Revisiting Deep Structured Models for Pixel-Level Labelling with Gradient-Based Inference [*†]

Måns Larsson [‡], Anurag Arnab [§], Shuai Zheng [§], Philip Torr [§], and Fredrik Kahl [‡]

Abstract. Pixel-level labelling tasks such as semantic segmentation have witnessed 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.

Key words. Deep Structured Models, Conditional Random Fields, Deep Learning, Semantic Segmentation

AMS subject classifications. 68T45, 68R10

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 [32, 9, 4]. 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 [42] using graph cuts and the semantic segmentation works of [34, 45] 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 [27, 44, 41]. However, we observe that several leading segmentation approaches still include CRFs, either as a post-processing step [14, 15, 24, 13], or as part of the deep neural network itself [48, 37, 3, 39, 31, 47].

We also leverage this idea of embedding inference of graphical models into a neural network. An early example of this idea was presented in [12] where the authors back propagated through the Viterbi algorithm when designing a document recognition system. Similar to [48, 3, 6, 47], we use a recurrent neural network to unroll the iterative inference steps of a CRF. This was first used in [48] and [43] to imitate mean-field inference and to train a fully

[*]Submitted to the editors on 2018-01-30.

[†]This article is an extended version of the conference paper "A Projected Gradient Descent Method for CRF Inference allowing End-To-End Training of Arbitrary Pairwise Potentials" [35].

[‡]Department of Electrical Engineering, Chalmers University of Technology , Gothenburg (mans.larsson@chalmers.se, fredrik.kahl@chalmers.se).

[§]Department of Engineering Science at University of Oxford (aarnab@robots.ox.ac.uk, szheng@robots.ox.ac.uk, phst@robots.ox.ac.uk).

This manuscript is for review purposes only.

convolutional network [40, 14] 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 probability 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 [6] and the followup work [7] for multi-label classification problems in machine learning with impressive results. Moreover, [20, 2] 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 [33, 48, 3] 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 [16], 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 [31] 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 [37] 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 [29], 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 [29], 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 [21] 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 [47].

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 [1]. 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 [17] 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 [8] 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 [30] and all source code is publicly available to facilitate further research. [1]

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

$$P(\mathcal{X} = \boldsymbol{x}|\boldsymbol{I}) = \frac{1}{Z(\boldsymbol{I})} \exp(-E(\boldsymbol{x}|\boldsymbol{I})), \tag{2.1}$$

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

$$E(\boldsymbol{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j) \tag{2.2}$$

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

[1] https://github.com/maunzzz/caffe-crfgd

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

$$\psi_i(x_i) = -w_u \log(z_{i:x_i} + \epsilon) \tag{2.3}$$

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.

2.1.1. 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

$$\psi_{ij}(x_i, x_j) = k^{spatial}_{x_i,x_j}(\boldsymbol{p}_i - \boldsymbol{p}_j) + k^{bilateral}_{x_i,x_j}(\boldsymbol{f}_i - \boldsymbol{f}_j) \tag{2.4}$$

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.

2.1.2. Filterbank. The pairwise potentials of the *filterbank* version has the following form

$$\psi_{ij}(x_i, x_j) = \sum_{f=1}^{F} g_f(\boldsymbol{p}_i, I) k^{spatial}_{x_i,x_j,f}(\boldsymbol{p}_i - \boldsymbol{p}_j), \tag{2.5}$$

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 be 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

$$\begin{aligned} \min \quad & \sum_{i\in\mathcal{V},\lambda\in\mathcal{L}} \psi_i(\lambda) x_{i:\lambda} + \sum_{\substack{(i,j)\in\mathcal{E}\\ \lambda,\mu\in\mathcal{L}}} \psi_{ij}(\lambda,\mu) x_{i:\lambda} x_{j:\mu} \\ \text{s.t.} \quad & x_{i:\lambda} \in \{0,1\} && \forall i \in \mathcal{V}, \lambda \in \mathcal{L} \\ & \sum_{\lambda\in L} x_{i:\lambda} = 1 && \forall i \in \mathcal{V}. \end{aligned} \tag{2.6}$$

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

$$\begin{aligned} \min \quad & \sum_{i\in\mathcal{V},\lambda\in\mathcal{L}} \psi_i(\lambda) q_{i:\lambda} + \sum_{\substack{(i,j)\in\mathcal{E}\\ \lambda,\mu\in\mathcal{L}}} \psi_{ij}(\lambda,\mu) q_{i:\lambda} q_{j:\mu} \\ \text{s.t.} \quad & q_{i:\lambda} \geq 0 && \forall i \in \mathcal{V}, \lambda \in \mathcal{L} \\ & \sum_{\lambda\in\mathcal{L}} q_{i:\lambda} = 1 && \forall i \in \mathcal{V}. \end{aligned} \tag{2.7}$$

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 [11].

Proposition 2.1. *Let $E(\boldsymbol{x}^*)$ and $E(\boldsymbol{q}^*)$ denote the optimal values of (2.6) and (2.7), 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 (2.7) we propose an optimization scheme based on projected gradient descent, see Algorithm 3.1. It was designed with an extra condition in mind, that all operations should be differentiable to enable back propagation during training.

Algorithm 3.1 Algorithm 1. Projected gradient descent algorithm.

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

3.1. Gradient Computations. The gradient $\nabla_{\boldsymbol{q}} E$ of the objective function $E(\boldsymbol{q})$ in (2.7) has the following elements

$$\frac{\partial E}{\partial q_{i:\lambda}} = \psi_i(\lambda) + \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} \psi_{ij}(\lambda,\mu) q_{j:\mu}. \tag{3.1}$$

The contribution from the spatial kernel in ψ_{ij}, cf. (2.4), can be written as

$$v_{i:\lambda}^{spatial} = \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} k_{\lambda,\mu}^{spatial}(\boldsymbol{p}_i - \boldsymbol{p}_j) q_{j:\mu}. \tag{3.2}$$

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

$$v_{i:\lambda}^{bilateral} = \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} k_{\lambda,\mu}^{bilateral}(\boldsymbol{f}_i - \boldsymbol{f}_j) q_{j:\mu}. \tag{3.3}$$

For this computation we utilize the method presented by Jampani *et al.* [29] which is based on the permutohedral lattice introduced by Adams *et al.* [1]. 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

$$v_{i:\lambda}^{bank} = \sum_{f=1}^{F} g_f(\boldsymbol{p}_i, I) \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} k_{x_i,x_j,f}^{spatial}(\boldsymbol{p}_i - \boldsymbol{p}_j) q_{j:\mu}, \tag{3.4}$$

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 which 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

$$\tilde{\boldsymbol{q}}^{\boldsymbol{t+1}} = \mathbf{q}^{\mathbf{t}} - \gamma \, \nabla_{\mathbf{q}} \boldsymbol{E}, \tag{3.5}$$

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.* [19]. This method is used by by Larsson *et al.* in [35]. 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.* [5]. 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

$$q_{ij}^{k+1} = \frac{q_{ij}^{k} \exp\left(-t_k \nabla_{q^k} E\right)}{\sum q_{ij}^{k} \exp\left(-t_k \nabla_{q^k} E\right)}, \qquad t_k = \frac{\sqrt{2 \ln n}}{L_f} \frac{1}{\sqrt{k}} \tag{3.6}$$

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.

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 [34]. 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 3.1 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.

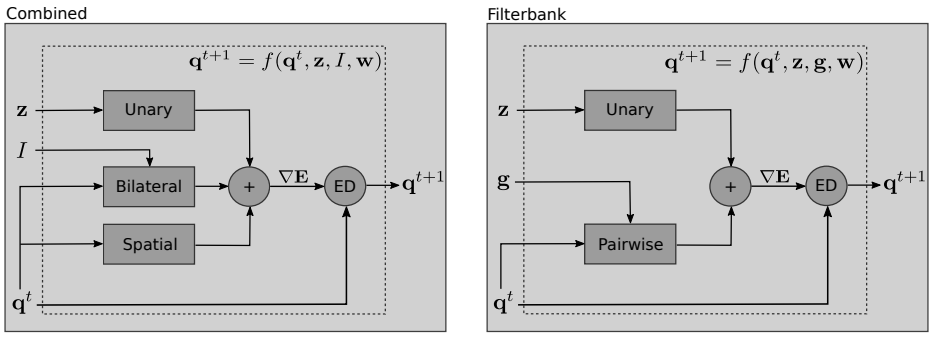

Figure 1. *The data flow of one iteration of the projected gradient descent algorithm. Each rectangle or circle represent an operation that can be performed within a deep learning framework, the ED component performs an entropic descent update step according to equation (3.6). Left: Combined version of CRF, Right: Filterbank version of CRF.*

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 (2.3) 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.* [29] 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.

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

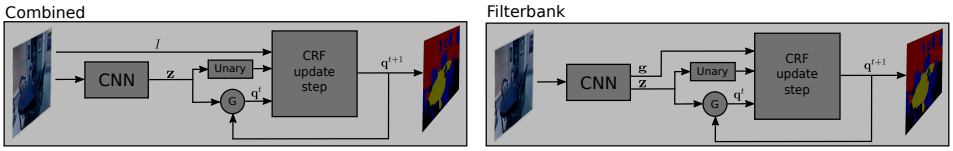

Figure 2. *The data flow of the deep structure model. Each rectangle or circle represent an operation that can be performed within a deep learning framework. Note that the CNN outputs both class probabilities* $\boldsymbol{z}$ *and filterbank features* $\boldsymbol{g}$ *for the filterbank version.*

are given according to

$$\frac{\partial L}{\partial q_{ij}^{k}} = \frac{q_{ij}^{k+1}}{q_{ij}^{k}} \left(\frac{\partial L}{\partial q_{ij}^{k+1}} - \sum_{l=1}^{n} \frac{\partial L}{\partial q_{il}^{k+1}} q_{il}^{k+1} \right), \tag{4.1}$$

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

$$\frac{\partial L}{\partial y_{ij}} = -t_k q_{ij}^{k+1} \left(\frac{\partial L}{\partial q_{ij}^{k+1}} - \sum_{l=1}^{n} \frac{\partial L}{\partial q_{il}^{k+1}} q_{il}^{k+1} \right). \tag{4.2}$$

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

5. Recurrent Formulation of 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, $\mathbf{z}$, 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, $\mathbf{g}$, 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 (2.7). Thus, one update step can be represented by:

$$\boldsymbol{q}^{t+1} = f(\boldsymbol{q}^{t}, \boldsymbol{z}, I, \boldsymbol{w}). \tag{5.1}$$

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 [33, 48], 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 [30] 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.* [18]. 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 [17]. 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.

Both the *combined* and the *filterbank* models can be trained from scratch, however the training converges faster and more reliably when the unary part is pretrained.

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) [34, 14, 48].

7. Experiments. We evaluate the proposed approach on three datasets: Weizmann Horse dataset [10], NYU V2 geometric dataset [46] and Pascal VOC 2012 [23]. In these experiments, we show that the proposed approach, has advantages over similar approaches such as CRF-RNN [48]. 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.

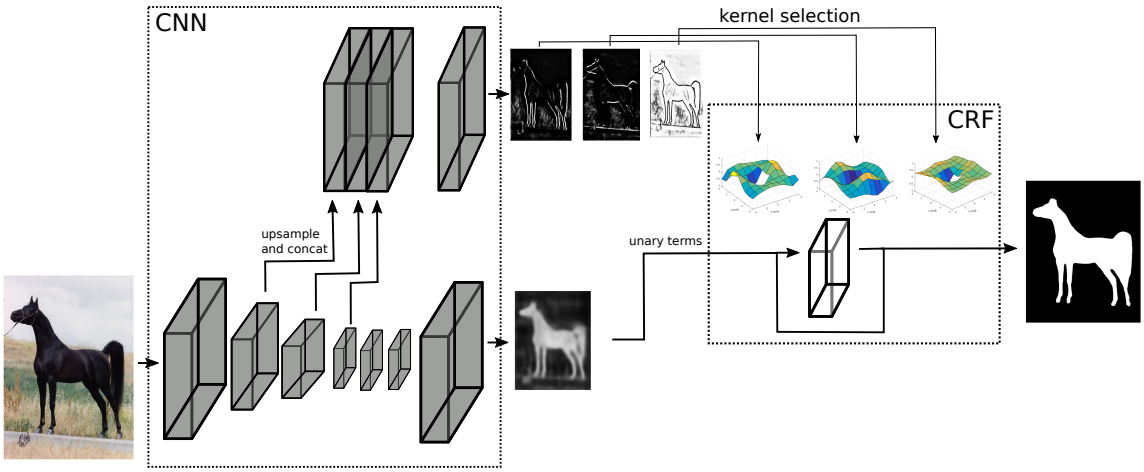

Figure 3. *Schematic of the filterbank version of our model. The CNN part outputs initial class probability maps as well as filter selection maps. The structure used for the CNN is a modified version of the Deeplab-LargeFOV [18] with an extra added head.*

Method	mIoU (%)
Unary CNN - Deeplab [14]	90.89
CRF-RNN [48]	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 [48]). 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

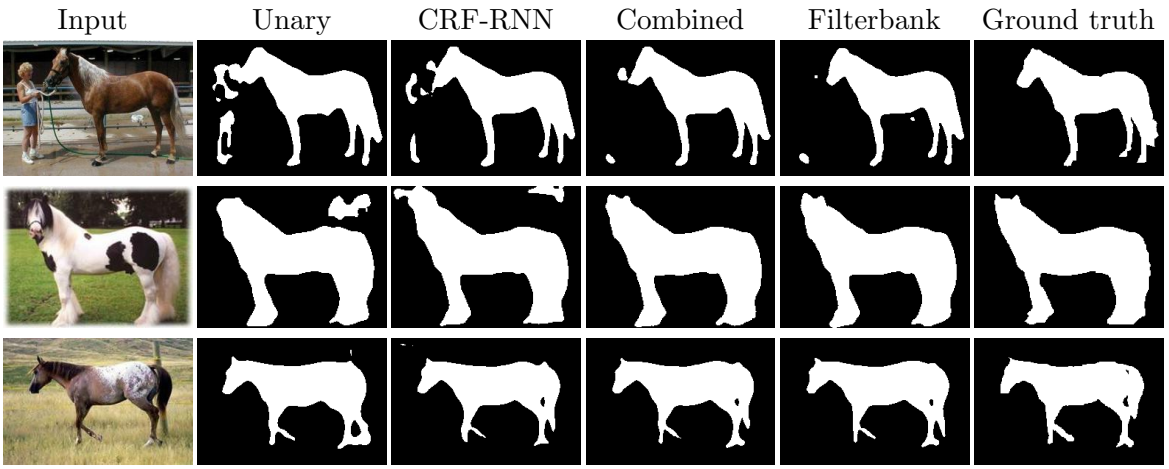

Figure 4. *Qualitative results on the* Weizmann Horse *dataset. Note that the proposed methods capture the shape of the horses better than the baselines, especially compared to the unary netwok.*

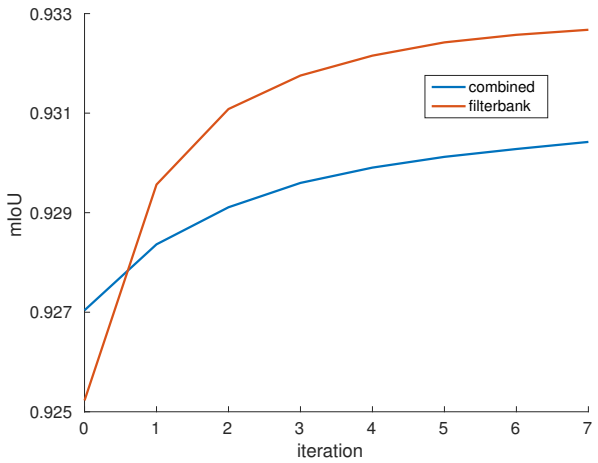

Figure 5. Weizmann Horse *test set results in terms of mean Intersection over Union plotted as a function of the number of iterations for the CRF inference method. During training the number of iterations were set to five.*

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 of iterations were set to five. As can be seen in the figure, increasing the number of inference step will only slightly increase

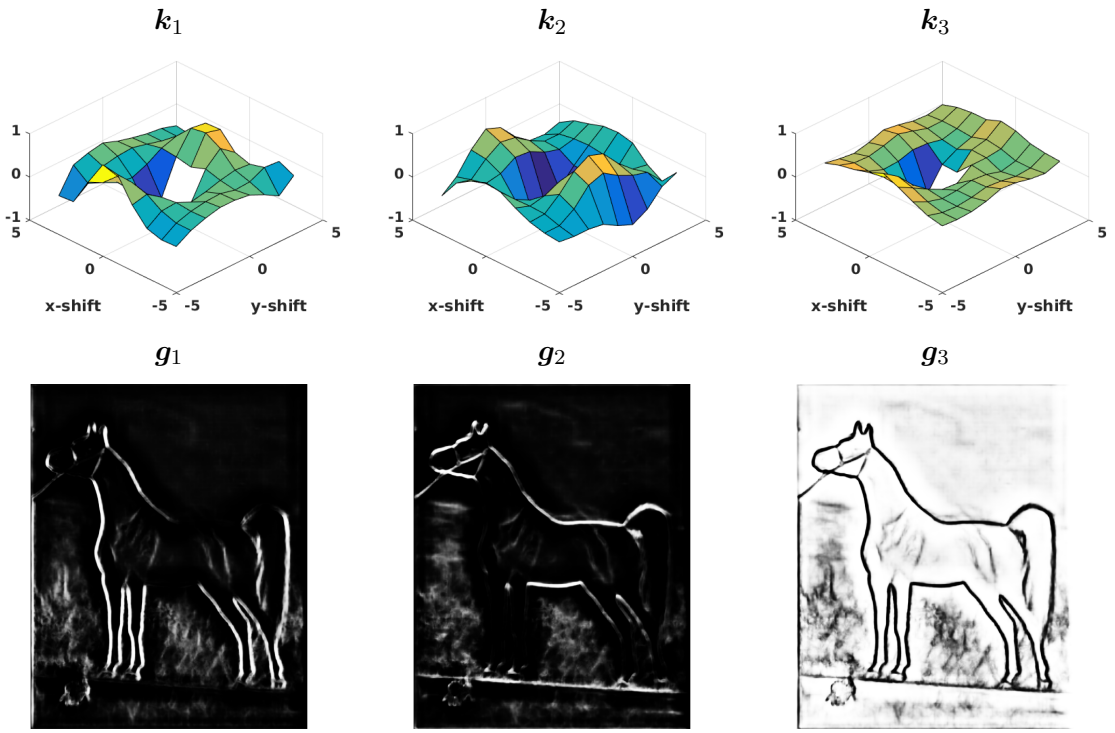

Figure 6. *Visualization of the pairwise kernel weights for the filterbank version trained on the* Weizmann Horse *data set. These weights are for the classes "background" and "horse", the plots can be understood as the energy added when assigning the pixels with the relative positions (x,y) and (x+x-shift,y+y-shift) as background and horse. This energy is then multiplied by the "filter selection"-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. Note that the middle position has been removed from the kernel plots since it does not provide the same structural information as the other weights and can hence not be interpreted in the same way.*

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.* [46], 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 [46] 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.

7.3. PASCAL VOC. The PASCAL VOC 2012 segmentation benchmark [22] consists of 20 foreground and one background class. The unary network used for these experiments is again

Method	mIoU (%)
R-CNN [25]	40.3
Semantic HCRF [46]	42.7
Joint HCRF [46]	44.2
Modular CNN [28]	54.3
Unary CNN - Deeplab [14]	62.8
CRF-RNN [48]	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.*

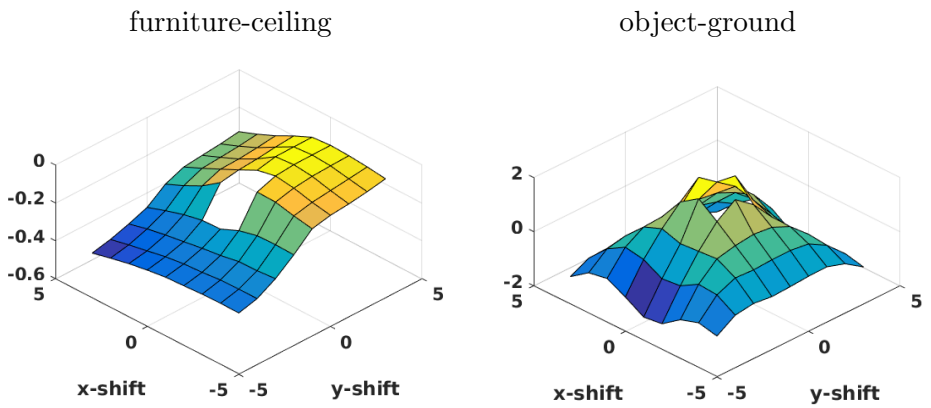

Figure 7. *Visualization of the pairwise kernel weights for the filterbank version trained on the* NYU V2 *data set. These weights are for the classes shown above the plots and for the third filter selection map which usually has a high value for pixels with no semantic edge. The furniture-ceiling kernel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

the Deeplab-LargeFOV network [18], this network has been pretrained on the MS-COCO 2014 dataset [38] and then trained on the PASCAL VOC training data as well as a training set created from annotations of the semantic boundaries dataset [26]. 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 been several CNNs with different base architectures presented that perform well, even without a CRF. The top entry at the moment is PSPNet [36] with a mIoU of 85.4, we leave it to future work to explore

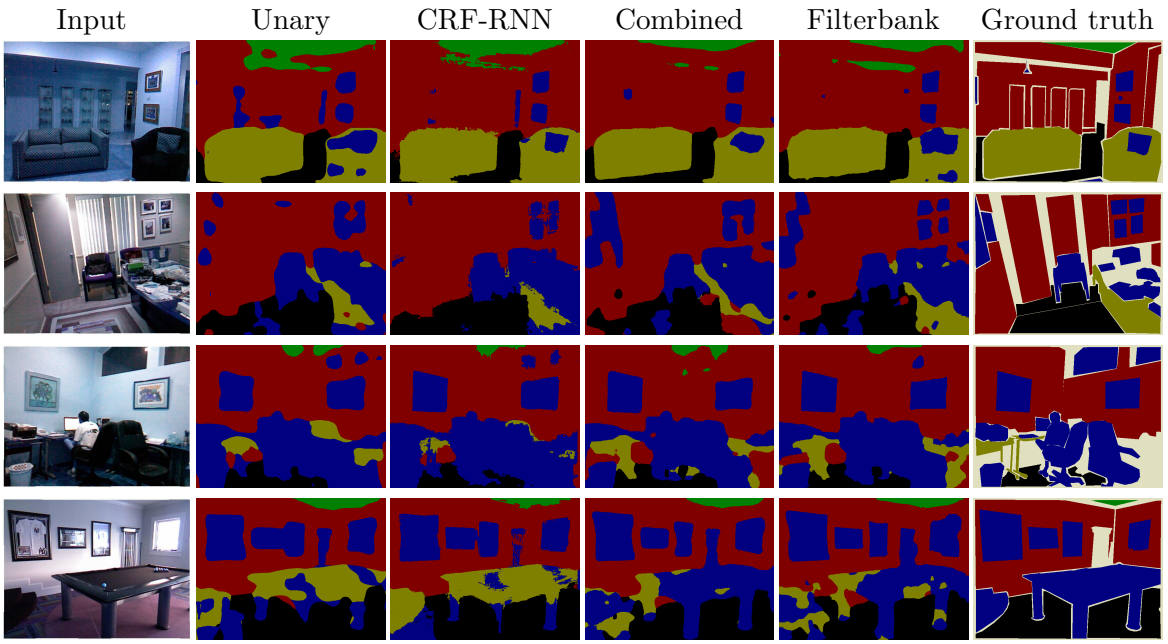

Figure 8. *Qualitative results on the* NYU V2 *dataset. Note that the proposed methods captures the shape of the object instances better than the baselines. This effect is perhaps most pronounced for the paintings hanging on the walls. The pixels colored off-white are "ignore"-pixels, these are not counted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

Method	mIoU (%)
Unary CNN - Deeplab [14]	68.5
CRF-RNN [48]	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 [38].

whether these architectures can be improved using our proposed methodology.

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.* [29] 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

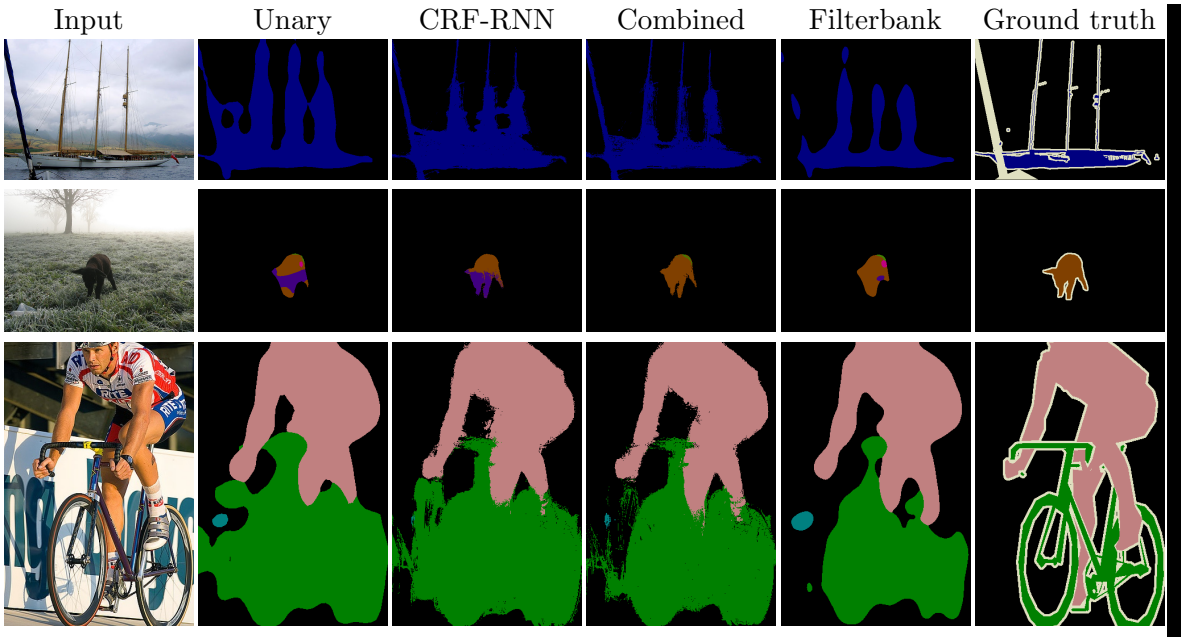

Figure 9. *Qualitative results on the* PASCAL VOC *dataset. [22]. The pixels colored off-white are "ignore"-pixels, these are not counted in the evaluation. The training images have similar "ignore"-pixels.*

Method	mIoU (%)
Unary CNN - Deeplab [14]	68.9
DT-EdgeNet [17]	71.7
CRF-RNN [48]	72.2
Combined	72.5
Filterbank	69.5

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 [38], but note that our models were not trained using MS-COCO.

second.

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. Since the *filterbank* version of the model learns how the pairwise term should depend on the image it is in this aspect more expressive than the *combined* version. The pairwise terms of the *combined* model is however, due to its dependence on the color gradient, hand-crafted to preserve and refine edges. This is beneficial for the PASCAL VOC dataset where the unary network generally capture the context well but outputs "blobby" segmentations. 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.

Acknowledgements. This work has been funded by the Swedish Research Council (grant no. 2016-04445), the Swedish Foundation for Strategic Research (Semantic Mapping and Visual Navigation for Smart Robots), Vinnova / FFI (Perceptron, grant no. 2017-01942), ERC grant ERC-2012-AdG 321162-HELIOS, EPSRC grant Seebibyte EP/M013774/1 and EPSRC/MURI grant EP/N019474/1.

REFERENCES

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

[2] T. Ajanthan, A. Desmaison, R. Bunel, M. Salzmann, P. H. Torr, and M. P. Kumar, *Efficient linear programming for dense crfs*.

[3] 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.

[4] 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, 35 (2018), pp. 37–52, https://doi.org/10.1109/MSP.2017.2762355.

[5] A. Beck and M. Teboulle, *Mirror descent and nonlinear projected subgradient methods for convex optimization*, Operations Research Letters, 31 (2003), pp. 167–175.

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

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

[8] 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.

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

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

[11] E. Boros and P. Hammer, *Pseudo-boolean optimization*, Discrete Appl. Math., 123 (2002), pp. 155–225.

[12] 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.

[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] 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.

[15] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*, arXiv preprint arXiv:1606.00915, (2016).

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

[17] 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.

[18] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2017).

[19] Y. Chen and X. Ye, *Projection onto a simplex*, arXiv preprint arXiv:1101.6081, (2011).

[20] 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.

[21] J. Domke, *Learning graphical model parameters with approximate marginal inference*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. 2454–2467.

[22] 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, 111 (2015), pp. 98–136, https://doi.org/10.1007/s11263-014-0733-5, https://doi.org/10.1007/s11263-014-0733-5.

[23] 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, 88 (2010), pp. 303–338.

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

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

[26] B. Hariharan, P. Arbelez, L. Bourdev, S. Maji, and J. Malik, *Semantic contours from inverse detectors*, in 2011 International Conference on Computer Vision, Nov 2011, pp. 991–998, https://doi.org/10.1109/ICCV.2011.6126343.

[27] 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.

[28] 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.

[29] 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, June 2016.

[30] 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).

[31] 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.

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

[33] 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.

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

[35] 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.

[36] 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).

[37] 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, June 2016.

[38] 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*, Springer International Publishing, Cham, 2014, pp. 740–755, https://doi.org/10.1007/978-3-319-10602-1_48, https://doi.org/10.1007/978-3-319-10602-1_48.

[39] 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.

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

[41] 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.

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

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

[44] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, in

International Conference on Learning Representations, 2015.
[45] 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.
[46] 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.
[47] W. Wang, S. Fidler, and R. Urtasun, *Proximal deep structured models*, in Neural Information Processing Systems, 2016.
[48] 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.