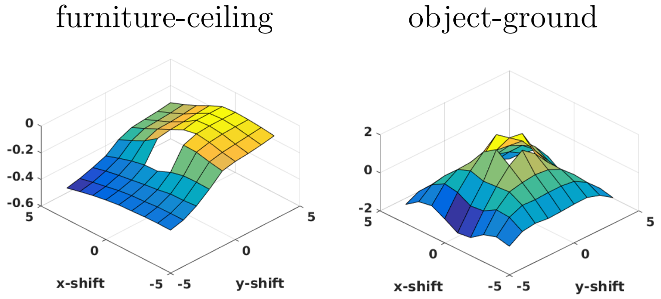

Figure 7: *Visualization of the pairwise kernel weights for the filterbank version trained on the* NYU V2 *data set. These weights are for the classes shown above the plots and for the third filter choosing map which usually has a high value for pixels with no semantic edge. The furniture-ceiling kernel favors putting furniture labels below ceiling labels while the obejct-ground kernel has a more Gaussian-like shape.*

8 Conclusion

In this paper we have presented a gradient descent based method for inference in Conditional Random Fields. This method allows for backpropagation of error derivative hence enabling end-to-end training with an Convolutional Neural Network of choice. We show that this inference method has beneficial properties and performs better on some tasks compared to other methods such as mean field. In addition, we present two types of Conditional Random Field models tailored for semantic segmentation. The *combined* model that uses spatial pairwise terms as well as image-dependent bilateral pairwise terms. This model performs well but is somewhat computational expensive due to the high dimensionality of the bilateral filtering. We also present the *filterbank* model which is also image dependent but only requires 2D convolutions during inference. The image dependence of the model comes from an output map of the base CNN that acts as a "filter choosing" map. This enables the model to, for example, use one filter representing pairwise interaction between pixel labels at semantic edges and another filter far away from semantic edges. This model gives a speedup by a factor of 32 compared to the *combined* model without loosing performance in terms of segmentation quality. For the smaller dataset it achieves similar segmentation quality as the *combined* model. For all the models presented the pairwise kernels can have arbitrary shape, instead of commonly used Gaussian kernels. This enables the models to learn more complicated pairwise label interactions.

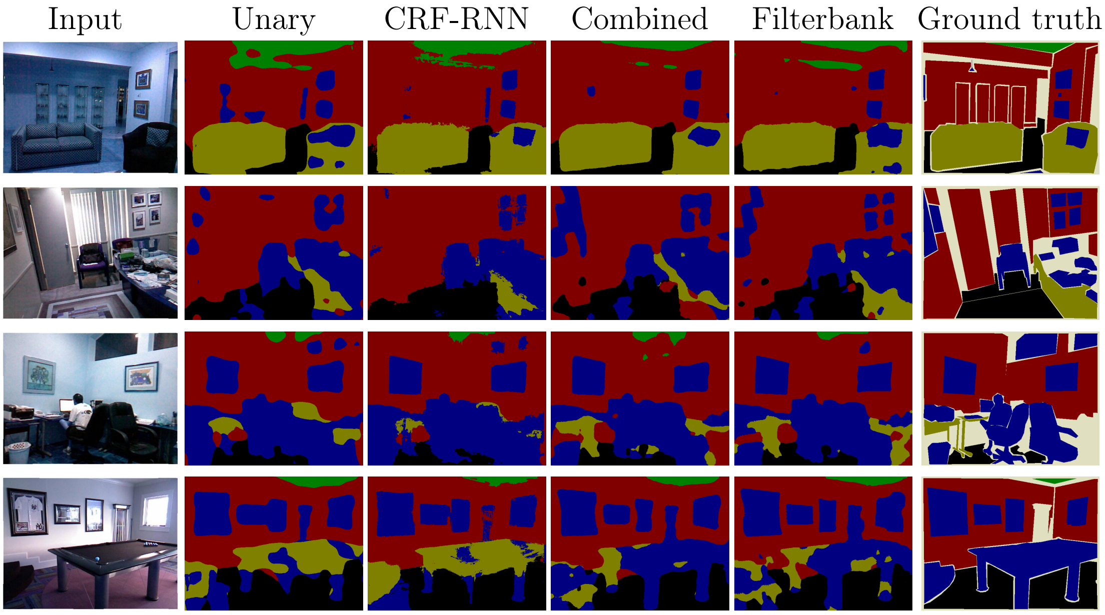

Figure 8: *Qualitative results on the* NYU V2 *dataset. Note that the proposed methods captures the shape of the object instances better than the baselines. This effect is perhaps most pronounced for the paintings hanging on the walls. The pixels colored off-white are "ignore"-pixels, these are not counted in the evaluation. The training images have similar "ignore"-pixels.*

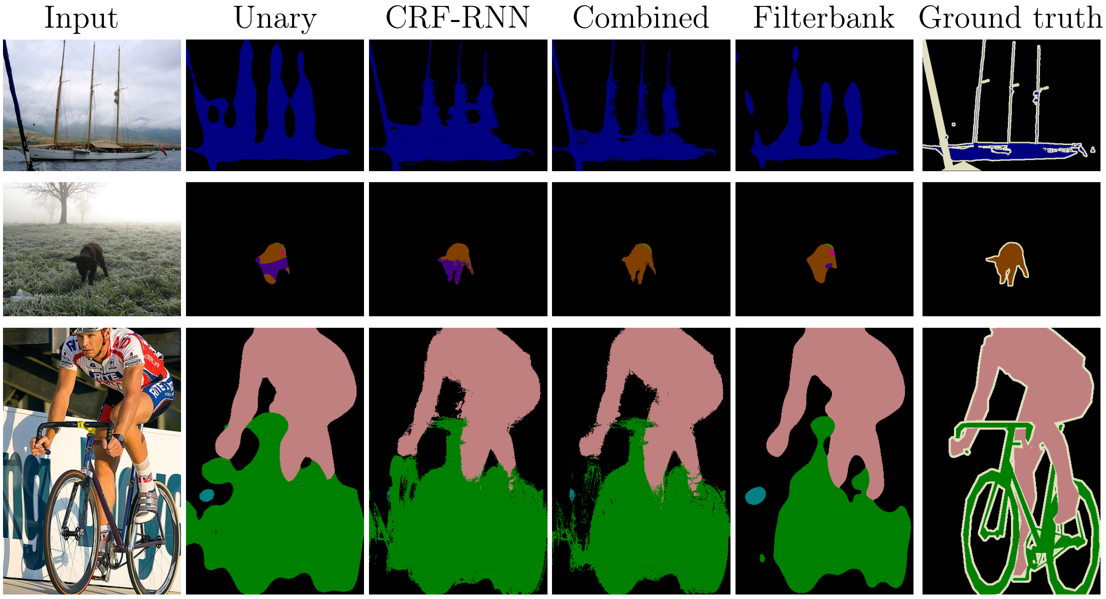

Figure 9: *Qualitative results on the* PASCAL VOC *dataset. [45]. The pixels colored off-white are "ignore"-pixels, these are not counted in the evaluation. The training images have similar "ignore"-pixels.*

Bibliography

[1] D. Koller and N. Friedman, *Probabilistic Graphical Models.* MIT Press, 2009.

[2] A. Blake, P. Kohli, and C. Rother, *Markov Random Fields for Vision and Image Processing.* MIT Press, 2011.

[3] A. Arnab, S. Zheng, S. Jayasumana, B. Romera-Paredes, M. Larsson, A. Kirillov, B. Savchynskyy, C. Rother, F. Kahl, and P. H. S. Torr, “Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction,” *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 37–52, Jan 2018.

[4] C. Rother, V. Kolmogorov, and A. Blake, “"GrabCut": Interactive foreground extraction using iterated graph cuts,” in *ACM Transactions on Graphics*, 2004, pp. 309–314.

[5] P. Krähenbühl and V. Koltun, “Efficient inference in fully connected CRFs with gaussian edge potentials,” in *Neural Information Processing Systems*, 2011.

[6] V. Vineet, J. Warrell, and P. Torr, “Filter-based mean-field inference for random fields with higher order terms and product label-spaces,” in *European Conference on Computer Vision*, 2012.

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Neural Information Processing Systems*, 2015.

[10] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” in *International Conference on Learning Representations*, 2015.

[11] ——, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *arXiv preprint arXiv:1606.00915*, 2016.

[12] G. Ghiasi and C. Fowlkes, "Laplacian reconstruction and refinement for semantic segmentation," in *European Conference on Computer Vision*, 2016.

[13] S. Chandra and I. Kokkinos, "Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs," in *European Conference on Computer Vision*, 2016.

[14] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional random fields as recurrent neural networks," in *International Conference on Computer Vision*, 2015.

[15] G. Lin, C. Shen, A. Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016.

[16] A. Arnab, S. Jayasumana, S. Zheng, and P. H. S. Torr, "Higher order conditional random fields in deep neural networks," in *European Conference on Computer Vision*, 2016.

[17] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang, "Semantic image segmentation via deep parsing network," in *International Conference on Computer Vision*, 2015.

[18] A. Kirillov, D. Schlesinger, S. Zheng, B. Savchynskyy, P. Torr, and C. Rother, "Joint training of generic cnn-crf models with stochastic optimization," in *Asian Conference on Computer Vision*, 2016.

[19] W. Wang, S. Fidler, and R. Urtasun, "Proximal deep structured models," in *Neural Information Processing Systems*, 2016.

[20] L. Bottou, Y. Bengio, and Y. Le Cun, "Global training of document processing systems using graph transformer networks," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1997, pp. 489–494.

[21] D. Belanger and A. McCallum, "Structured prediction energy networks," in *International Conference on Machine Learning*, 2016.

[22] A. Schwing and R. Urtasun, "Fully connected deep structured networks," in *arXiv preprint arXiv:1503.02351*, 2015.

[23] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[24] D. Belanger, B. Yang, and A. McCallum, "End-to-end learning for structured prediction energy networks," *arXiv preprint arXiv:1703.05667*, 2017.

[25] A. Desmaison, R. Bunel, P. Kohli, P. H. S. Torr, and M. P. Kumar, "Efficient continuous relaxations for dense CRF," in *European Conference on Computer Vision*, 2016.

[26] P. Kraehenbuehl and V. Koltun, "Parameter learning and convergent inference for dense random fields," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 513–521.

[27] L. Chen, A. Schwing, A. Yuille, and R. Urtasun, "Learning deep structured models," in *Int. Conf. Machine Learning*, Lille, France, 2015.

[28] V. Jampani, M. Kiefel, and P. V. Gehler, "Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016.

[29] J. Domke, "Learning graphical model parameters with approximate marginal inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2454–2467, 2013.

[30] A. Adams, J. Baek, and M. A. Davis, "Fast high-dimensional filtering using the permutohedral lattice," *Computer Graphics Forum*, 2010.

[31] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4545–4554.

[32] G. Bertasius, L. Torresani, S. X. Yu, and J. Shi, "Convolutional random walk networks for semantic image segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

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

[34] E. Boros and P. Hammer, "Pseudo-boolean optimization," *Discrete Appl. Math.*, vol. 123, pp. 155–225, 2002.

[35] Y. Chen and X. Ye, "Projection onto a simplex," *arXiv preprint arXiv:1101.6081*, 2011.

[36] M. Larsson, A. Arnab, F. Kahl, S. Zheng, and P. H. S. Torr, “A projected gradient descent method for crf inference allowing end-to-end training of arbitrary pairwise potentials,” in *11th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, (EMMCVPR)*. Springer, 2017.

[37] A. Beck and M. Teboulle, “Mirror descent and nonlinear projected subgradient methods for convex optimization,” *Operations Research Letters*, vol. 31, no. 3, pp. 167–175, 2003.

[38] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.” *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[39] E. Borenstein and S. Ullman, “Class-specific, top-down segmentation,” in *European Conf. on Computer Vision*, 2002.

[40] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille, “Towards unified depth and semantic prediction from a single image,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[41] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *Int. Journal Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

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

[43] O. H. Jafari, O. Groth, A. Kirillov, M. Y. Yang, and C. Rother, “Analyzing modular cnn architectures for joint depth prediction and semantic segmentation,” in *International Conference on Robotics and Automation*, 2017.

[44] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft COCO: Common Objects in Context*. Cham: Springer International Publishing, 2014, pp. 740–755.

[45] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan 2015. [Online]. Available: https://doi.org/10.1007/s11263-014-0733-5

[46] G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation,” *arXiv preprint arXiv:1611.06612*, 2016.

[47] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.

[48] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 991–998.

Supplementary Material

Proof of Proposition 2.1

Let $E(\boldsymbol{x}^*)$ and $E(\boldsymbol{q}^*)$ denote the optimal values of (7), where $\boldsymbol{x}^*$ is restricted to boolean values. Then,

$$E(\boldsymbol{x}^*) = E(\boldsymbol{q}^*).$$

Proof. We will show that for *any* real $\boldsymbol{q}$, one can obtain a binary $\boldsymbol{x}$ such that $E(\boldsymbol{x}) \leq E(\boldsymbol{q})$. In particular, it will be true for $\boldsymbol{x}^*$ and $\boldsymbol{q}^*$, which implies $E(\boldsymbol{x}^*) = E(\boldsymbol{q}^*)$.

Let $\boldsymbol{q}$ be given, and let $\boldsymbol{x} \in \mathcal{L}^N$. One may define

$$E^m(\boldsymbol{x}, \boldsymbol{q}) = E(x_1, \dots, x_m, q_{m+1}, \dots, q_N)$$

such that each x_i or q_i is a vector with entries such as $q_{i:\lambda}$ or $x_{i:\lambda}$, but for each i only one value $x_{i:\lambda}$ is non-zero (and equal to 1). Since $E^0 = E(\boldsymbol{q})$ and $E^N = E(\boldsymbol{x})$ it will be sufficient to find a $\boldsymbol{x}$ such that $E^m(\boldsymbol{x}, \boldsymbol{q}) \leq E^{m-1}(\boldsymbol{x}, \boldsymbol{q})$ for all m. The required $\boldsymbol{x}$ will be constructed one element at a time.

The key observation is that E^m is multilinear in the q_i. Then, it follows that

$$\begin{aligned} E^{m-1}(\boldsymbol{x}, \boldsymbol{q}) &= E(x_1, \dots, x_{m-1}, q_m, \dots q_N) \\ &= \sum_{x_m \in \mathcal{L}} q_{m:x_m} E(x_1, \dots, x_m, q_{m+1}, \dots q_N). \end{aligned}$$

Here, x_m is treated as a variable and $x_1, \dots, x_{m-1}$ are fixed. Since $\sum_{x_m \in \mathcal{L}} q_{m:x_m} = 1$ there must be at least one choice of x_m such that

$$E^{m-1}(\boldsymbol{x}, \boldsymbol{q}) \geq E(x_1, \dots, x_m, q_{m+1}, \dots q_m) = E^m(\boldsymbol{x}, \boldsymbol{q}).$$

It is clear that $E(\boldsymbol{q}^*) \leq E(\boldsymbol{x}^*)$ since the domain is enlarged, which proves the desired result. □

Error Derivatives for the CRF-Grad layer

In this section, we will explicitly formulate the error derivative necessary to train our deep structure model jointly. The following section describes the calculations for the combined version of the CRF, the changes needed for the filterbank version is described in the next paragraph. The notation used in the section is not very strict. Derivatives, gradients and jacobians are all referred to as derivatives. Denoting the output of our CRF-Grad layer $\boldsymbol{y}$ we need expressions for the derivatives $\frac{\partial \mathbf{y}}{\partial \mathbf{z}}$, where $\boldsymbol{z}$ is the output from the CNN and hence also the input to the CRF-Grad layer. In addition we need to calculate $\frac{\partial \mathbf{y}}{\partial w_u}$, $\frac{\partial \mathbf{y}}{\partial \mathbf{w}_s}$ and $\frac{\partial \mathbf{y}}{\partial \mathbf{w}_b}$ to be able to update the weights of the layer. To simplify the notation we abbreviate the update step by $\boldsymbol{q}^{t+1} = f(\boldsymbol{q}^t, \boldsymbol{z}, I, \boldsymbol{w})$. Note that the output $\boldsymbol{y} = \boldsymbol{q}^T$ where T is the total number of iterations for the RNN. We have

$$\frac{\partial \boldsymbol{y}}{\partial w_u} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^T}\frac{\partial f(\mathbf{q}^{T-1})}{\partial w_u} + \ldots + \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^1}\frac{\partial f(\mathbf{q}^0)}{\partial w_u} \tag{17}$$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{w}_s} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^T}\frac{\partial f(\mathbf{q}^{T-1})}{\partial \boldsymbol{w}_s} + \ldots + \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^1}\frac{\partial f(\mathbf{q}^0)}{\partial \boldsymbol{w}_s} \tag{18}$$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{w}_b} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^T}\frac{\partial f(\mathbf{q}^{T-1})}{\partial \boldsymbol{w}_b} + \ldots + \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^1}\frac{\partial f(\mathbf{q}^0)}{\partial \boldsymbol{w}_b} \tag{19}$$

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^0}\frac{\partial \boldsymbol{q}^0}{\partial \boldsymbol{z}} + \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\psi}_u}\frac{\partial \boldsymbol{\psi}_u}{\partial \boldsymbol{z}}, \tag{20}$$

where $\boldsymbol{\psi}_u$ denote the unary part of the CRF energy function. Note that

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^{t-1}} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{q}^t}\frac{\partial f(\mathbf{q}^{t-1})}{\partial \boldsymbol{q}^{t-1}} \tag{21}$$

To be able to calculate these we need the derivatives of the function f with respect to $\mathbf{q}^t$, w_u, $\boldsymbol{w}_s$ and $\boldsymbol{w}_b$. We denote the spatial and bilateral filtering operations as $\boldsymbol{\psi_s} * \boldsymbol{q}^t$ and $\boldsymbol{\psi_b} * \boldsymbol{q}^t$ respectively. An update step can then be written as

$$\boldsymbol{q}^{t+1} = \mathrm{ED}(\mathbf{q}^t, \nabla_{\boldsymbol{q}^t} E).$$

where ED denotes the entropic descent update step for which the expressions needed for backpropagation is shown in the main paper. The expression for calculating the derivative is

$$\nabla_{\boldsymbol{q}^t} E = \boldsymbol{\psi}_u + \boldsymbol{\psi}_s * \mathbf{q^t} + \boldsymbol{\psi}_b * \mathbf{q^t} \tag{22}$$

Denoting $\nabla_{\boldsymbol{q}^t} E$ as $\boldsymbol{e}^t$ the aforementioned derivatives become

$$\frac{\partial f}{\partial \mathbf{q}^t} = \frac{\partial \mathrm{ED}}{\partial \boldsymbol{q}^t} + \frac{\partial \mathrm{ED}}{\partial \boldsymbol{e}^t}\frac{\partial \boldsymbol{e}^t}{\partial \boldsymbol{q}^t}, \tag{23}$$

where

$$\frac{\partial \boldsymbol{e}^t}{\partial \boldsymbol{q}^t} = \frac{\partial(\boldsymbol{\psi}_s * \mathbf{q}^{\mathbf{t}})}{\partial \mathbf{q}^{\mathbf{t}}} + \frac{\partial(\boldsymbol{\psi}_b * \mathbf{q}^{\mathbf{t}})}{\partial \mathbf{q}^{\mathbf{t}}}. \tag{24}$$

For the weights

$$\frac{\partial f}{\partial w_u} = \frac{\partial \mathrm{ED}}{\partial \boldsymbol{e}^t} \frac{\partial \boldsymbol{e}^t}{\partial \boldsymbol{\psi}_u} \cdot \frac{\partial \boldsymbol{\psi}_u}{\partial w_u}, \tag{25}$$

$$\frac{\partial f}{\partial \boldsymbol{w}_s} = \frac{\partial \mathrm{ED}}{\partial \boldsymbol{e}^t} \cdot \frac{\partial(\boldsymbol{\psi}_s * \mathbf{q}^{\mathbf{t}})}{\partial \boldsymbol{w}_s}, \tag{26}$$

and

$$\frac{\partial f}{\partial \boldsymbol{w}_b} = \frac{\partial \mathrm{ED}}{\partial \boldsymbol{e}^t} \cdot \frac{\partial(\boldsymbol{\psi}_b * \mathbf{q}^{\mathbf{t}})}{\partial \boldsymbol{w}_b}, \tag{27}$$

Note that $\frac{\partial(\boldsymbol{\psi}_s * \mathbf{q}^{\mathbf{t}})}{\partial \boldsymbol{q}^t}$, $\frac{\partial(\boldsymbol{\psi}_b * \mathbf{q}^{\mathbf{t}})}{\partial \boldsymbol{q}^t}$, $\frac{\partial(\boldsymbol{\psi}_s * \mathbf{q}^{\mathbf{t}})}{\partial \boldsymbol{w}_s}$ and $\frac{\partial(\boldsymbol{\psi}_b * \mathbf{q}^{\mathbf{t}})}{\partial \boldsymbol{w}_b}$ can be calculated using the backward routines for a standard convolutional layer and bilateral filtering layer described in the main paper.

Filterbank Version

For the filterbank version we don't have a bilateral term for our gradient. Also, the pairwise part of the gradient is calculated by first doing a standard convolution operation and later a weighted sum over the feature values. Since we can divide this into three separate operations: convolution, element-wise multiplication and summing, the expressions for backpropagating the error can be derived from these separate operations. Hence we leave the full explicit expression out of the supplementary material. Note though that we also calculate the error derivative for the "filter choosing"-features enabling backpropagation through that part of the CNN as well.

Paper II

Max-Margin Learning of Deep Structured Models for Semantic Segmentation

M. Larsson, J. Alvén and F. Kahl

In Image Analysis: 20th Scandinavian Conference, SCIA 2017, Springer International Publishing, 28-40

Comment: The layout of this paper has been reformatted in order to comply with the rest of the thesis.

Max-Margin Learning of Deep Structured Models for Semantic Segmentation

M. Larsson, J. Alvén and F. Kahl

Abstract

During the last few years most work done on the task of image segmentation has been focused on deep learning and Convolutional Neural Networks (CNNs) in particular. CNNs are powerful for modeling complex connections between input and output data but lack the ability to directly model dependent output structures, for instance, enforcing properties such as smoothness and coherence. This drawback motivates the use of Conditional Random Fields (CRFs), widely applied as a post-processing step in semantic segmentation.

In this paper, we propose a learning framework that jointly trains the parameters of a CNN paired with a CRF. For this, we develop theoretical tools making it possible to optimize a max-margin objective with back-propagation. The max-margin loss function gives the model good generalization capabilities. Thus, the method is especially suitable for applications where labelled data is limited, for example, medical applications. This generalization capability is reflected in our results where we are able to show good performance on two relatively small medical datasets. The method is also evaluated on a public benchmark (frequently used for semantic segmentation) yielding results competitive to state-of-the-art. Overall, we demonstrate that end-to-end max-margin training is preferred over piecewise training when combining a CNN with a CRF.

1 Introduction

Convolutional Neural Networks (CNNs) have, during the last few years, been used with great success on a variety of computer vision problems such as image classification [1] and object detection [2]. The capability of CNNs to learn high-level abstraction of data makes them well suited for the task of image classification. Following this development, there have been several successful attempts to extend CNN based methods to tasks done on the pixel level such as semantic segmentation [3–5].

A drawback of CNNs is that they do not have the ability to directly model statistical dependencies of output variables. Hence they cannot explicitly enforce smoothness constraints or encourage spatial consistency of the output, something that arguably is important for the task of semantic segmentation. To deal with this a Markov Random Field (MRF), or its variant Conditional Random Field (CRF), can be used as a refinement step. This was done by Chen *et al.* in [6] where they used CNNs to form the unary potential of the dense CRF model presented by Krähenbühl *et al.* in [7]. However, the CNN and the CRF models are trained separately in [6] meaning that the parameters of the CRF are learnt while holding the CNN weights fixed. In other words, the deep features are learnt disregarding statistical dependencies of the output variables. In reaction to this, several approaches for jointly training deep structured models, combining CNNs and CRFs, have recently been proposed [8–12]. In these approaches, as well as the one presented in this paper, the parameters of the CRF and the weights of the CNN can be trained jointly, enabling the possibility to learn deep image features taking dependencies of the output variables into account.

1.1 Contributions

What differentiates this paper from previous work done on learning deep structured models is mainly the joint learning algorithm. We apply a max-margin based learning framework inspired by [13]. This removes the need to calculate, or approximate, the partition function present in learning algorithms that try to maximize the log-likelihood. For instance, in [9, 12], the inference step is approximately solved using a few iterations of the mean-field algorithm or gradient descent, respectively. Similarly, in [8], sampling techniques are used to approximate the partition function. In our learning framework, we can use standard graph cut methods to perform optimal inference in the CRF model. We also show how the CNN weights can be trained to optimize the max-margin criterion via standard back-propagation. To our knowledge, we are the first to present a method for jointly training deep structured models with a max-margin objective for semantic segmentation.

Our experiments show that training deep structured models using our method gives better results than piecewise training where the CNN and CRF models are trained separately. This proves that training deep structured models jointly enables the model to learn deep features that take output dependencies into account which in turn gives better segmentations. We tested our method on the Weizmann Horse dataset [14] for proof of concept. In addition we applied it to two medical datasets, one for heart ventricle segmentation in ultrasound images and one for pericardium segmentation in CTA slices.

1.2 Related Work

The concept of deep structured models has been examined extensively in recent work. In [15] Ning *et al.* combine a CNN with an energy based model, similar to a MRF, for segmentation of cell nuclei and in [6] a dense CRF with unary potentials from a CNN is used to achieve state-of-the-art results on several semantic segmentation benchmarks.

Methods for jointly training these deep structured models have also received a lot of attention lately. In [11] Tompson *et al.* present a single learning framework unifying a novel ConvNet Part-Detector and an MRF inspired Spatial-Model achieving state-of-the-art performance on the task of human body pose recognition. Further, Chen *et al.* present a more general framework for joint learning of deep structured models that they apply to image tagging and word from image problems in [8]. Zheng *et al.* [12] show that the mean-field inference algorithm with Gaussian pairwise potentials from [7] can be modeled as a Recurrent Neural Networks. This enabled them to train their model within a standard deep learning framework using a log-likelihood loss. In [10], they formulated a CRF model with CNNs for estimating the unary and pairwise potentials via piecewise training.

In the field of medical image analysis, methods based on CNNs have also received an increased interest during the last few years with promising results [16–18]. Recently, more intricate deep learning approaches have been proposed. Ronneberger *et al.* [19] proposed the U-Net, a network based on the idea of "fully convolutional networks" [4]. A similar network structure was also proposed by Brosch *et al.* in [20]. However, to our knowledge, methods utilizing end-to-end training of deep structured models have yet to be presented for medical image segmentation tasks.

2 A Deep Conditional Random Field Model

The deep structured model proposed in this paper consists of a CNN coupled with a CRF. This setup allows the model to learn deep features while still taking dependencies in the output data into account. Denote the set of input instances by $X = \{\boldsymbol{x}^{(n)}\}_n$ and their corresponding labelings by $Y = \{\boldsymbol{y}^{(n)}\}_n$. The input and output instances are images indexed for each pixel by $\boldsymbol{x}^{(n)} = (x_1^{(n)}, \dots, x_N^{(n)})$ and $\boldsymbol{y}^{(n)} = (y_1^{(n)}, \dots, y_N^{(n)})$ respectively. We only consider the binary labeling case, hence $y_i^{(n)} = \{0, 1\}$. Our deep structured model is described by a CRF of the form

$$P(Y|X; \boldsymbol{w}, \boldsymbol{\theta}) = \frac{1}{Z} e^{-\sum_n E(\boldsymbol{y}^{(n)}, \boldsymbol{x}^{(n)}; \boldsymbol{w}, \boldsymbol{\theta})}, \tag{1}$$

where $\boldsymbol{w}$ are the weights of the CRF, $\boldsymbol{\theta}$ are the weights of the CNN and Z is the partition function. The energy E considered decomposes over unary and pairwise terms according to the following form

$$E(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{i \in \mathcal{V}} \boldsymbol{\phi}_i(y_i, \boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{\phi}_{ij}(y_i, y_j, \boldsymbol{x}; \boldsymbol{w}), \tag{2}$$

where $\mathcal{V}$ is the set of nodes (*i.e.* pixels) and $\mathcal{E}$ is the set of edges connecting neighbouring pixels.

The unary term of the energy E has the following form

$$\boldsymbol{\phi}_i(y_i, \boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) = w_1 \log(\Phi_i(y_i, \boldsymbol{x}; \boldsymbol{\theta})), \tag{3}$$

where $\Phi_i(y_i, \boldsymbol{x}; \boldsymbol{\theta})$ denotes the output of the neural network for pixel i. There are no explicit requirements for the CNN except that it should output an estimate of the probability for each pixel being either foreground or background.

The pairwise term consists of two parts both penalizing two neighbouring pixels being labeled differently. The first part adds a constant cost while the other one adds a cost based on the contrast of the neighbouring pixels. If $\mathbf{1}_{y_i \neq y_j}$ denotes the indicator function equaling one if $y_i \neq y_j$, the pairwise term has the following form

$$\boldsymbol{\phi}_{ij}(y_i, y_j, \boldsymbol{x}; \boldsymbol{w}) = \mathbf{1}_{y_i \neq y_j} \left(w_2 + w_3 \, e^{-\frac{(x_i - x_j)^2}{2}} \right). \tag{4}$$

Note that, given these unary and pairwise terms, the energy is linear with respect to the weights $\boldsymbol{w}$.

2.1 Inference

Given an input instance $\boldsymbol{x}$, the inference problem equates to finding the maximum a posteriori labeling $\boldsymbol{y}^*$ given the model in (1). This is equivalent to finding a minimizer of the energy E in (2):

$$\boldsymbol{y}^* = \arg\min_{\boldsymbol{y}} E(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}). \tag{5}$$

For our deep structured model the inference is done in two steps. Firstly, an estimation of the probability of each pixel being either foreground or background is computed by a forward pass of the CNN. Secondly, problem (5) is solved. We add the constraints $w_i \geq 0$, $i = 2, 3$ when learning the weights to make the energy E submodular. This means that graph cut algorithm can be used to efficiently find a global optimum [21].

2.2 Max-Margin Learning

There are two sets of learnable parameters, the weights of the CRF $\boldsymbol{w}$ and the weights of the CNN $\boldsymbol{\theta}$. The method of learning is based on an algorithm proposed by Szummer *et al.* [13] where the goal is to find a set of parameters $\boldsymbol{w}, \boldsymbol{\theta}$ such that

$$E(\boldsymbol{y}^{(n)}, \boldsymbol{x}^{(n)}; \boldsymbol{w}, \boldsymbol{\theta}) \leq E(\boldsymbol{y}, \boldsymbol{x}^{(n)}; \boldsymbol{w}, \boldsymbol{\theta}) \quad \forall \boldsymbol{y} \neq \boldsymbol{y}^{(n)}, \tag{6}$$

i.e. we want to learn a set of weights that assign the ground truth labeling an equal or lower energy than any other labeling. Since this problem might have multiple or no solutions we introduce a margin ζ and try to maximize it according to

$$\begin{aligned} &\max_{\boldsymbol{w}:|\boldsymbol{w}|=1} \quad \zeta \\ s.t. \quad &E(\boldsymbol{y}, \boldsymbol{x}^{(n)}; \boldsymbol{w}, \boldsymbol{\theta}) - E(\boldsymbol{y}^{(n)}, \boldsymbol{x}^{(n)}; \boldsymbol{w}, \boldsymbol{\theta}) \geq \zeta \quad \forall \boldsymbol{y} \neq \boldsymbol{y}^{(n)}. \end{aligned} \tag{7}$$

Finding the set of parameters that provides the largest margin regularizes the problem and tends to give good generalization to unseen data. However, for the final objective we make a few changes suggested by Szummer *et al.* [13]. To start of, a slack variable for each training sample ξ_n is introduced to make the method more robust to noisy data. In addition, we use a rescaled margin, demanding a larger energy margin for labelings that differ a lot from the ground truth. Also, the program described in (7) includes an exponential amount of constraints which makes solving it intractable, we therefore perform the optimization over a much smaller set $S^{(n)}$. These changes, given the variable transformation $||\boldsymbol{w}|| \leftarrow 1/\zeta$, give rise to the following problem

$$\begin{aligned} \gamma = \min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{N}\sum_n \xi_n \quad s.t. \quad \forall \boldsymbol{y} \in S^{(n)} \;\; \forall n \\ E(\boldsymbol{y}, \boldsymbol{x}^{(n)}; \boldsymbol{w}, \boldsymbol{\theta}) - E(\boldsymbol{y}^{(n)}, \boldsymbol{x}^{(n)}; \boldsymbol{w}, \boldsymbol{\theta}) \geq \Delta(\boldsymbol{y}^{(n)}, \boldsymbol{y}) - \xi_n \\ \xi_n \geq 0, w_2 \geq 0, w_3 \geq 0, \end{aligned} \tag{8}$$

where N is the number of training samples, C is a hyperparameter regulating the slack penalty and $\Delta(\boldsymbol{y}^{(n)}, \boldsymbol{y})$ is the Hamming loss $\Delta(\boldsymbol{y}^{(n)}, \boldsymbol{y}) = \sum_i \delta(y_i^{(n)}, y_i)$.

The constraint set $S^{(n)}$ is iteratively grown by adding labelings that violate the constraints in (8) the most. For each iteration, the weights are then updated to satisfy the new, larger constraint set. This weight update is repeated until the weights no longer change. The complete learning algorithm is summarized in Algorithm 2.

```
Input: image-labeling pairs ${(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})}$ in the training set
initialize $S^{(n)} = \emptyset$ for each training instance $n$ and $\boldsymbol{w} = \boldsymbol{w}_0$
while $\boldsymbol{w}$ not converged do
    for all training instances $n$ do
        find MAP labeling of instance $n$: $\boldsymbol{y}^* \leftarrow \arg\min_{\boldsymbol{y}} E(\boldsymbol{y}, \boldsymbol{x}^{(n)}; \boldsymbol{w}, \boldsymbol{\theta}) - \Delta(\boldsymbol{y}^{(n)}, \boldsymbol{y})$
        if $\boldsymbol{y}^* \neq \boldsymbol{y}^{(n)}$ then
            add $\boldsymbol{y}^*$ to constraint set: $S^{(n)} \leftarrow S^{(n)} \cup \{\boldsymbol{y}^*\}$
        end
        update $\boldsymbol{w}$ to ensure ground truth has the lowest energy by solving program (8)
    end
end
Output: $\boldsymbol{w}$
```

Algorithm 2: Pseudocode for the CRF weight learning algorithm from [13].

2.3 Back-propagation of Error Derivatives

In this section, we show how the max-margin objective from the previous section can be optimized for our coupled CNN and CRF model. Our main goal during learning is to maximize the margin, or equivalently, minimize the objective γ as defined in (8). To be able to perform a gradient based weight update we need to calculate the derivative of this objective with respect to the weights of the network

$$\frac{\partial \gamma}{\partial \theta_j} = \sum_n \sum_i \frac{\partial \gamma}{\partial \Phi_i} \frac{\partial \Phi_i}{\partial \theta_j}, \tag{9}$$

where the two sums are over the training instances, n, and the pixels, i. As previously, Φ_i is the output of the network. Given a well-defined network structure the term $\frac{\partial \Phi_i}{\partial \theta_j}$ can be easily calculated using standard back-propagation. Henceforth we will focus on calculating the term $\frac{\partial \gamma}{\partial \Phi_i}$. To simplify notation we will introduce z_i as the output of the network of pixel i being foreground, $z_i = \Phi_i(y_i = 1, \boldsymbol{x}; \boldsymbol{\theta})$. We start of by expressing (8) on the following compact form

$$\begin{aligned} \gamma(\boldsymbol{z}) &= \min_{\boldsymbol{w}, \boldsymbol{\xi}} f(\boldsymbol{w}, \boldsymbol{\xi}), \\ \text{s.t.} \quad h_k(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{z}) &\leq 0, \quad k = 1, \ldots, M, \end{aligned} \tag{10}$$

where f is the objective function, h_k characterize the constraints and M is the total number of constraints. We will treat γ as a function depending on the network output, $\gamma(\boldsymbol{z})$.

In addition, the minimizers $\boldsymbol{w}^*$ and $\boldsymbol{\xi}^*$ can also be seen as functions of $\boldsymbol{z}$, that is, $\boldsymbol{w}^* = \boldsymbol{w}^*(\boldsymbol{z})$ and $\boldsymbol{\xi}^* = \boldsymbol{\xi}^*(\boldsymbol{z})$, which gives that

$$\begin{aligned} \gamma(\boldsymbol{z}) &= f(\boldsymbol{w}^*(\boldsymbol{z}), \boldsymbol{\xi}^*(\boldsymbol{z})) = \frac{1}{2}\|\boldsymbol{w}^*\|^2 + \frac{C}{N}\sum_{n=1}^{N} \xi_n^*, \\ \frac{\partial \gamma}{\partial z_i} &= \sum_{j=1}^{D} f_{w_j} \frac{\partial w_j}{\partial z_i} + \sum_{n=1}^{N} f_{\xi_n} \frac{\partial \xi_n}{\partial z_i} = \sum_{j=1}^{D} w_j \frac{\partial w_j}{\partial z_i} + \frac{C}{N}\sum_{n=1}^{N} \frac{\partial \xi_n}{\partial z_i}, \end{aligned} \tag{11}$$

where D is the number of weights and N is the number of slack variables. To be able to calculate $\frac{\partial \gamma}{\partial z_i}$ we need $\frac{\partial w_j}{\partial z_i}$ and $\frac{\partial \xi_n}{\partial z_i}$. These derivatives are found by creating and solving a system of equations from the optimality conditions of the problem. The Lagrangian for the constrained minimization problem in (10) is

$$L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\lambda}) = f(\boldsymbol{w}, \boldsymbol{\xi}) + \sum_{k=1}^{M} \lambda_k h_k(\boldsymbol{w}, \boldsymbol{\xi}),$$

where $\boldsymbol{\lambda}$ is the vector of Langrangian multipliers with elements λ_k. At optimum, the first-order optimality conditions are satisfied:

$$\nabla_{\boldsymbol{w}} L = \boldsymbol{w} + \sum_{k=1}^{M} \lambda_k \nabla_{\boldsymbol{w}} h_k = \boldsymbol{0} \text{ and } \nabla_{\boldsymbol{\xi}} L = \frac{C}{N} + \sum_{k=1}^{M} \lambda_k \nabla_{\boldsymbol{\xi}} h_k = \boldsymbol{0}. \tag{12}$$

Now, the conditions for the implicit function theorem are satisfied and we also get that

$$\frac{\partial(\nabla_{\boldsymbol{w}} L)}{\partial z_i} = \frac{\partial \boldsymbol{w}}{\partial z_i} + \sum_{k=1}^{M} \left(\frac{\partial \lambda_k}{\partial z_i} \nabla_{\boldsymbol{w}} h_k + \lambda_k \frac{\partial \nabla_{\boldsymbol{w}} h_k}{\partial z_i} \right) = \boldsymbol{0}, \tag{13}$$

$$\frac{\partial(\nabla_{\boldsymbol{\xi}} L)}{\partial z_i} = \sum_{k=1}^{M} \left(\frac{\partial \lambda_k}{\partial z_i} \nabla_{\boldsymbol{\xi}} h_k + \lambda_k \frac{\partial \nabla_{\boldsymbol{\xi}} h_k}{\partial z_i} \right) = \boldsymbol{0}. \tag{14}$$

Note that λ_k is a function of $\boldsymbol{z}$. For the active constraints, where $h_k = 0$, it holds that $\frac{\partial h_k}{\partial z_i} = 0$. For the passive constraints, $h_k < 0$, we use the following identities:

$$\lambda_k = 0 \qquad \text{and} \qquad \frac{\lambda h_k}{\partial z_i} = 0. \tag{15}$$

The equations in (12) to (15) give a linear system of equations with the unknowns $\frac{\partial w_j}{\partial z_i}$, $\frac{\partial \xi_n}{\partial z_i}$, λ_k and $\frac{\partial \lambda_k}{\partial z_i}$. Solving this enables us to calculate $\frac{\partial \gamma}{\partial z_i}$ from (11) and finally $\frac{\partial \gamma}{\partial \theta_j}$ according to (9). Having this derivative makes it possible to learn CNN weights that optimize the max-margin objective formulated in (8) using gradient based methods. For more details, see the supplementary material.

2.4 End-to-End Training in Batches

We have now derived all the theoretical tools needed to train our deep structured model in an end-to-end manner. The joint training is done in epochs, where all training samples are utilized in each epoch. In every training epoch, new CRF weights are computed and the CNN weights are updated using gradient descent: $\theta_j \leftarrow \theta_j + \eta \frac{\partial \gamma}{\partial \theta_j}$ for all j.

To facilitate the process of learning deep image features for the CNN we first pretrain the weights $\boldsymbol{\theta}$ on the dataset without the CRF part of the model. Note that the CNN we used is based on a network pretrained on the ImageNet dataset [22]. The pretraining is done using stochastic gradient descent with a standard pixelwise log-likelihood error function.

The original learning method involves the entire training set when computing the CRF weights. However, since the linear equation system that needs to be solved grows with the number of training instances the learning process quickly becomes impractical with an increasing number of images. Hence we propose a method to compute the derivatives in batches. In batch mode we apply the CRF learning method from Algorithm 2 for each batch separately, We also calculate $\frac{\partial \gamma_b}{\partial \theta_j}$ following the steps described in Section 2.3. Note that the objective γ_b that we actually minimize here is an approximation of the true objective since not all images are included. For each batch, the constraint set $S_b^{(n)}$ is saved. These are, at the end of the epoch, merged to a set $S^{(n)}$ containing the low-energy labelings for all training instances. Finally the optimization problem in (8) is solved with this $S^{(n)}$ to get the CRF weights. When solving for the CRF weights we also get the current value of our objective γ, which obviously should decrease during training. The algorithm is summarized in Algorithm 3.

3 Experiments and Results

Now, we present the performance of our method on three different segmentation tasks including comparisons to two baselines. For the first baseline, “CNN (only)“, the segmentation is created by thresholding the output of a pretrained CNN. For the second baseline, “CNN + CRF (piecewise)“, a CNN coupled with a CRF is trained in a piecewise manner, meaning that the network weights are kept fixed while learning the CRF weights. The results for the joint learning is denoted ”CNN + CRF (joint)“. For all experiments the CNN had the same structure as the FCN-8 network introduced by Long *et al.* [4]. The parameter settings were the same for all three segmentation tasks (learning rate = 10^{-4}, batch size = 10 and $C = 1$). All routines for training and testing were implemented in Matlab on top of MatConvNet [23].

```
Input: image-labeling pairs $\{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}$ in the training set.
initialize $\boldsymbol{w} = \boldsymbol{w}_0$ and $\boldsymbol{\theta} = \boldsymbol{\theta}_0$
for number of epochs do
    initialize $S^{(n)} = \emptyset$ for each training instance $n$
    for each batch $b$ do
        CNN forward pass → $\boldsymbol{z}$
        CRF learning by Algorithm 2
        add low-energy labelings to set $S^{(n)}$
        calculation of objective derivative → $\frac{\partial \gamma_b}{\partial z_i}$, back-propagation → $\frac{\partial \gamma_b}{\partial \theta_j}$
        update CNN weights, $\theta_j \leftarrow \theta_j + \eta \frac{\partial \gamma_b}{\partial \theta_j}$
    end
    update CRF weights by solving (8) → $\boldsymbol{w}, \gamma$
end
Output: $\boldsymbol{w}$, $\boldsymbol{\theta}$
```

Algorithm 3: Pseudocode for joint learning of parameters in batches.

3.1 Weizmann Horse Dataset

The Weizmann Horse dataset [14] is widely used for benchmarking object segmentation algorithms. The dataset contains 328 images of horses in different environments, we divide these images into a training set of 150 images, a validation set of 50 images and a test set of 128 images.

Our algorithm is compared to the, to our knowledge, best previously published results on the data set; Reseg [24], CRF-Grad [9] and PatchCut [25]. There are a few variations of the Weizmann Horse dataset available, we used the same one as in PatchCut [25]. Our algorithm is also compared to the two baselines, ”CNN (only)” and ”CNN + CRF (piecewise”). Quantative results (mean Jaccard index) are shown in Table 1 for the test images. In Fig. 1 some qualitative results are presented.

Table 1: *Mean Jaccard index for the Weizmann Horse dataset (test set).*

Method	Jaccard (%)	Method	Jaccard (%)
PatchCut [25]	84.03	CNN (only)	79.97
ReSeg [24]	**91.60**	CNN + CRF (piecewise)	81.62
CRF-Grad [9]	83.98	CNN + CRF (joint)	84.54

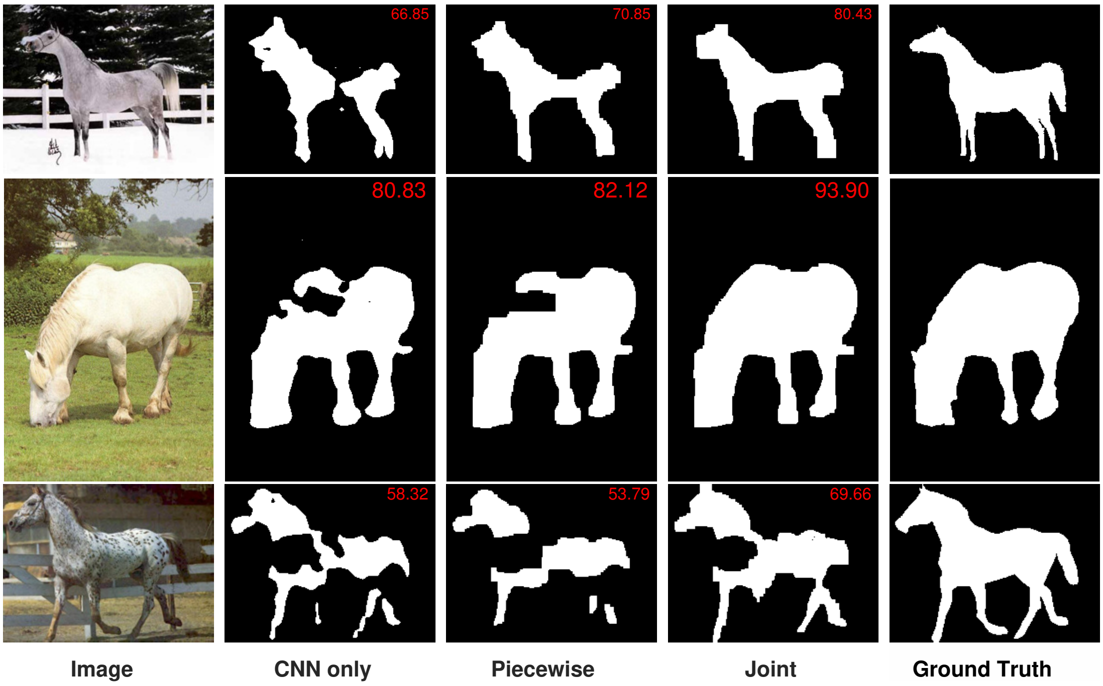

Figure 1: *Qualitative results on the Weizmann Horse dataset. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The number shown in the upper right corner is the Jaccard index (%).*

3.2 Cardiac Ultrasound Dataset

The second dataset we consider consists of 2D cardiac ultrasound images (2-chamber view, *i.e.* the left artrium and the left ventricle are visible). The ground truth consists of manual annotations of the left ventricle made by an experienced cardiologist according to the protocol in [26]. The dataset contains 66 images which are divided into a training set of 33 images, a validation set of 17 images and a test set of 16 images. See Fig. 2 and Table 2 for qualitative and quantitative results respectively.

3.3 Cardiac CTA Dataset

The third dataset we consider consists of 2D slices of cardiac CTA volumes originating from the SCAPIS pilot study [27]. The ground truth consists of slice-wise manual annotations of the pericardium made by a specialist in thoracic radiology and according to the protocol in [28]. The dataset includes in total 1500 2D slices which are divided into three subsets of equal size to be evaluated separately representing three different views (*i.e.* axial, coronal and sagittal view). For each view the 2D slices were divided into a training set of 300 images, a validation set of 100

Table 2: *Quantitative results for the Cardiac Ultrasound dataset (US) and the Cardiac CTA dataset (CTA). For the CTA dataset, the different types of slices are evaluated separately (ax - axial, cor - coronal and sag - sagittal). The mean Jaccard index (%) for the test sets are reported.*

Method	US	CTA-ax	CTA-cor	CTA-sag
CNN (only)	82.28	81.40	77.11	75.96
CNN + CRF (piecewise)	85.79	81.84	77.12	75.83
CNN + CRF (joint)	**86.20**	**82.10**	**77.71**	**76.34**

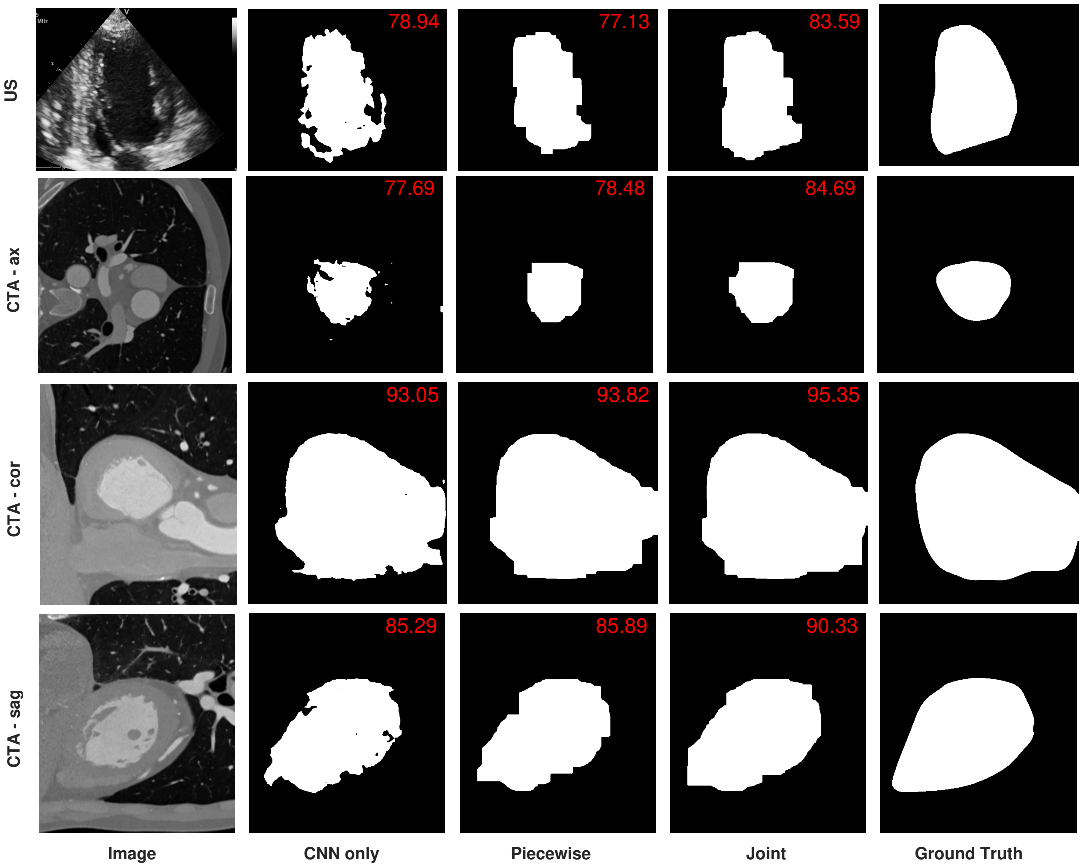

Figure 2: *Qualitative results on the Cardiac ultrasound dataset (US) and the Cardiac CTA dataset (CTA). For the CTA dataset, the different types of slices are evaluated separately (ax - axial, cor - coronal and sag - sagittal). "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%).*

images and a test set of 100 images. Some of the 2D slices originate from regions where the pericardium is not visible. Thus, these images were excluded from the quantitative results since the Jaccard index is undefined if the ground truth and segmentation are both empty sets. Some qualitative results of the joint training process are visualized in Fig. 2 and quantitative results are presented in Table 2.

4 Conclusion and Future Work

In this paper, we have proposed a segmentation algorithm based on a deep structured model consisting of a CNN paired with a CRF. We also presented a method for jointly learning the parameters of the CNN and the CRF using a max-margin approach. Conveniently, the max-margin objective could be optimized with standard back-propagation thanks to the theoretical results derived in Section 2.3.

We achieve superior results on two smaller medical datasets when comparing to using a CNN only and using a CNN paired with a CRF trained separately. Note that the CNN we used is based on a network pretrained on the ImageNet dataset [22]. It has hence learnt image features for standard RGB images and for classification tasks, which of course makes it more challenging learning CNN weights well-adjusted for medical image segmentation. In spite of this, we still achieve good results on the two medical datasets. A future continuation of this work would be to combine the CRF with a CNN trained on a larger set of medical images. Also, implementing the framework for 3D would increase its usability when it comes to medical applications.

In addition, other types of CRFs could be used. The ones considered in this paper only include pairwise terms depending on neighbouring pixels. One possible extension would be to consider longer distance relationships or higher order energy terms. Also, the pairwise terms could be learned with a trainable CNN in the same way as for unary terms. A trainable regularization term would surely enable the model to learn even more sophisticated relationships for the output pixels.

We gratefully acknowledge funding from SSF (Semantic Mapping and Visual Navigation for Smart Robots) and VR (project no. 2016-04445).

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Adv. Neural. Inf. Process. Syst.*, 2012.

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

[3] A. Giusti, D. Cireşan, J. Masci, L. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *Int. Conf. on Image Processing*, 2013.

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[5] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Int. Conf. on Computer Vision*, 2015.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *Int. Conf. on Learning Representations*, 2015.

[7] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with gaussian edge potentials," in *Adv. Neural. Inf. Process. Syst.*, 2011.

[8] L.-C. Chen, A. Schwing, A. Yuille, and R. Urtasun, "Learning deep structured models," in *Int. Conf. on Machine Learning*, 2015.

[9] M. Larsson, A. Arnab, F. Kahl, S. Zheng, and P. H. S. Torr, "A projected gradient descent method for crf inference allowing end-to-end training of arbitrary pairwise potentials," in *11th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, (EMMCVPR).* Springer, 2017.

[10] G. Lin, C. Shen, A. van den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Conf. on Computer Vision and Pattern Recognition*, 2016.

[11] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Adv. Neural. Inf. Process. Syst.*, 2014.

[12] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional random fields as recurrent neural networks," in *Int. Conf. on Computer Vision*, 2015.

[13] M. Szummer, P. Kohli, and D. Hoiem, "Learning CRFs using graph cuts," in *European Conf. on Computer Vision*, 2008.

[14] E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," in *European Conf. on Computer Vision*, 2002.

[15] M. Ranzato, P. Taylor, J. House, R. Flagan, Y. LeCun, and P. Perona, "Automatic recognition of biological particles in microscopic images," *Pattern Recognition Letters*, vol. 28, no. 1, pp. 31–39, 2007.

[16] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *MICCAI*, vol. 2, 2013, pp. 411–418.

[17] A. Prasoon, K. Petersen, C. Igel, F. Lauze, E. Dam, and M. Nielsen, "Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network," in *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, 2013.

[18] H. Roth, L. Lu, A. Farag, H. Shin, J. Liu, E. Turkbey, and R. Summers, "Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, ser. Lecture Notes in Computer Science, 2015, vol. 9349, pp. 556–564.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9351. Springer, 2015, pp. 234–241, (available on arXiv:1505.04597 [cs.CV]). [Online]. Available: http://lmb.informatik.uni-freiburg.de//Publications/2015/RFB15a

[20] T. Brosch, L. Y. W. Tang, Y. Yoo, D. K. B. Li, A. Traboulsee, and R. Tam, "Deep 3d convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1229–1239, 2016.

[21] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 147–159, 2004.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Conf. on Computer Vision and Pattern Recognition*, 2009.

[23] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Int. Conf. on Multimedia*, 2015.

[24] 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 *Conf. on Computer Vision and Pattern Recognition Workshops*, 2016.

[25] J. Yang, B. Price, S. Cohen, Z. Lin, and M.-H. Yang, "PatchCut: Data-driven object segmentation via local shape transfer," in *Conf. on Computer Vision and Pattern Recognition*, 2015.

[26] R. Lang *et al.*, "Recommendations for cardiac chamber quantification by echocardiography in adults: an update from the american society of echocardiography and the european association of cardiovascular imaging," *J. Am. Soc. Echocardiogr.*, vol. 28, no. 1, pp. 1–39, 2015.

[27] G. Bergström *et al.*, "The Swedish CArdioPulmonary bioImage Study: Objectives and design," *J. of Internal Medicine*, vol. 278, no. 6, pp. 645–659, 2015.

[28] A. Norlén, J. Alvén, D. Molnar, O. Enqvist, R. Norrlund, J. Brandberg, G. Bergström, and F. Kahl, "Automatic pericardium segmentation and quantification of epicardial fat from computed tomography angiography," *J. of Medical Imaging*, vol. 3, no. 3, 2016.

Supplementary Material

Constraint Derivatives

In this section we derive the derivatives of the constraints in (8) in the main paper. This is needed to calculate $\frac{\partial \gamma}{\partial z_i}$, i.e. the derivative of the objective with respect to the CNN output. We start off by rewriting all constraints on the form $h_k \leq 0$,

$$\begin{aligned} E(\boldsymbol{y}^{(n)}) - E(\boldsymbol{y}) + \Delta(\boldsymbol{y}^{(n)}, \boldsymbol{y}) - \xi_n &\leq 0 && (16)\\ -\xi_n &\leq 0 && (17)\\ -w_2, -w_3 &\leq 0. && (18)\end{aligned}$$

Note that, to clarify notation, we have not included all dependencies for the energies here. We divide the constraints into three categories; energy constraints (16), slack constraints (17) and weight constraints (18). As mentioned in Section 2.3, for the constraints that are active ($h_i = 0$) we add the equation $\frac{\partial h_i}{\partial z_j} = 0$ to our linear system. Hence we need to derive $\frac{\partial h_i}{\partial z_j}$ for the different types of constraints. In addition, in (12)-(15) we also need $\frac{\partial h_i}{\partial w_1}$, $\frac{\partial h_i}{\partial w_2}$, $\frac{\partial h_i}{\partial w_3}$ and $\frac{\partial^2 h_i}{\partial w_2 \partial z_j}$. Note that all other second order derivatives equal zero.

Energy Constraints.

The energy constraints are formulated as follows

$$\begin{aligned} & h_i(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) = w_1 \sum_{j \in V} (-\log(\Phi_j(y_j^{(n)}, \boldsymbol{x}; \boldsymbol{\theta})) + \log(\Phi_j(y_j, \boldsymbol{x}; \boldsymbol{\theta}))) + \\ & w_2 \sum_{(j,k)\in\varepsilon} (1_{y_j^{(n)} \neq y_k^{(n)}} - 1_{y_j \neq y_k}) + w_3 \sum_{(j,k)\in\varepsilon} (1_{y_j^{(n)} \neq y_k^{(n)}} - 1_{y_j \neq y_k}) e^{\left(-\frac{(x_j - x_k)^2}{2}\right)} + \qquad (19) \\ & + \Delta(y^{(n)}, y) - \xi_n = w_1 \sum_{j \in V} U_i(z_j) + C_2^{(i)} w_2 + C_3^{(i)} w_3 + \Delta(y^{(n)}, y) - \xi_n, \end{aligned}$$

where the index n denotes the ground truth instance that the energy is coupled with. Note that there are several of these constraints per training instance, one for each $\boldsymbol{y} \in S^{(n)}$. To simplify notation we have introduced $C_2^{(i)}$ and $C_3^{(i)}$ which are constants given a low energy labeling $\boldsymbol{y} \in S^{(n)}$ and a ground truth labeling $\boldsymbol{y}^{(n)}$. In addition we have introduced $U_i(z_j)$ which equals

$$U_i(z_j) = \begin{cases} 0 & \text{if} \quad y_j = y_j^{(n)} \\ \log(z_j) - \log(1 - z_j) & \text{if} \quad y_j = 1, y_j^{(n)} = 0 \\ \log(1 - z_j) - \log(z_j) & \text{if} \quad y_j = 0, y_j^{(n)} = 1 \end{cases} \qquad (20)$$

with the derivative

$$\frac{\partial U(z_j)}{\partial z_j} = \begin{cases} 0 & \text{if} \quad y_j = y_j^{(n)} \\ \frac{1}{z_j} + \frac{1}{1-z_j} & \text{if} \quad y_j = 1, y_j^{(n)} = 0 \\ -\frac{1}{z_j} - \frac{1}{1-z_j} & \text{if} \quad y_j = 0, y_j^{(n)} = 1 \end{cases}$$

We now calculate the needed derivatives, excluding the ones equal to zero

$$\begin{aligned} \frac{\partial h_i}{\partial z_j} &= \frac{\partial w_1}{\partial z_j} \sum_{k \in V} U_i(z_j) + \frac{\partial U_i(z_j)}{\partial z_j} w_1 \\ &\quad + C_2^{(i)} \frac{\partial w_2}{\partial z_j} + C_3^{(i)} \frac{\partial w_3}{\partial z_j} - \frac{\partial \xi_n}{\partial z_j} \\ \frac{\partial h_i}{\partial w_1} &= \sum_{j \in V} U_i(z_j) \\ \frac{\partial h_i}{\partial w_2} &= C_2^{(i)} \\ \frac{\partial h_i}{\partial w_3} &= C_3^{(i)} \\ \frac{\partial h_i}{\partial \xi_n} &= \begin{cases} -1 & \text{if} \quad \boldsymbol{y}^i \in S^{(n)} \\ 0 & \text{else} \end{cases} \\ \frac{\partial^2 h_i}{\partial w_1 \partial z_j} &= \frac{\partial U_i(z_j)}{\partial z_j}. \end{aligned} \tag{21}$$

Here $\boldsymbol{y}^i \in S^{(n)}$ means that that the energy constraint h_i is related to image and labeling $\{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}$. Note that the term $U_i(z_j)$ is the only explicit dependence on the network output in our optimization problem. Looking at the definition of $U_i(z_j)$ in (20) we see that it will be zero for a lot of pixels. The derivative for all those pixel will be the same and can hence be calculated by solving one linear system. However, for the pixels where $U_i(z_j) \neq 0$, the derivative needs to be calculated for each pixel.

Slack and Weight Constraints.

The slack constraints are formulated as follows

$$h_i = -\xi_i.$$

Calculating the needed derivatives, excluding the ones equal to zero gives

$$\begin{aligned} \frac{\partial h_i}{\partial \xi_i} &= -1 \\ \frac{\partial h_i}{\partial z_j} &= -\frac{\partial \xi_i}{\partial z_j}. \end{aligned}$$

Note that the weight constraints are identical (switch ξ_i for w_2 or w_3).

Final Explicit Linear System

The variables for the linear system of equations are as previously mentioned $\frac{\partial w_i}{\partial z}$, $\frac{\partial \xi_i}{\partial z}$, λ_i, $\frac{\partial \lambda_i}{\partial z}$. To compute these quantities, we set up and solve a linear system $\boldsymbol{Ax} = \boldsymbol{b}$ where

$$\boldsymbol{x} = \begin{bmatrix} \frac{\partial w_1}{\partial z} \\ \vdots \\ \frac{\partial w_D}{\partial z} \\ \frac{\partial \xi_1}{\partial z} \\ \vdots \\ \frac{\partial \xi_n}{\partial z} \\ \lambda_1 \\ \vdots \\ \lambda_N \\ \frac{\partial \lambda_1}{\partial z} \\ \vdots \\ \frac{\partial \lambda_N}{\partial z} \end{bmatrix}. \tag{22}$$

Note that $\boldsymbol{A}$ and $\boldsymbol{b}$ are calculated from the equations satisfied at the optimum. Following the derivations in Section 2.3 and the previous section, we are now ready to explicitly state these equations. Call the set of indices corresponding to the energy constraint I_E and the indices corresponding to the positive slack constraint I_S. For the first part of (12) we get

$$\begin{aligned} \frac{\partial L}{\partial w_1} &= w_1 + \sum_{i \in I_E} \lambda_i \sum_{j \in \mathcal{V}} U_i(z_j) = 0 \\ \frac{\partial L}{\partial w_2} &= w_2 + \sum_{i \in I_E} \lambda_i C_2^{(i)} - \lambda_{w_2} = 0 \\ \frac{\partial L}{\partial w_3} &= w_3 + \sum_{i \in I_E} \lambda_i C_3^{(i)} - \lambda_{w_3} = 0, \end{aligned} \tag{23}$$

where λ_{w_2} and λ_{w_3} are the Lagrangian multipliers corresponding to the weight constraints. For the second part of (12) we get

$$\frac{\partial L}{\partial \xi_n} = \frac{C}{N} - \sum_{k \in I_E^{(n)}} (\lambda_k) - \lambda_{(n)} = 0,$$

where $I_E^{(n)}$ is the set containing the indices for the energy constraints for training instance (n) and $\lambda_{(n)}$ is the lagrangian multiplier corresponding to the slack constraint of instance (n). For (13) we get

$$\begin{aligned}
\frac{\partial^2 L}{\partial w_1 \partial z_j} &= \frac{\partial w_1}{\partial z_j} + \sum_{i \in I_E} \left(\frac{\partial \lambda_i}{\partial z_j} \sum_{k \in \mathcal{V}} (U_i(z_k)) + \lambda_i \frac{\partial U_i(z_j)}{\partial z_j} \right) = 0 \\
\frac{\partial^2 L}{\partial w_2 \partial z_j} &= \frac{\partial w_2}{\partial z_j} + \sum_{i \in I_E} \left(\frac{\partial \lambda_i}{\partial z_j} C_2^{(i)} \right) - \frac{\partial \lambda_{w_2}}{\partial z_j} = 0 \\
\frac{\partial^2 L}{\partial w_3 \partial z_j} &= \frac{\partial w_3}{\partial z_j} + \sum_{i \in I_E} \left(\frac{\partial \lambda_i}{\partial z_j} C_3^{(i)} \right) - \frac{\partial \lambda_{w_3}}{\partial z_j} = 0.
\end{aligned} \tag{24}$$

And finally for (14)

$$\frac{\partial^2 L}{\partial \xi_n \partial z_j} = - \sum_{k \in I_E^{(n)}} \left(\frac{\partial \lambda_k}{\partial z_j} \right) - \frac{\partial \lambda_{(n)}}{\partial z_j} = 0.$$

In addition to these equations we also get one equation for each active constraint, $\frac{\partial h_i}{\partial z_j} = 0$ and two equations for each passive constraint, see (15). All of these equations make up $\boldsymbol{A}$ and $\boldsymbol{b}$ of our linear system. Solving it we get $\boldsymbol{x}$ as defined in (22).

Additional Results

In this section we present some additional results for the different datasets. For the Weizmann Horse dataset results from the joint training process can be seen in Fig. 3 and some qualitative results can be seen in Fig. 4. For the Cardiac Ultrasound dataset qualitative results can be seen in Fig. 5 and for the Cardiac CTA dataset the results on the axial, coronal and sagittal slices can be seen in Fig. 6, Fig. 7 and Fig. 8 respectively.

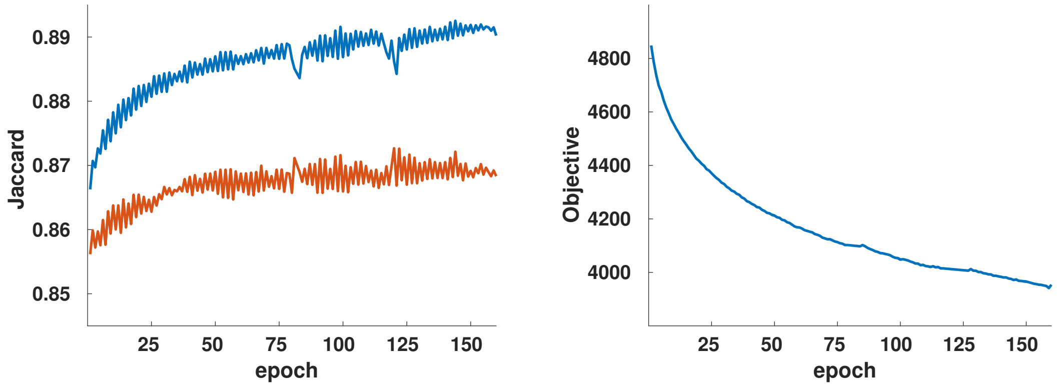

Figure 3: *Joint training results for the Weizmann Horse dataset. The left figure shows the mean Jaccard index versus epochs for the training images (blue upper graph) and the validation images (red lower graph). The right figure shows the max-margin objective, γ, from* (8) *versus epochs.*

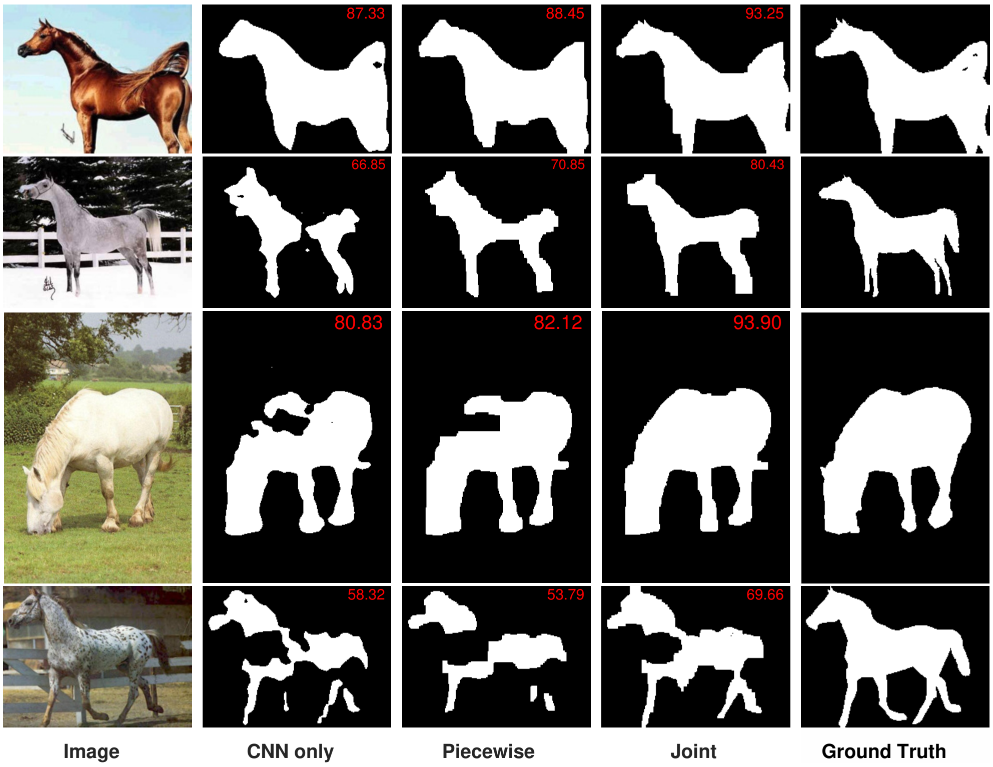

Figure 4: *Qualitative results on the Weizmann Horse dataset. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.*

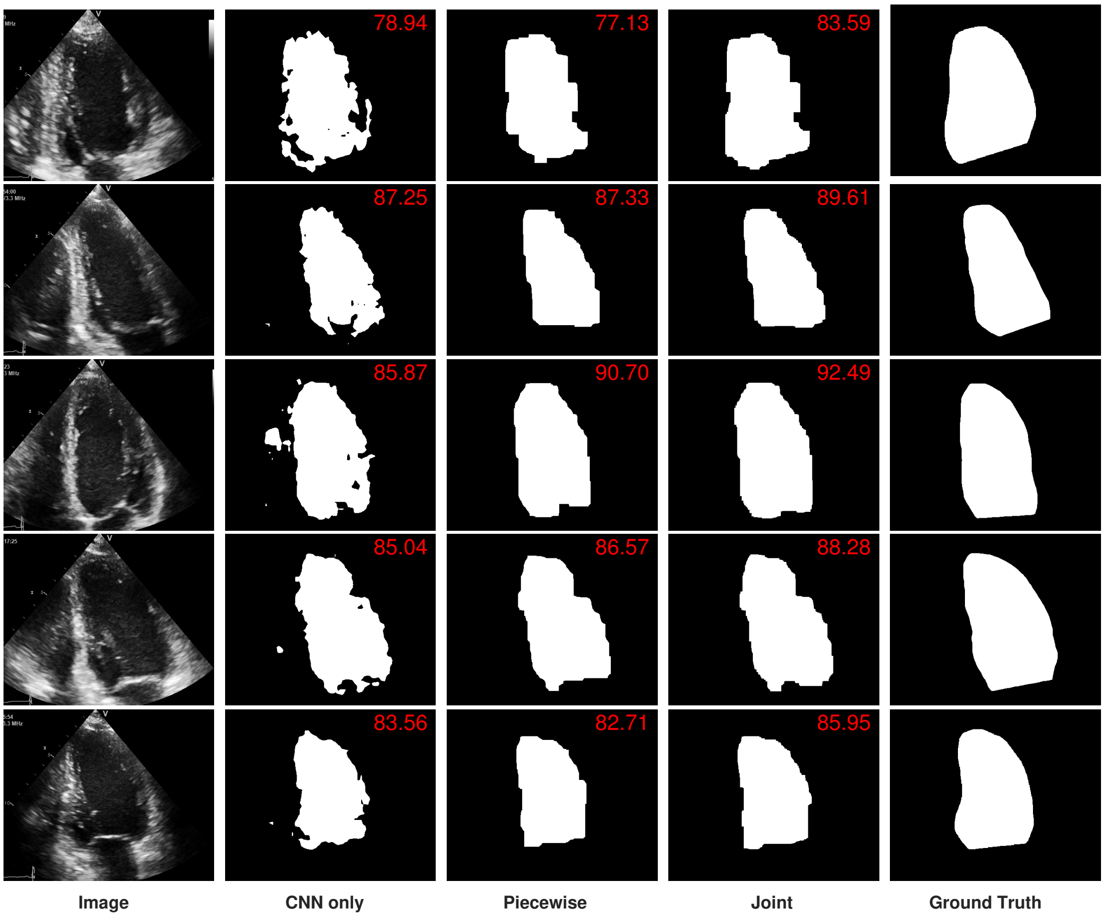

Figure 5: *Qualitative results on the Cardiac ultrasound dataset. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.*

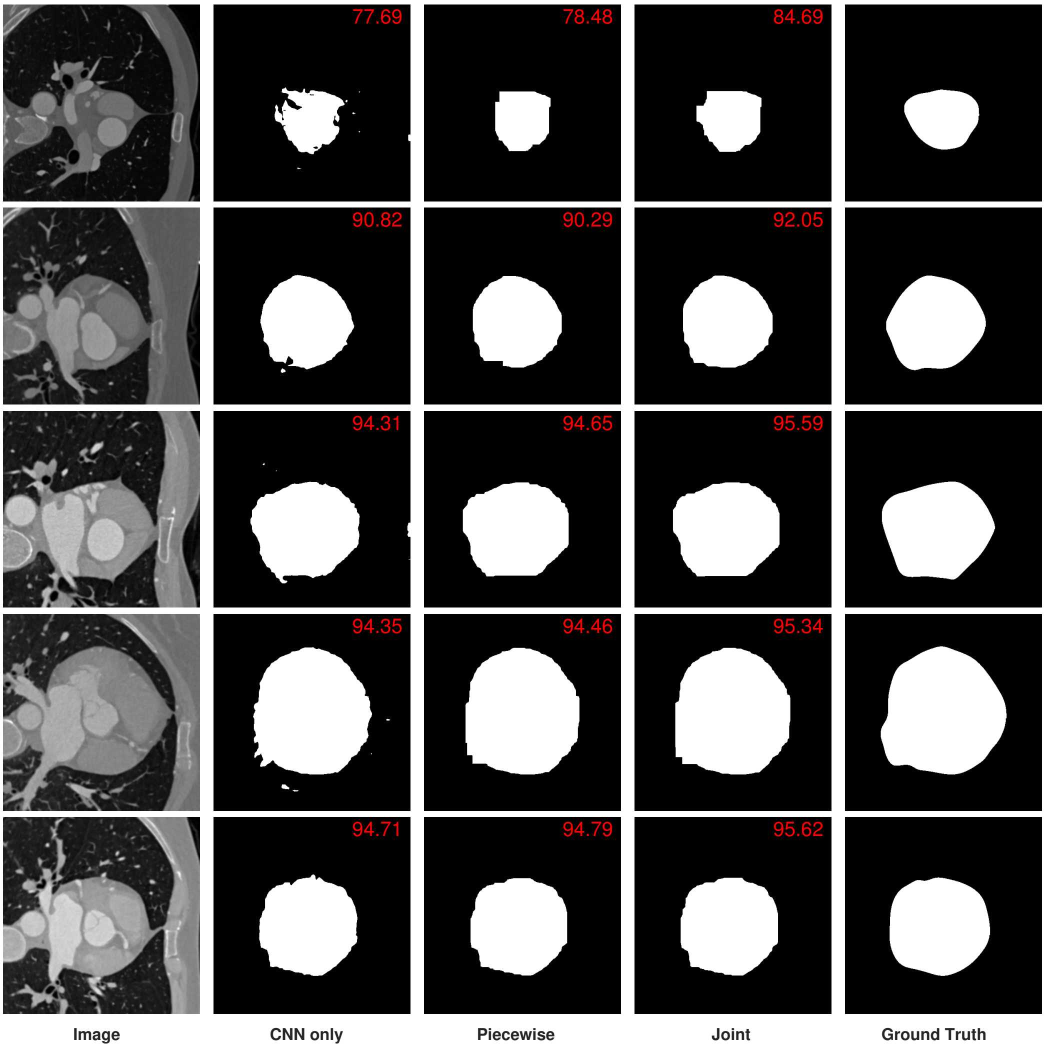

Figure 6: *Qualitative results on the Cardiac CTA dataset for the axial slices. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.*

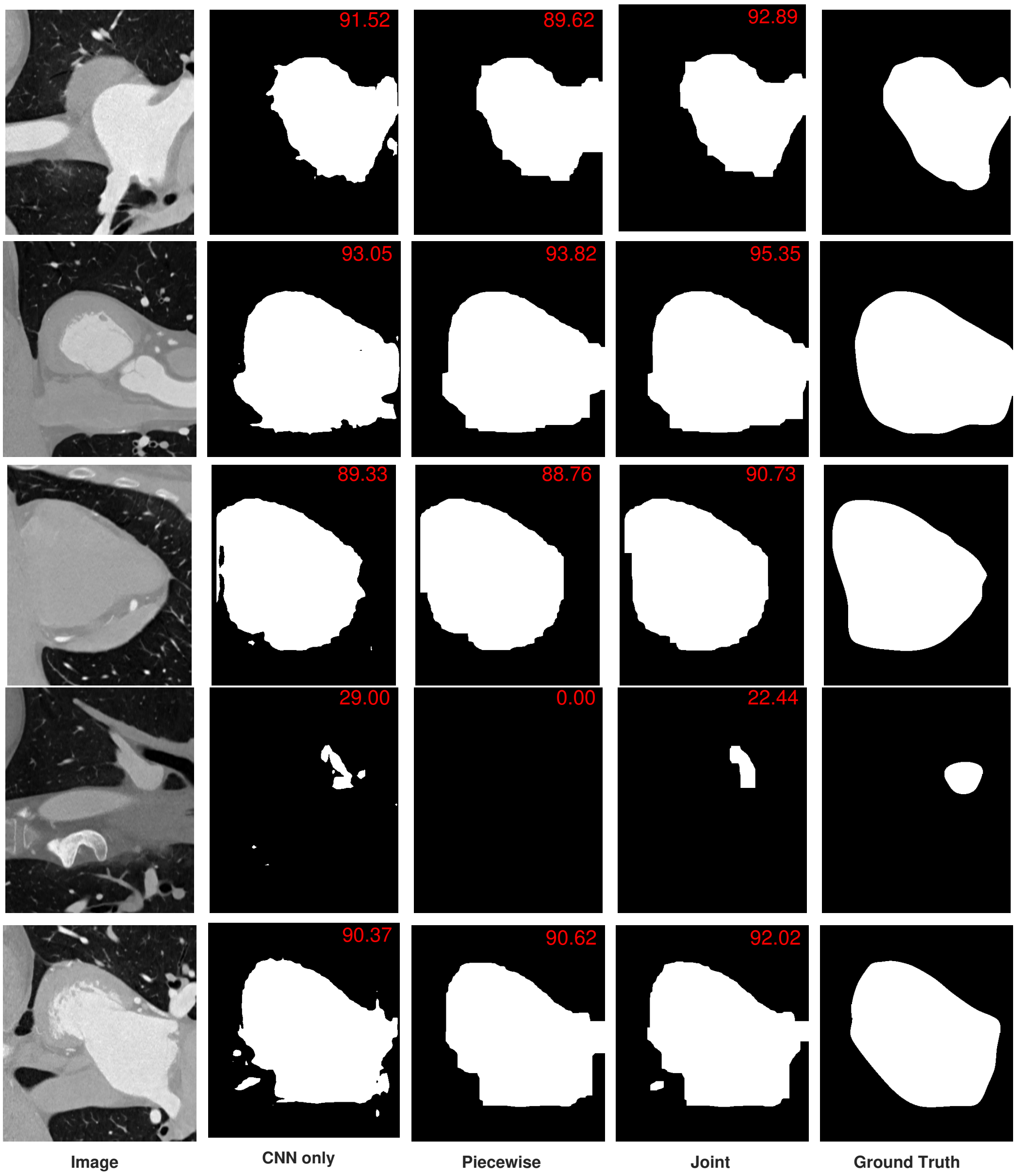

Figure 7: *Qualitative results on the Cardiac CTA dataset for the coronal slices. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.*

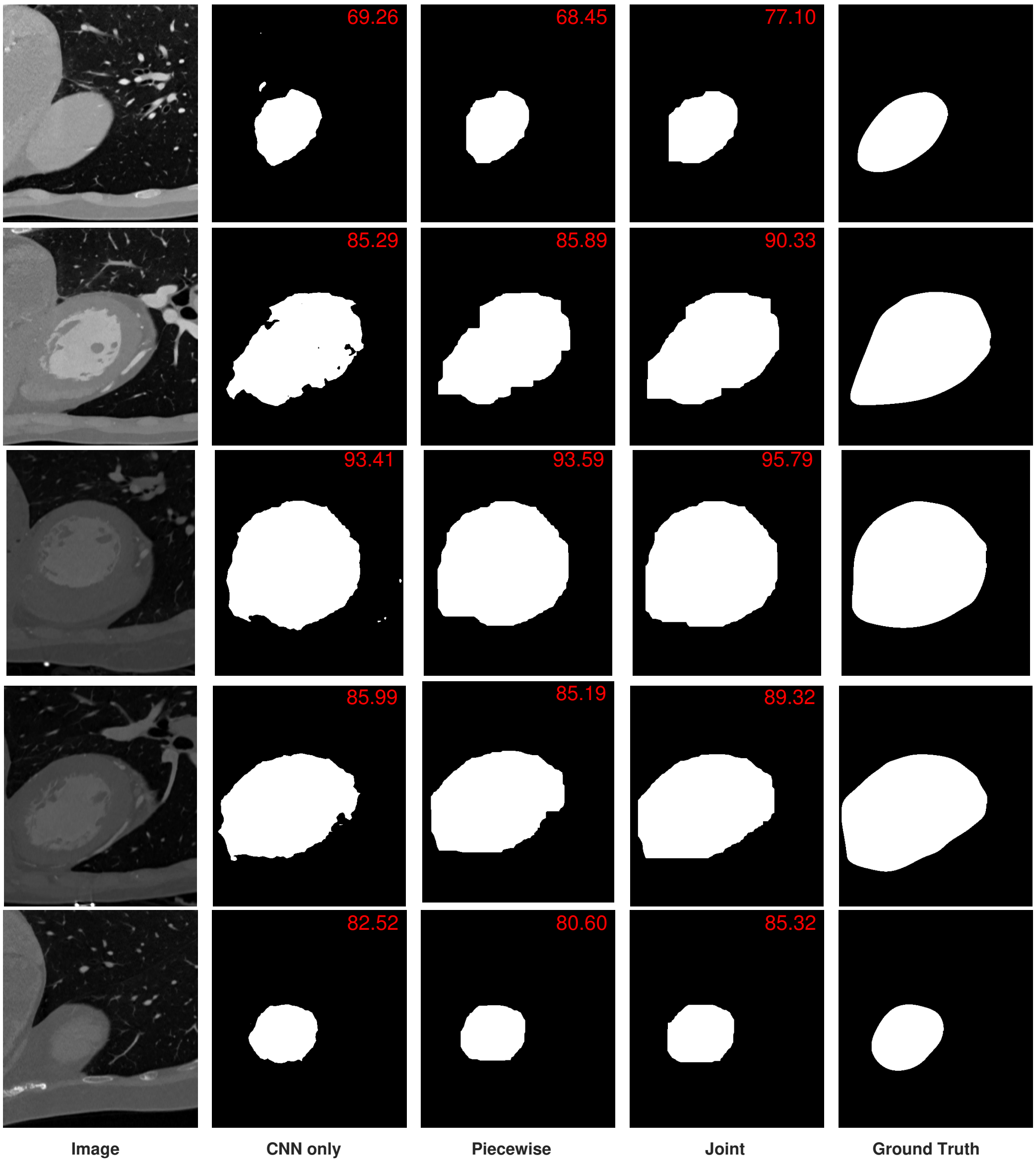

Figure 8: *Qualitative results on the Cardiac CTA dataset for the sagittal slices. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.*

Paper III

Robust Abdominal Organ Segmentation Using Regional Convolutional Neural Networks

M. Larsson, Y. Zhang and F. Kahl

Submitted to Applied Soft Computing

Comment: The layout of this paper has been reformatted in order to comply with the rest of the thesis.

Robust Abdominal Organ Segmentation Using Regional Convolutional Neural Networks

M. Larsson, Y. Zhang and F. Kahl

Abstract

A fully automatic system for abdominal organ segmentation is presented. As a first step, an organ localization is obtained via a robust and efficient feature registration method where the center of the organ is estimated together with a region of interest surrounding the center. Then, a convolutional neural network performing voxelwise classification is applied. Two convolutional neural networks of different architecture are compared. The first one has a structure similar to networks used for classification and is applied using a sliding window approach. The second one has a structure allowing it to be applied in a fully convolutional manner reducing computation time. Despite limited training data, our experimental results are on par with state-of-the-art approaches that have been developed over many years. More specifically the method is applied to the MICCAI2015 challenge “Multi-Atlas Labeling Beyond the Cranial Vault” in the free competition for organ segmentation in the abdomen. The method performed well for both types of convolutional neural networks. For the fully convolutional network a mean Dice coefficient of **0.767** was achieved, for the network applied with sliding window this number was **0.757**.

1 Introduction

Segmentation is a key problem in medical image analysis, and an automated method for organ segmentation can be crucial for numerous applications in medical research and clinical care such as computer aided diagnosis and surgery assistance. The high variability of the shape and position of abdominal organs makes segmentation a challenging task. Previous work done on segmentation of abdominal organs includes, among others, multi-atlas methods [1], patch-based methods [2], and methods based on a probabilistic atlas [3, 4]. These techniques achieve great results for several abdominal organs but struggle with the segmentation of organs where the anatomical variability is large such as the gallbladder.

During the last few years, deep convolutional neural networks have shown great performance and achieved state of the art results in many computer vision applications [5, 6]. This fact can be partly attributed to the constant increase in available computing power, most notably GPU computing solutions, and the availability of large annotated datasets. In the field of medical image analysis this development has led to an increased interest in methods based on deep convolutional neural networks with promising results [7, 8]. Recently, more intricate deep learning approaches have been proposed in the field of image analysis. Ronneberger et al [9] proposed the U-Net, a network based on the idea of "fully convolutional networks" [6]. This work was extended to a network utilizing 3D convolutional filters by Çiçek et al [10]. A similar network structure was also proposed by Brosch et al [11]. Another example is Kamnitsas et al [12] that used an 11-layer deep 3D convolutional network for brain lesion segmentation with good results.

In this paper, instead of designing larger and more complicated network structures we propose a two-step method, simplifying the task the convolutional neural network need to solve. In summary, an automatic system for the segmentation of abdominal organs in contrast enhanced CT images is presented. Our main contribution is to show that despite limited training data (compared to other successful deep learning approaches), it is possible to design a system that achieves on par with state-of-the-art for very challenging segmentation tasks with high anatomical variability. Another contribution is that we develop a computationally efficient framework that allows for a fully integrated 3D approach to multi-organ segmentation. To our knowledge, this is the first attempt on using 3D CNNs for multi-organ abdominal segmentation.

The presented method is trained and tested on the MICCAI2015 challenge "Multi-Atlas Labeling Beyond the Cranial Vault" [13] where it achieved state of the art results in the free competition for organ segmentation in the abdomen.

2 Proposed Solution

Our system segments each organ independently and can be divided into three steps:

1. Localization of region of interest using a multi-atlas approach.
2. Voxelwise binary classification using a convolutional neural network.
3. Postprocessing by thresholding and removing all positive samples except the largest connected component.

Each step will now be described in detail.

2.1 Localization of region of interest

This part provides a robust initialization of the segmentation. The goal is to locate the center voxel of the organ in the target image. When this has been done a prediction mask is placed centered around the predicted organ center. The prediction mask later defines the region of interest where the convolutional neural network is initially applied. The use of an initialization method enables us to train more specialized, or regional, networks that only need to differentiate between a certain organ and the background. This means that the classification task that the network needs to perform is simplified and computationally less demanding networks can be applied.

The location of the organ center in the target image is obtained using a feature-based multi-atlas approach. Each atlas image is registered to the target using the method described in [14]. The method is computationally efficient compared to traditional intensity-based registration methods. More importantly though, this approach provides a robust and reliable estimate for organ locations which have been demonstrated for many different settings and modalities. In our framework, the registration is performed individually for each organ and atlas image. Affine transformations are computed and then used to transform each organ center point from an atlas image to the target image. The median of the transformed center points provides us with an estimate of the center point for the region of interest in the target image. The reason for using the median, and not for instance the mean operator, is that it provides a robust estimate of the center point, that is, it is not affected by a few, spurious outliers.

The prediction mask is estimated using the ground truth segmentations of the atlas images. Let the ground truth segmentations be represented by a binary image of the same dimension as the atlas image $G^{(l)}$, where l is the image id, and $G^{(l)}_{ijk} = 1$ if and only if voxel with index i, j, k in image with id l is foreground (or organ). Further, define $D^{(l)}$ as the binary image formed by dilating $G^{(l)}$ by a cube of size $25 \times 25 \times 25$ voxels and translating it so that the center of the organ is located at the center of the image. The prediction mask P is then defined as the binary image where each element P_{ijk} is

$$P_{ijk} = \begin{cases} 1 & \text{if} \quad \frac{1}{N}\sum_{l=1}^{N} D^{(l)}_{ijk} \geq \delta \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where N is the number of atlas images and δ is a threshold set to $\delta = 0.5$ for the majority of the organs. Finally, the region of interest R is defined as the prediction mask centered around the estimated center point. An example of a localization of region of interest is shown in Figure 1.

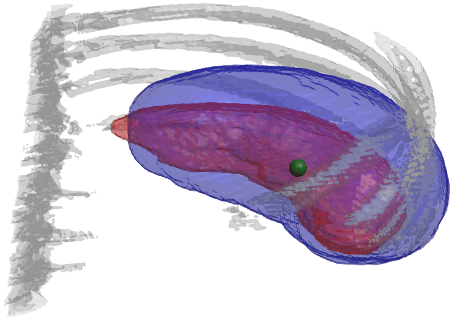

Figure 1: *Example of region of interest localization for the Spleen. The green sphere is the estimated center point, the red mask describes the ground truth and the blue mask describes the estimated region of interest.*

2.2 Voxel classification using a convolutional neural network

Two different types of CNNs are compared for this step. The first one, from now on referred to as CNN-sw, has similar structure as a CNN used for classification and is applied using a sliding window approach. For each voxel to be segmented two cubes of different resolutions centered around said voxel are extracted and used as input features to the network. The network in return outputs a probability, denoted p_{ijk}, of the voxel being organ. The second type of CNN, from now on referred to as FCN, has a structure allowing it to be applied in a fully convolutional manner, hence the probability of several voxels can be calculated in the same forward pass. The specific structure for both networks are presented in Sections 2.2 and 2.2, respectively.

As previously mentioned, the CNN-sw outputs the probability for one voxel being organ each forward pass. To speed up the voxel classification process the network is not applied to every voxel in the area that is being segmented, denoted S. Instead, it takes steps of three in each dimension over S. The probabilities output by the network are then interpolated to every voxel in S. All voxels in S that have been assigned an interpolated probability neither close to zero nor close to one (voxels with a probability between 0.1 and 0.9 to be specific) will be classified by the network once more. The idea behind this approach is that for easily classified regions the network is only applied to a grid of the voxels while for regions where classification might be harder, such as the boundaries of organs, the network classifies every voxel explicitly. For the FCN, this approach is unnecessary since the network can be applied to several voxels in one forward pass, calculating the probabilities for each voxel in a rectangular cuboid.

To reduce the dependency on the quality of the initial region of interest where the convolutional neural network is applied, a region growing algorithm is used. Call the set of voxels that should be segmented S. Further, call the set of voxels already classified by D and the set of voxels with an assigned probability larger than 0.5 by O. The region growing algorithm is then described by Algorithm 4. The use of a region growing algorithm means that even though the initial region may only cover part of the organ, a successful segmentation is still possible, see Figure 5.

```
Initialize:
    • $S$ as the region of interest $R$
    • $D$ as $\varnothing$
    • $O$ as $\varnothing$.
while $S \neq \varnothing$ do
    • Classify voxels in $S$
    • Set $D = D \cup S$, and $O$ as the set of voxels with an assigned
      probability larger than 0.5
    • Let $O^+$ be the set $O$ dilated by a cube of size $12 \times 12 \times 4$ voxels
    • Set $S = O^+ \setminus D$
end
Output: $O$
```

Algorithm 4: Region growing algorithm for efficient classification.

CNN-sw Setup

The CNN-sw performs voxelwise binary classification. The input features for the network are two image cubes, one with a fine resolution similar to the original CT image and the other with a coarse resolution. The fine resolution input feature is meant to provide the network with local information ensuring local precision while the coarse resolution input feature is meant to ensure global spatial consistency. The inputs are processed in separate pipelines and the aggregated features from both pipelines are then merged for the last part of the network. A schematic of the network is shown in Figure 2.

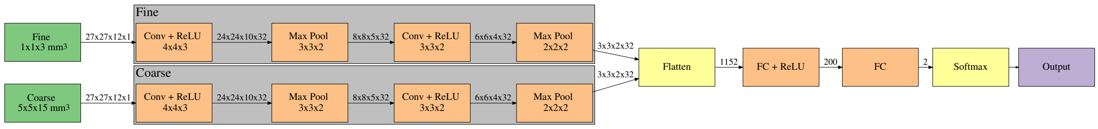

Figure 2: *Structure of the CNN-sw. Both type and kernel size of each layer are shown, in addition the size of the features are shown at each edge. The following abbreviations are used, Conv: Convolutional layer, ReLU: Rectified Linear Unit and FC: Fully Connected. Both inputs are rectangular cuboids centered around the voxel to be classified, the resolutions are shown in the figure. Best viewed in color.*

Implementation and training.

For the implementation of the CNN-sw the framework Torch7 was used [15]. For each network, the training and validation sets were extracted from the region of interest calculated as previously described as well as the area around the organ in the image (which in some cases might not be included in the region of interest). This was done for each image in the training set. For the majority of the organs a balanced training set was used, meaning that there was an equal amount of foreground and background samples in the training set. However, since some of the organs are quite small this leads to a relatively small training set. Several methods, listed below, were used to deal with this problem.

1. For organs present in pairs, kidneys and adrenal glands, training samples from both the left and the right organs were used. Note that this does not pose a problem during inference since the initialization part of the method will separate the organs.

2. Expansion of the training set by adding slightly distorted CT images, transforming them using a random affine transformation similar to the identity transformation. The transformation matrix T was randomized as

$$T = \begin{pmatrix} 1+\delta_{11} & \delta_{12} & \delta_{13} & 0 \\ \delta_{21} & 1+\delta_{22} & \delta_{23} & 0 \\ \delta_{31} & \delta_{32} & 1+\delta_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

   where δ_{ij} are independently and uniformly randomized numbers between -0.25 and 0.25 for $i = 1, 2, 3$ and $j = 1, 2, 3$.

3. Including a greater number of background samples than foreground samples in the training set. This leads to a larger but unbalanced training set.

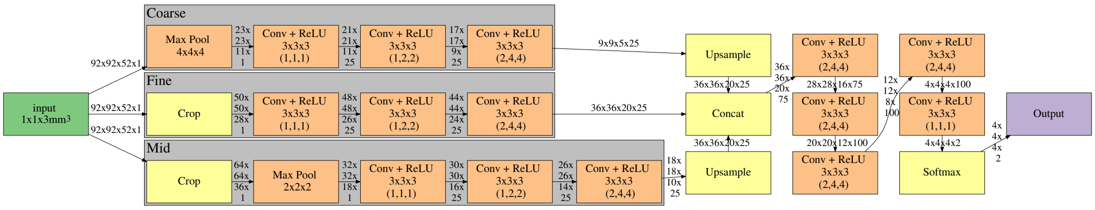

Figure 3: *Structure of the FCN. Type of layer, kernel size and dilation rate of each layer are shown, in addition the size of the features are shown at each edge. Note that the feature size can be changed, the ones shown in the figure are the ones used for training. The following abbreviations are used, Conv: Convolutional layer and ReLU: Rectified Linear Unit. Best viewed in color.*

The choice of what method to use was empirically decided individually for each organ. The evaluation used for the decision was how well the network performed on the validation set. The networks were trained in mini batches using stochastic gradient descent with Nesterov's momentum [16] and weight decay. The training parameters were set to: batch size 100, learning rate $5 \cdot 10^{-3}$, momentum weight 0.9, weight decay 10^{-5}. The error function used was negative log likelihood and the weights were randomly initialized. When an unbalanced training set was used the loss was multiplied by a factor k for foreground samples where k is the ratio between background and foreground samples. To avoid overfitting, dropout was applied during training [17]. The networks were trained for 10 epochs or more. The network that obtained the highest validation score was finally picked for the segmentation of the test images (see experimental section for different network and data settings).

FCN Setup

The FCN network takes an input box of varying size and hence also outputs the organ probability of all voxels in a box. During training the size of the input box is $92x92x52$ giving an output of size $4x4x4$. However, during inference the input size is changed to $120x120x64$ giving an output size of $32x32x16$. The network has three separate pipelines processing the information at three different scales (fine, middle and coarse). The features from the three pipes are then merged and processed jointly through several dilated convolution layers [18]. The network is designed to allow for a large receptive field while still being able to process information at the image original resolution. A schematic of the network is shown in Figure 3.

Implementation and training.

For the implementation of the FCN the framework keras [19] with Tensorflow [20] backend was used. The samples for the training and validation set is created by dividing the area inside and around the organ into seven different areas. A schematic of these areas is shown in Figure 4. The voxels inside the organ are divided into two regions. Region one contains all voxels further than $\alpha = 3$ mm away from the organ boundary and region two contains all voxels closer than α away from the boundary. The voxels outside the organ are divided into five regions, the first one contains all voxels closer than α to the organ boundary. Each other region are also characterized by the distance to the organ boundary and has a thickness of $\beta = 10$ mm. Each region is sampled with a specific probability. For the seven regions, from inside out, these probabilities are $25\%, 25\%, 15\%, 12.5\%, 12.5\%, 5\%$ and 5%.

Due to creating the training set by random sampling there might be a lot of voxels difficult to classify not included. To overcome this, the training set is altered during the training process, enabling us to extract so called *hard examples*. This is done by, every 10 epochs of training, the current network is applied to all voxels in the training set images. The incorrectly classified voxels are saved as hard examples. Then 20% of the training set used for the next 10 epochs are chosen from the hard examples. This is repeated for five times or until no further improvement can be seen.

2.3 Postprocessing

As a final step the probabilities from the convolutional neural network are thresholded, with a organ specific value estimated from data, in order to create a binary image. Everything but the largest connected component is set to zero producing the final segmentation.

3 Experimental Results

Our system was evaluated by submitting two entries to the MICCAI 2015 challenge "Multi-Atlas Labeling Beyond the Cranial Vault" in the free competition for organ segmentation in the abdomen [13]. In this challenge, there are 30 CT images coupled with manual segmentations of 13 organs, listed in Table 1. These 30 images and segmentations are available for method development and validation. Out of the 30 images 20 were used for training and 10 for validation.

In addition to these images, training data from the VISCERAL challenge [21] was also used for training. The VISCERAL training data consists of 20 unenhanced whole body CT images and 20 contrast enhanced CT images over the abdomen

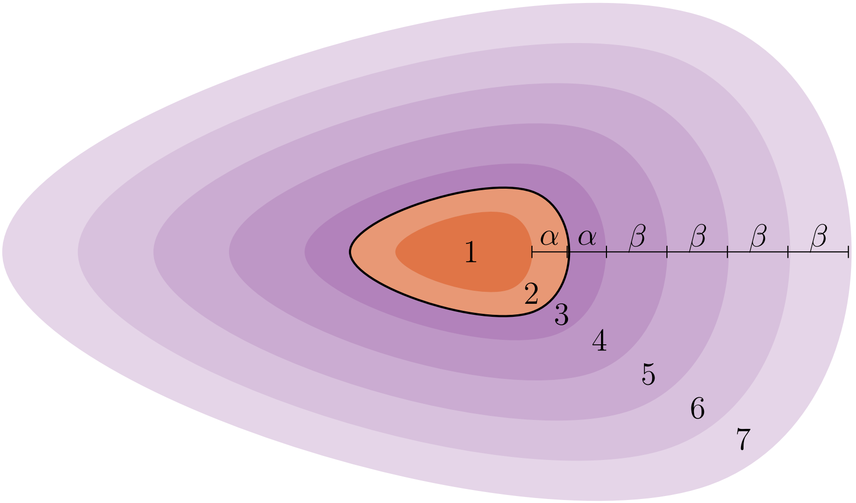

Figure 4: *Sampling strategy for stochastic training. Given an organ (the orange region inside the bold line), the image volume is divided into seven different regions, depending on the distance to the organ boundary. The different regions are then sampled at different rates.*

and thorax. In these images, organ ids 1, 2, 3, 4, 6, 8, 11, 12 and 13 were manually segmented. The unenhanced whole body CT images were excluded from the training set for organs with organ ids 1, 2, 6, 8, 10, 11 and 12 since they differed too much from the enhanced CT images. All images were resampled to the same resolution of 1 mm × 1 mm × 3mm. For the right kidney, networkk parameters were set based on on a training set formed by samples from both the right and the left kidney. For the stomach, the data set was expanded with distorted CT images and for the left adrenal gland, an unbalanced data set was used with twice as many background samples as foreground samples.

The test set of the MICCAI challenge consists of 20 CT images. The submitted segmentations are evaluated by calculating the Dice coefficient for each organ. The final results are given in Table 1 with the currently two best competitors:

- **IMI** - algorithm name: *IMI_deeds_SSC_jointCL* submitted by Mattias Heinrich at the Institute of Medical Informatics, Lübeck, Germany.
- **CLS** - algorithm name: *CLSIMPLEJLF_organwise* submitted by Zhoubing Xu at the Vanderbilt University, Nashville, TN, USA.
- **Other** - this column contains results from other competitors. The score is only shown if they are the highest for that organ.

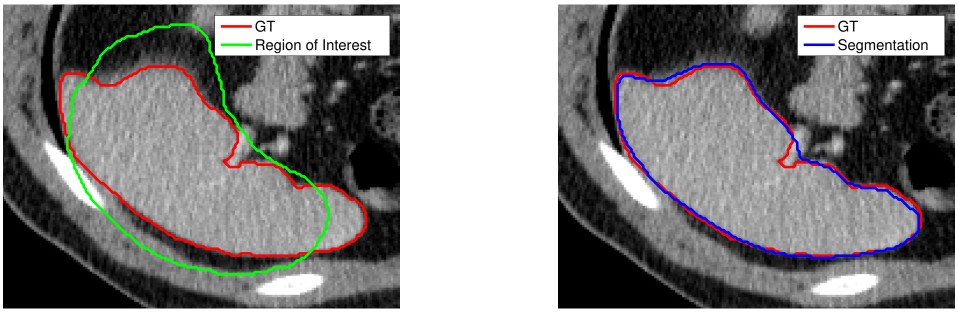

Figure 5: *Example of the resulting segmentation of the spleen for a CT slice. Note that even though the initial region of interest did not contain the entire organ the final result still does. This is due to region growing.*

3.1 Runtimes

The organ center estimation took about 20 seconds. For the voxelwise classification using a CNN, the runtimes between using the CNN-sw and FCN are compared for each organ. The results for these are shown in Table 2. All CNN computations were performed on a GeForce GTX TITAN X GPU with Maxwell architecture. The FCN network have a shorter runtime for most of the organs, this was expected since it has the ability to classify several voxels in each forward pass. However, due to the region growing algorithm used during inference, the total number of voxels classified may differ between the two methods. That is why the FCN have a longer runtime for some of the organs.

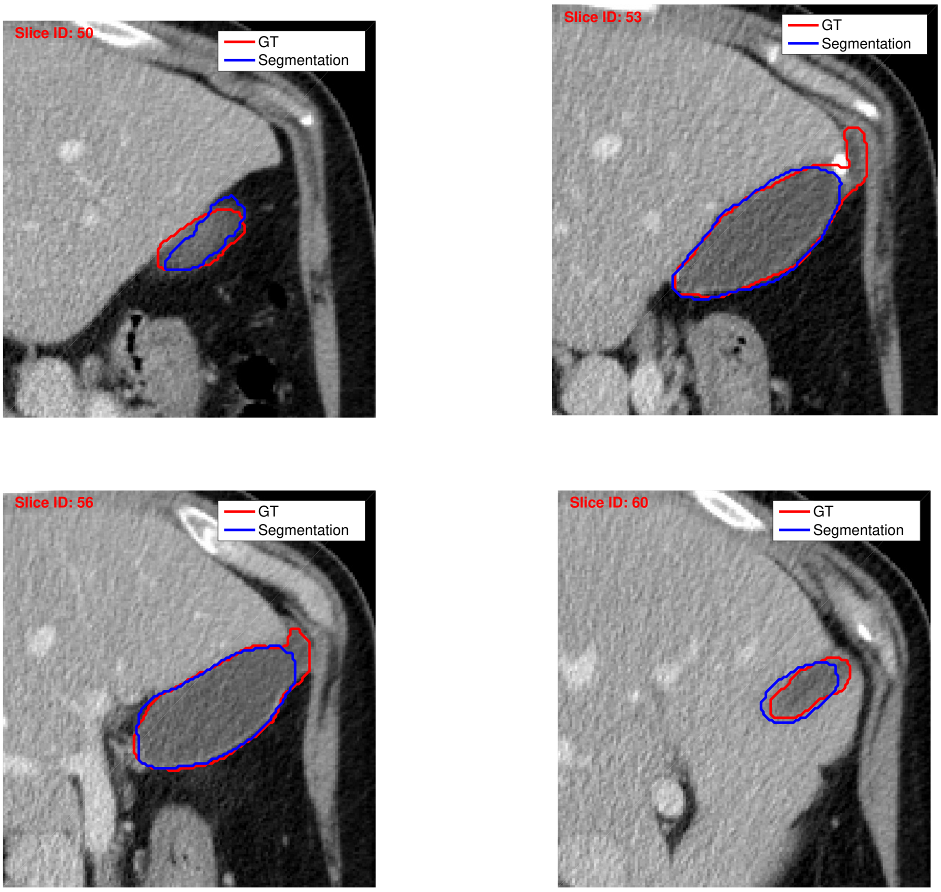

Figure 6: *Example of the resulting segmentation of the gallbladder. This is one of many images where our method achieved good results. For this image we got a Dice coefficient of 0.90.*

Table 1: *Final results measured in Dice metric for organ segmentation in CT images. Our approach gives the best results for 3 out of the 13 organs. Here '-' means that one of the specified methods achieved best result.*

Organ	*IMI*	*CLS*	*Other*	*CNN-sw$_{our}$*	*FCN$_{our}$*
Spleen	0.919	0.911	**0.964**	0.930	0.936
Right Kidney	**0.901**	0.893	-	0.866	0.897
Left Kidney	0.914	0.901	**0.917**	0.911	0.911
Gallbladder	0.604	0.375	-	**0.624**	0.613
Esophagus	**0.692**	0.607	-	0.662	0.588
Liver	0.948	0.940	-	0.946	**0.949**
Stomach	**0.805**	0.704	-	0.775	0.764
Aorta	0.857	0.811	-	0.860	**0.870**
Inferior Vena Cava	**0.828**	0.760	-	0.776	0.758
Portal and Splenic Vein	0.754	0.649	**0.756**	0.567	0.715
Pancreas	**0.740**	0.643	-	0.602	0.646
Right Adrenal Gland	0.615	0.557	-	**0.631**	0.630
Left Adrenal Gland	0.623	0.582	-	0.583	**0.631**
Average	**0.790**	0.723	-	0.757	0.767

Table 2: *Runtime results for the voxelwise classification comparing the CNN-sw to the FCN network archiectures. The results show the average runtime in seconds of the complete voxelwise classification process (including all the region growing iterations) for the test set.*

Organ	*CNN-sw* [s]	*FCN* [s]
Spleen	118	49
Right Kidney	89	25
Left Kidney	92	23
Gallbladder	108	19
Esophagus	83	22
Liver	234	88
Stomach	336	107
Aorta	139	201
Inferior Vena Cava	129	72
Portal and Splenic Vein	321	256
Pancreas	111	61
Right Adrenal Gland	47	17
Left Adrenal Gland	49	32

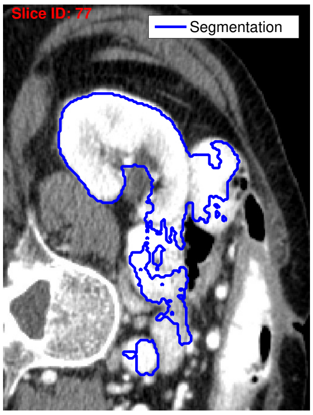

Figure 7: *Example of the resulting segmentation of the right kidney for a CT slice from the test set. The final segmentation is marked in blue. This segmentation was one of the examples where the method performed poorly.*

4 Discussion

During evaluation of our method we noticed a substantial difference in performance for the validation set and the test set. Using the CNN-sw the mean Dice coefficient on the validation set was 0.790 while the same measure on the test set was 0.757. This difference was most apparent for some organs such as the right kidney where the Dice coefficient dropped from 0.940 to 0.866 and the inferior vena cava where the Dice coefficient dropped from 0.823 to 0.775. This means that our networks do not generalize well to the test set for these organs which might be an indication of overfitting and that the input features and the structure of our network is not ideal to learn high order information that generalize to all other CT images. However, since the validation data has not been used for the actual training, only for the decision on when to stop the training, these differences might not be only due to overfitting. Instead it might be due to the existence of anatomical variations in the test set that differ too much from anything seen in the training and validation images for the network to perform well. A specific example of where our method performed poorly on the test data, for an organ with good validation results, is shown in Figure 7. Here, the network has classified most of the right kidney correctly. However, it has also classified a lot of surrounding organs or tissues as right kidney as well.

The ideal solution to this problem would be to include more images in the training set. This however, requires more manually segmented CT images which are not always easy to acquire. Other approaches to solve this problem would be to train a network on several organs, and then fine tune the network weights for each specific organ. This could enable the network to learn higher order features that differentiates well between all organs in the CT image, not only between the organs located closest to the organ that is currently being segmented. Other future work would be to add a postprocessing step using a Conditional Random Field (CRF). Recently, methods for training CRFs and CNNs jointly have been proposed for medical images [22].

5 Conclusion

In this paper, an efficient system for abdominal organ segmentation was presented. Our approach first uses a robust localization algorithm for finding the region of interest. As a second step a convolutional neural network is applied performing voxelwise classification. Two convolutional neural networks of different architecture are compared for this task. The method was evaluated by submitting two entries to the MICCAI2015 challenge “Multi-Atlas Labeling Beyond the Cranial Vault” in the free competition for organ segmentation in the abdomen. The entries achieved on par with state-of-the-art for a majority of the organs with a mean Dice coefficient of 0.757 and 0.767 for CNN-sw and FCN, respectively.

Bibliography

[1] R. Wolz, C. Chu, K. Misawa, M. Fujiwara, K. Mori, and D. Rueckert, “Automated abdominal multi-organ segmentation with subject-specific atlas generation,” *Medical Imaging, IEEE Transactions on*, vol. 32, no. 9, pp. 1723–1730, 2013.

[2] Z. Wang, K. K. Bhatia, B. Glocker, A. Marvao, T. Dawes, K. Misawa, K. Mori, and D. Rueckert, “Geodesic patch-based segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*, ser. Lecture Notes in Computer Science, 2014, vol. 8673, pp. 666–673.

[3] H. Park, P. H. Bland., and C. R. Meyer, “Construction of an abdominal probabilistic atlas and its application in segmentation,” *Medical Imaging, IEEE Transactions on*, vol. 22, no. 4, pp. 483–492, April 2003.

[4] C. Chu, M. Oda, T. Kitasaka, K. Misawa, M. Fujiwara, Y. Hayashi, Y. Nimura, D. Rueckert, and K. Mori, “Multi-organ segmentation based on spatially-divided probabilistic atlas from 3D abdominal CT images,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2013, pp. 165–172.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25.* Curran Associates, Inc., 2012, pp. 1097–1105.

[6] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[7] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Mitosis detection in breast cancer histology images with deep neural networks,” in *MICCAI*, vol. 2, 2013, pp. 411–418.

[8] H. Roth, L. Lu, A. Farag, H. Shin, J. Liu, E. Turkbey, and R. Summers, “Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, ser. Lecture Notes in Computer Science, 2015, vol. 9349, pp. 556–564.

[9] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9351. Springer, 2015, pp. 234–241, (available on arXiv:1505.04597 [cs.CV]). [Online]. Available: http://lmb.informatik.uni-freiburg.de//Publications/2015/RFB15a

[10] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation.* Cham: Springer International Publishing, 2016, pp. 424–432. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46723-8_49

[11] T. Brosch, L. Y. W. Tang, Y. Yoo, D. K. B. Li, A. Traboulsee, and R. Tam, “Deep 3d convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation,” *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1229–1239, 2016.

[12] K. Kamnitsas, L. Chen, C. Ledig, D. Rueckert, and B. Glocker, “Multi-scale 3d convolutional neural networks for lesion segmentation in brain mri,” *Ischemic Stroke Lesion Segmentation*, p. 13, 2015.

[13] Z. Xu, “Multi-atlas labeling beyond the cranial vault - workshop and challenge,” 2016, [Online; accessed 10-January-2017]. [Online]. Available: https://www.synapse.org/#!Synapse:syn3193805/wiki/217752

[14] F. Kahl, J. Alvén, O. Enqvist, F. Fejne, J. Ulén, J. Fredriksson, M. Landgren, and V. Larsson, “Good features for reliable registration in multi-atlas segmentation,” in *VISCERAL Anatomy3 Segmentation Challenge*, 2015, pp. 12–17.

[15] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in *BigLearn, NIPS Workshop*, 2011.

[16] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th international conference on machine learning (ICML-13)*, 2013, pp. 1139–1147.

[17] N. Sivastava, G., A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[18] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in *ICLR*, 2016.

[19] F. Chollet, “keras,” https://github.com/fchollet/keras, 2015.

[20] J. Allaire, D. Eddelbuettel, N. Golding, and Y. Tang, *tensorflow: R Interface to TensorFlow*, 2016. [Online]. Available: https://github.com/rstudio/tensorflow

[21] A. Hanbury, H. Müller, G. Langs, and B. H. Menze, *The Future Internet: Future Internet Assembly 2013: Validated Results and New Horizons*, Berlin, Heidelberg, 2013, ch. Cloud–Based Evaluation Framework for Big Data, pp. 104–114.

[22] M. Larsson, J. Alvén, and F. Kahl, “Max-margin learning of deep structured models for semantic segmentation,” in *20th Scandinavian Conference Image Analysis, (SCIA 2017).* Springer International Publishing, 2017.