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

Måns Larsson[1], Jennifer Alvén[1], and Fredrik Kahl[1,2]

[1] Chalmers University of Technology, Gothenburg, Sweden,
{mans.larsson,alven,fredrik.kahl}@chalmers.se
[2] Centre for Mathematical Sciences, Lund University, Lund, Sweden

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.

Keywords: Segmentation, Convolutional Neural Networks, Markov Random Fields

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 [12] and object detection [8]. 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 [9, 17, 18].

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 [4] where they used CNNs to form the unary

potential of the dense CRF model presented by Krähenbühl *et al.* in [11]. However, the CNN and the CRF models are trained separately in [4] 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 [5, 14–16, 24, 27, 31]. 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 [26]. 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 [14, 31], the inference step is approximately solved using a few iterations of the mean-field algorithm or gradient descent, respectively. Similarly, in [5], 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 [2] 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 [21] Ning *et al.* combine a CNN with an energy based model, similar to a MRF, for segmentation of cell nuclei and in [4] 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 [27] 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. In [24] Schwing *et al.* develop a method for jointly training the model from [4], *i.e.* a CNN coupled with a dense CRF. Further, Chen and Schwing present a more general framework for joint learning of deep structured models that they apply to image tagging and word from image problems in [5]. Zheng *et al.* [31] show that the mean-field inference algorithm

with Gaussian pairwise potentials from [11] could be modeled with Recurrent Neural Networks. This enabled them to train their model within a standard deep learning framework using a log-likelihood loss. In parallel, Lin *et al.* [16] developed a method where CNNs are used to estimate the messages in the message passing algorithm for CRF inference. This in contrast to most other work where the CNNs have been used to estimate the potential functions of the CRFs. In [15], they formulated a CRF model with CNNs for estimating the unary and pairwise potentials. For training they used the piecewise training approach proposed by Sutton and McCallum in [25].

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 [6, 20, 23]. Recently, more intricate deep learning approaches have been proposed. Ronneberger *et al.* [22] proposed the U-Net, a network based on the idea of "fully convolutional networks" [17]. A similar network structure was also proposed by Brosch *et al.* in [3]. 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 $\boldsymbol{x}^{(n)} = (x_1^{(n)}, \ldots, x_N^{(n)})$ and $\boldsymbol{y}^{(n)} = (y_1^{(n)}, \ldots, y_N^{(n)})$ 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

$$E(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{i \in \mathcal{V}} \boldsymbol{w}_u \boldsymbol{\phi}_i^1(y_i, \boldsymbol{x}; \boldsymbol{\theta}) + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{w}_{pw} \boldsymbol{\phi}_{ij}^2(y_i, y_j, \boldsymbol{x}), \tag{2}$$

where $\mathcal{V}$ is the set of nodes (*i.e.* pixels) and $\mathcal{E}$ is the set of edges connecting neighbouring pixels. Note that this energy is linear with respect to the weights, $\boldsymbol{w}_u$ and $\boldsymbol{w}_{pw}$.

The unary term of the energy E has the following form

$$\boldsymbol{w}_u \boldsymbol{\phi}_i^1(y_i, \boldsymbol{x}; \boldsymbol{\theta}) = w_1 \log(\Phi_i(y_i, \boldsymbol{x}, \boldsymbol{\theta})), \tag{3}$$

where $\Phi(y_i, \boldsymbol{x}, \boldsymbol{\theta})$ 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 $\mathbb{1}_{y_i \neq y_j}$ denotes the indicator function equaling one if $y_i \neq y_j$, the pairwise term has the following form

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

2.1 Inference

Given an input instance $\boldsymbol{x}^{(n)}$, 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 constraint $\boldsymbol{w}_2 \leq 0$ when learning the weights to make the energy submodular. This means that graph cut algorithms can be used to efficiently find a global optimum [10].

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.* [26] 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.* [26]. 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

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

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 1.

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: $\boldsymbol{y}^* \leftarrow \arg\min_{\boldsymbol{y}} E(\boldsymbol{y}, \boldsymbol{x}^{(n)}, \boldsymbol{w}) - \Delta(\boldsymbol{y}^{(n)}, \boldsymbol{y})$
    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 1: Pseudocode for the CRF weight learning algorithm from [26].

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 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:

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

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

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

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

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 $\boldsymbol{\theta}$ without the CRF part of the model. This 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 1 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 2.

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 $\rightarrow \boldsymbol{z}$
    CRF learning by Algorithm 1
    add low-energy labelings to set $S^{(n)}$
    calculation of objective derivative $\rightarrow \frac{\partial \gamma_b}{\partial z_i}$, back-propagation $\rightarrow \frac{\partial \gamma_b}{\partial \theta_j}$
    update CNN weights, $\theta_j \leftarrow \theta_j + \eta \frac{\partial \gamma_b}{\partial \theta_j}$
  end
  update CRF weights by solving (8) $\rightarrow \boldsymbol{w}, \gamma$
end
Output: $\boldsymbol{w}$, $\boldsymbol{\theta}$

Algorithm 2: Pseudocode for joint learning of parameters in batches.

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.* [17]. 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 [28].

3.1 Weizmann Horse Dataset

The Weizmann Horse dataset [2] 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 [29], CRF-Grad [14] and PatchCut [30]. There are a few variations of the Weizmann Horse dataset available, we used the same one as in PatchCut [30]. 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 [30]	84.03	CNN (only)	79.97
ReSeg [29]	**91.60**	CNN + CRF (piecewise)	81.62
CRF-Grad [14]	83.98	CNN + CRF (joint)	84.54

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 [13]. 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 [1]. 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 [19]. The dataset includes in total 1500 2D slices which are divided

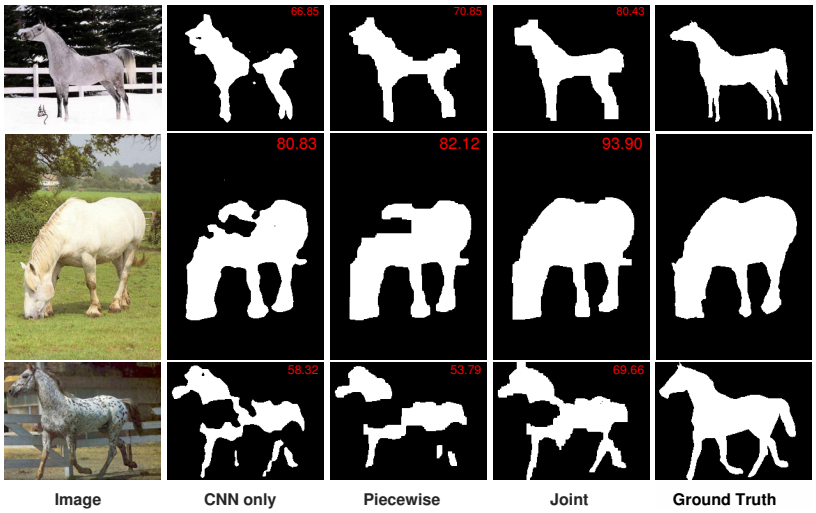

Fig. 1. Qualitative results on the Weizmann Horse dataset. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The number shown in the upper right corner is the Jaccard index (%).

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 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.

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.

	Mean Jaccard index (%)			
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**

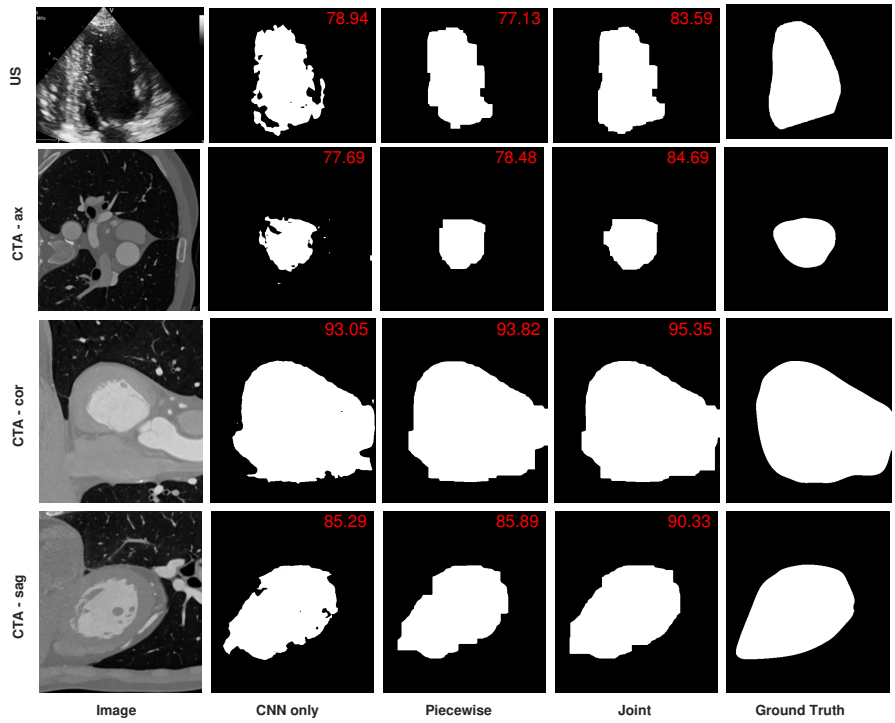

Fig. 2. Qualitative results on the Cardiac ultrasound dataset (US) and the Cardiac CTA Dataset (CTA). For the CTA dataset, the different types of slices are evaluated separately (ax - axial, cor - coronal and sag - sagittal). "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF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

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 [7]. 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.

References

1. G Bergström et al. The Swedish CArdioPulmonary bioImage Study: Objectives and design. *J. of Internal Medicine*, 278(6):645–659, 2015.
2. E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *European Conf. on Computer Vision*, pages 109–122, 2002.
3. T. Brosch et al. Deep 3D convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation. *IEEE Trans. Med. Imag.*, 35(5):1229–1239, 2016.
4. L.-C. Chen et al. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *arXiv preprint arXiv:1412.7062*, 2014.
5. L.-C. Chen, A.G. Schwing, A.L. Yuille, and R. Urtasun. Learning deep structured models. In *Proc. of the Int. Conf. on Machine Learning*, 2015.
6. D.C. Cireşan, A. Giusti, L.M. Gambardella, and J. Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, pages 411–418, 2013.
7. J. Deng et al. ImageNet: A large-scale hierarchical image database. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
8. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 580–587, 2014.
9. A. Giusti et al. Fast image scanning with deep max-pooling convolutional neural networks. In *IEEE Int. Conf. on Image Processing*, pages 4034–4038, 2013.
10. V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004.

11. P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *Adv. Neural. Inf. Process. Syst.*, pages 109–117, 2011.
12. A. Krizhevsky, I. Sutskever, and G.E. Hinton. ImageNet classification with deep convolutional neural networks. In *Adv. Neural. Inf. Process. Syst.*, pages 1097–1105, 2012.
13. R.M. 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.*, 28(1):1–39, 2015.
14. Måns Larsson et al. Learning arbitrary potentials in CRFs with gradient descent. *arXiv preprint (Under submission)*, 2017.
15. G. Lin et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv preprint arXiv:1504.01013*, 2015.
16. G. Lin, C. Shen, I. Reid, and A. van den Hengel. Deeply learning the messages in message passing inference. In *Adv. Neural. Inf. Process. Syst.*, pages 361–369, 2015.
17. J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2015.
18. H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proc. of the IEEE Int. Conf. on Computer Vision*, pages 1520–1528, 2015.
19. A. Norlén et al. Automatic pericardium segmentation and quantification of epicardial fat from computed tomography angiography. *J. of Medical Imaging*, 3(3), 2016.
20. A. Prasoon et al. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, pages 246–253, 2013.
21. M. Ranzato et al. Automatic recognition of biological particles in microscopic images. *Pattern Recognition Letters*, 28(1):31–39, 2007.
22. O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.
23. H. Roth et al. DeepOrgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, pages 556–564, 2015.
24. A.G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.
25. C. Sutton and A. McCallum. Piecewise training for undirected models. *arXiv preprint arXiv:1207.1409*, 2012.
26. M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *European Conf. on Computer Vision*, pages 582–595, 2008.
27. J.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.*, pages 1799–1807, 2014.
28. A. Vedaldi and K. Lenc. MatConvNet: Convolutional neural networks for MATLAB. In *Proc. of the 23rd ACM Int. Conf. on Multimedia*, pages 689–692, 2015.
29. F. Visin et al. ReSeg: A recurrent neural network-based model for semantic segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, pages 41–48, 2016.
30. J. Yang et al. PatchCut: Data-driven object segmentation via local shape transfer. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1770–1778, 2015.
31. Shuai Zheng et al. Conditional random fields as recurrent neural networks. In *Proc. of the IEEE Int. Conf. on Computer Vision*, pages 1529–1537, 2015.

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 $h_k \leq 0$,

$$E(\boldsymbol{y}^{(n)}) - E(\boldsymbol{y}) + \Delta(\boldsymbol{y}^{(n)}, \boldsymbol{y}) - \xi_n \leq 0 \tag{16}$$

$$-\xi_n \leq 0 \tag{17}$$

$$-\boldsymbol{w}_{pw} \leq 0. \tag{18}$$

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 mention in Section 2.3, for the constraint 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 Equations 12 to 15 we also need $\frac{\partial h_i}{\partial w_1}$, $\frac{\partial h_i}{\partial w_2}$, $\frac{\partial h_i}{\partial w_3}$, $\frac{\partial h_i}{\partial w_4}$ and $\frac{\partial^2 h_i}{\partial w_2 \partial z_j}$. Note that all all other second order derivative equal zero.

Energy Constraints The energy constraints are formulated as follows

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

where the index n denotes the ground truth instance that the energy is coupled with. Note that there are several of these constraint per training instance, one for each $\boldsymbol{y} \in S^{(n)}$. To simplify notation we have introduced $C_2^{(i)}$ and $C_3^{(i)}$, note that these are constant 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

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

with the derivative

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

We now calculate the needed derivatives, excluding the ones equaling zero

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

$\boldsymbol{y}^i \in S^{(n)}$ 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

$$h_i = -\xi_i.$$

Calculate the needed derivatives, excluding the ones equaling 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_3 or w_4).

Final Explicit Linear System

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

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

$\boldsymbol{A}$ and $\boldsymbol{b}$ are calculated from the equations satisfied at the optimum. Following the derivations in Section 2.3 and Appendix 4 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

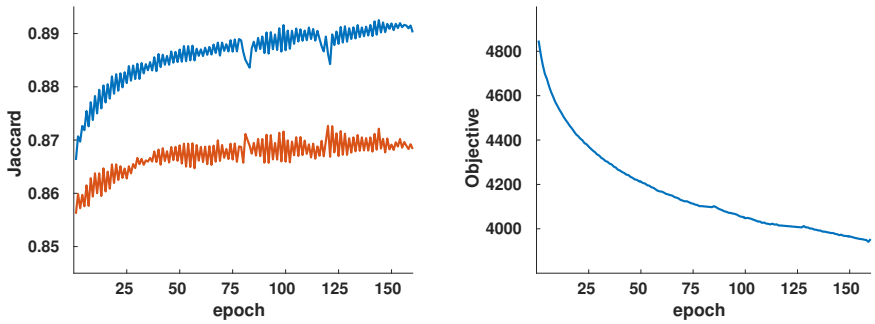

Fig. 3. Joint training results for the Weizmann Horse dataset. The left figure shows the mean Jaccard index versus epochs for the training images (blue upper graph) and the validation images (red lower graph). The right figure shows the max-margin objective, γ, from (8) versus epochs.

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

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, $\frac{\partial h_i}{\partial z_j} = 0$ or see Appendix 4 for an explicit formulation, 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.

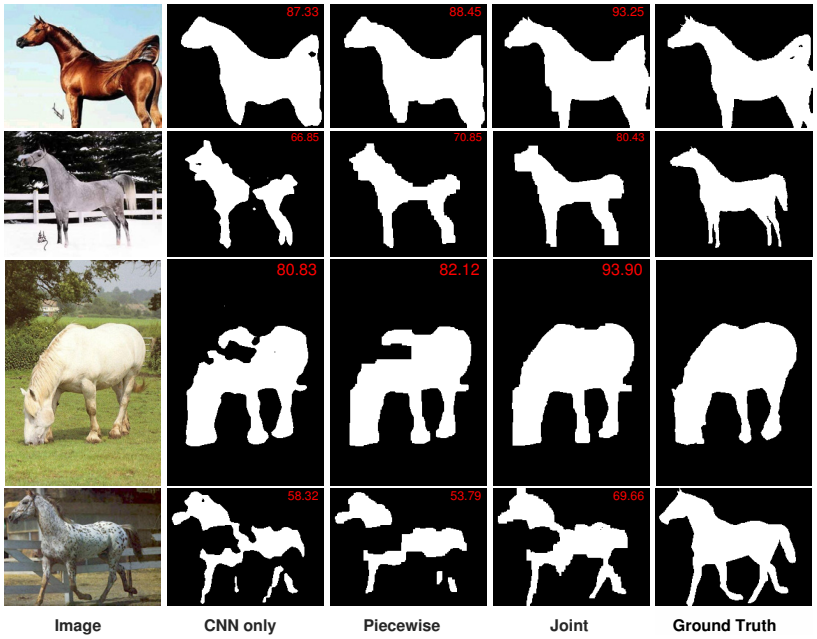

Fig. 4. Qualitative results on the Weizmann Horse dataset. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.

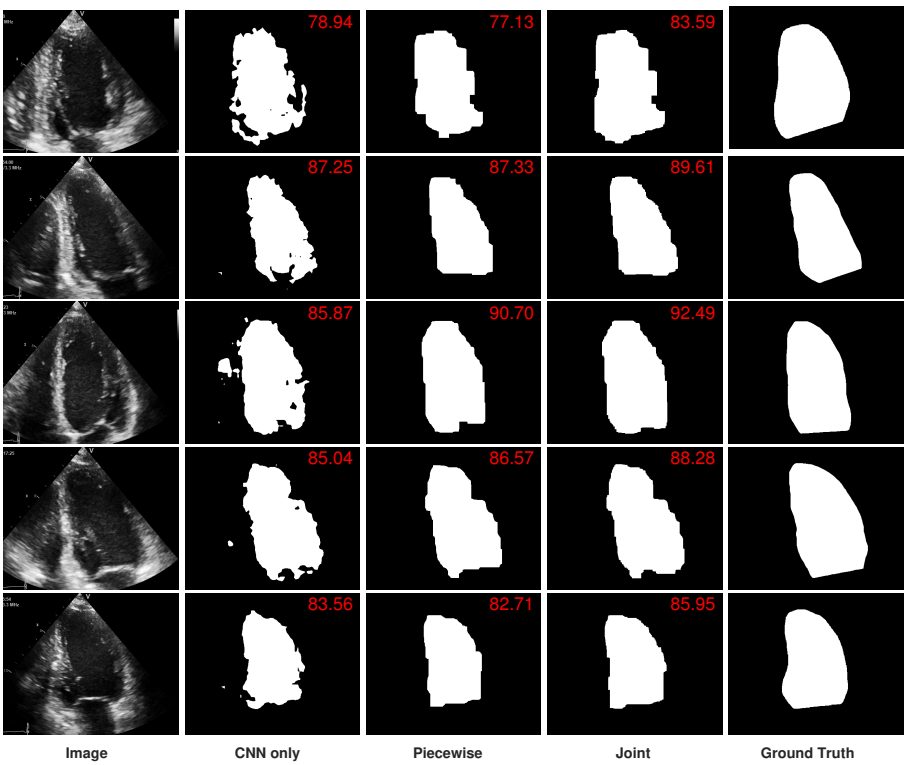

Fig. 5. Qualitative results on the Cardiac ultrasound dataset. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.

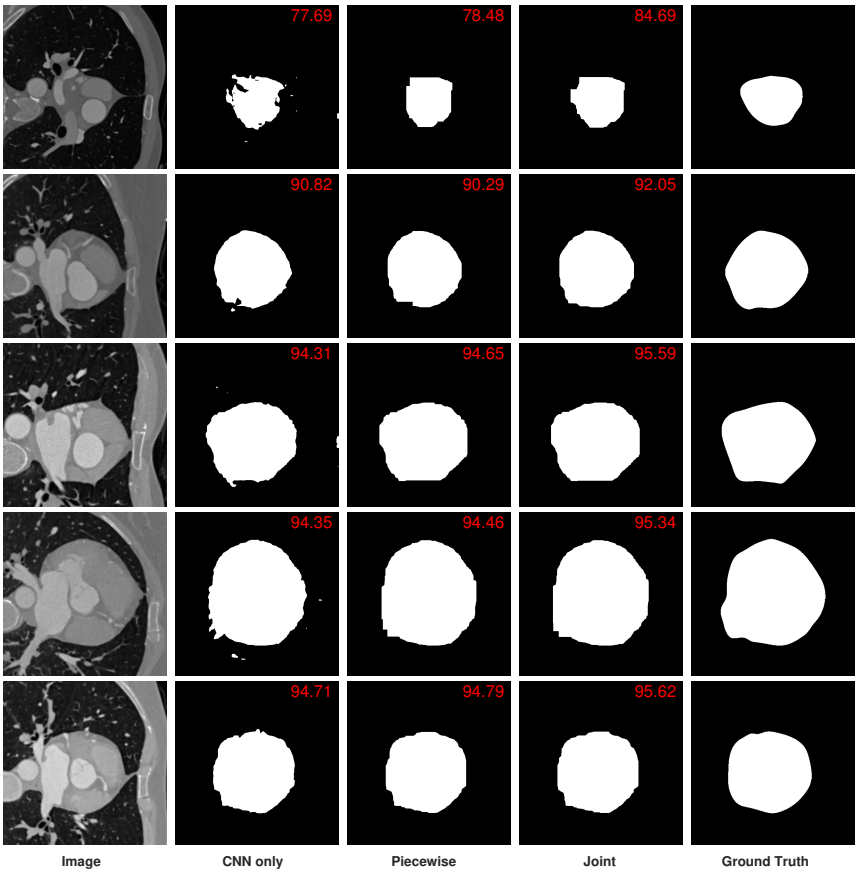

Fig. 6. Qualitative results on the Cardiac CTA Dataset for the axial slices. ”Piecewise” denotes ”CNN + CRF (piecewise)” and ”Joint” denotes ”CNN + CRF (joint)”. The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.

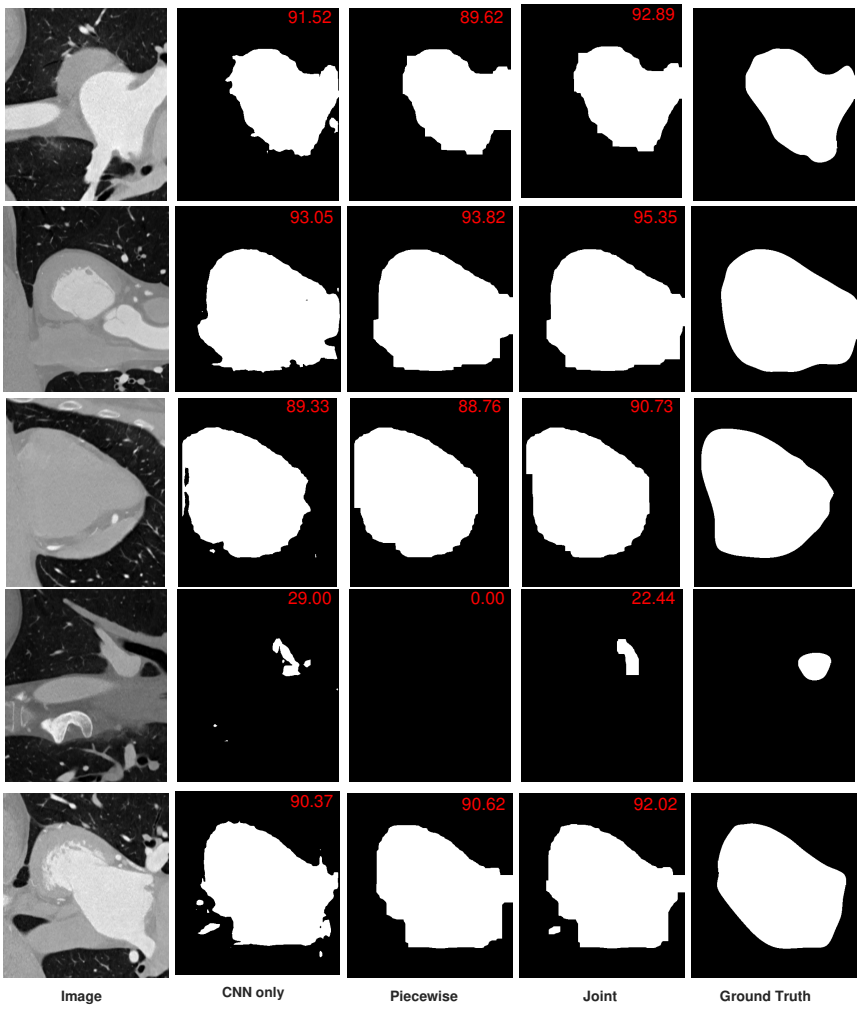

Fig. 7. Qualitative results on the Cardiac CTA Dataset for the coronal slices. "Piecewise" denotes "CNN + CRF (piecewise)" and "Joint" denotes "CNN + CRF (joint)". The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.

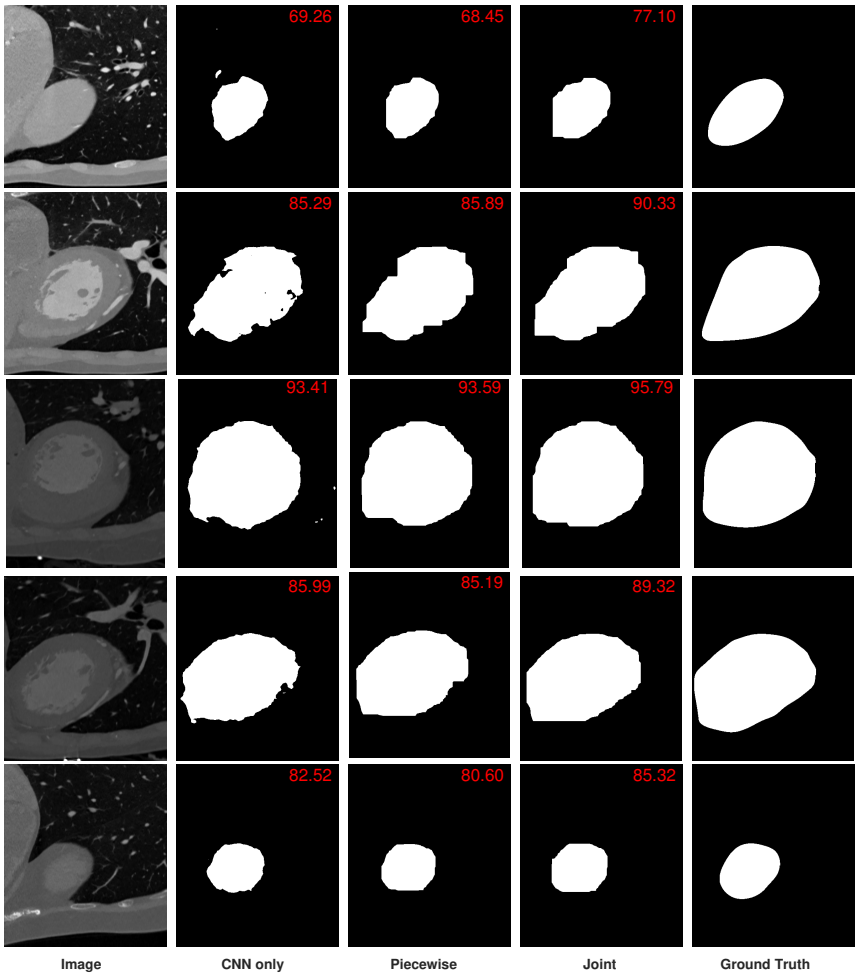

Fig. 8. Qualitative results on the Cardiac CTA Dataset for the sagittal slices. ”Piecewise” denotes ”CNN + CRF (piecewise)” and ”Joint” denotes ”CNN + CRF (joint)”. The red number shown in the upper right corner is the Jaccard index (%). The figure is best viewed in color.