A Projected Gradient Descent Method for CRF Inference allowing End-To-End Training of Arbitrary Pairwise Potentials

Måns Larsson[1], Anurag Arnab[2], Fredrik Kahl[1,3], Shuai Zheng[2], and Philip Torr[2]

[1] Chalmers University of Technology, Gothenburg, Sweden
[2] University of Oxford, Oxford, England
[3] Centre for Mathematical Sciences, Lund University, Lund, Sweden

Abstract. Are we using the right potential functions in the Conditional Random Field models that are popular in the Vision community? Semantic segmentation and other pixel-level labelling tasks have made significant progress recently due to the deep learning paradigm. However, most 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.

In this paper, we challenge this view by developing a new inference and learning framework which can learn pairwise CRF potentials restricted only by their dependence on the image pixel values and the size of the support. Both standard spatial and high-dimensional bilateral kernels are considered. Our framework is based on the observation that CRF inference can be achieved via projected gradient descent and consequently, can easily be integrated in deep neural networks to allow for end-to-end training. 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. In addition, we compare our inference method to the commonly used mean-field algorithm. Our framework is evaluated on several public benchmarks for semantic segmentation with improved performance compared to previous state-of-the-art CNN+CRF models.

Keywords: Conditional Random Fields, Segmentation, Convolutional Neural Networks

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 [23, 4]. In this paper, we focus on the application of CRFs 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 [31] using graph cuts and the semantic segmentation works of [25, 35] where the parallel mean-field inference algorithm was applied for fast inference. Recently, Convolutional Neural Networks
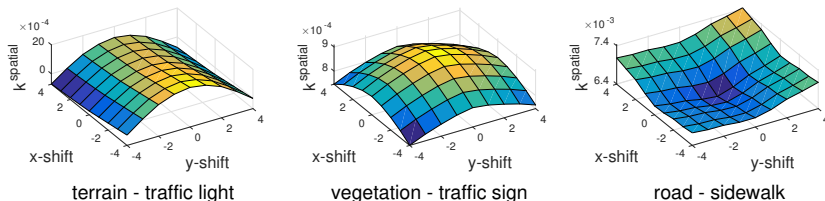
Fig. 1. Learned spatially invariant CRF filters for CITYSCAPES. These filters model contextual relationships between classes and their values can be understood as the energy added when setting one pixel to the first class (e.g., vegetation) and the other pixel with relative position (x-shift,y-shift) to the second class (e.g., traffic sign). Note how the terrain-traffic light filter favours vertical edges. In addition, the model can learn filters of different shapes which is shown by the road-sidewalk filter.

(CNNs) have dominated the field in a variety of recognition tasks [18, 33, 30]. However, we observe that leading segmentation approaches still include CRFs, either as a post-processing step [10, 16, 9, 11], or as part of the deep neural network itself [38, 26, 2, 27, 22].

We leverage on the idea of embedding inference of graphical models into a neural network. An early example of this idea was presented in [7] where the authors back propagated through the Viterbi algorithm when designing a document recognition system. Similar to [38, 2, 3, 37], we use a recurrent neural network to unroll the iterative inference steps of a CRF. This was first used in [38] and [32] to imitate mean-field inference and to train a fully convolutional network [28, 10] along with a CRF end-to-end via back propagation. In contrast to mean-field, we do not optimize the KL-divergence. Instead, we use a gradient descent approach for the inference that directly minimises the Gibbs energy of the random field and hence avoids the approximations of mean-field. A similar framework was recently suggested in [3] for multi-label classification problems in machine learning with impressive results. However, [3] uses a Structured SVM approach for training whereas we do back propagation through the actual steps of the gradient descent method. Moreover, [15] have recently shown that one can obtain lower energies compared to mean-field inference using gradient descent based optimization schemes. Still, we lack formal algorithmic guarantees of the solution quality compared to, e.g., graph cuts [8].

In many works, the pairwise potentials consist of parametrized Gaussians [24, 38, 2] and it is only the parameters of this Gaussian which are learned. Our framework can learn arbitrary pairwise potentials which need not be Gaussian, cf. Fig. 1. An early work which learned potentials of a linear chain CRF for sequence modelling is [29]. In [12], a general framework for learning arbitrary potentials in deep structured models was proposed based on approximate Maximum Likelihood 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. The experiments of [12] and [3] are limited to multi-label classification where there is no spatial relationship between different labels as in pixel-labelling tasks. Morever, [22] have also observed that the memory requirements of the method of [12] renders it infeasible for

2

the large datasets common in Computer Vision. Another approach to learning arbitrary pairwise potentials was presented in [22] 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 [26] and [9] 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 [20], a pairwise potential is learned based on sparse bilateral filtering. Applying such a filter can be regarded as one iteration of mean field CRF inference. Contrasting [20], 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.

In summary, our contributions are as follows.

– Our main contribution is a new framework for non-parametric CRF inference and learning which is integrated with standard CNNs. During inference, we directly minimize the CRF energy using gradient descent and during training, we back propagate through the gradient descent steps for end-to-end learning.
– We analyze the learned filter kernels empirically and demonstrate that in many cases it is advantageous with non-Gaussian potentials.
– We experimentally compare our approach to several leading methodologies, e.g., [38, 16, 28] and improve on state of the art on two public benchmarks: NYU V2 [36] and CITYSCAPES [14].

Our framework has been implemented in both CAFFE [21] and MATCONVNET [34], and all source code will be made publicly available to facilitate further research.

2 CRF Formulation

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

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

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

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

where $\psi_i : \mathcal{L} \to \mathbb{R}$ and $\psi_{ij} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ are the unary and pairwise potentials, respectively.

2.1 Potentials

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

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

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 data into account. We consider the following set of pairwise potentials

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

Here $k_{x_i,x_j}^{spatial}$ denotes 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, cf. Fig. 1. However, we restrict the support of the potential meaning that if pixels i and j are far apart, then the value of $k_{x_i,x_j}^{spatial}(\boldsymbol{p}_i - \boldsymbol{p}_j)$ will be zero. 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 and are commonly truncated to 0 after two standard deviations.

The term $k_{x_i,x_j}^{bilateral}$ 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 [24, 38, 11], 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 to learn differently shaped kernels for each of these interactions.

2.2 Multi-label Graph Expansion and Relaxation

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. 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

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{V}, \lambda \in \mathcal{L}} \psi_i(\lambda) q_{i:\lambda} + \sum_{\substack{(i,j) \in \mathcal{E} \\ \lambda, \mu \in \mathcal{L}}} \psi_{ij}(\lambda, \mu) q_{i:\lambda} q_{j:\mu} \\
\text{s.t.} \quad & q_{i:\lambda} \geq 0 && \forall i \in \mathcal{V}, \lambda \in \mathcal{L} \\
& \sum_{\lambda \in \mathcal{L}} q_{i:\lambda} = 1 && \forall i \in \mathcal{V}.
\end{aligned} \qquad (5)$$

4

Note that the added constraints ensure that our solution lies on the probability simplex. A natural question is what happens when the domain is enlarged, allowing real values instead of booleans. Somewhat surprisingly, the relaxation is tight [6].

Proposition 1 *Let $E(\boldsymbol{x}^*)$ and $E(\boldsymbol{q}^*)$ denote the optimal values of (5), where $\boldsymbol{x}^*$ is restricted to boolean values. 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.

In summary, it has been shown that to minimize the energy function $E(\boldsymbol{x})$ over $\boldsymbol{x} \in \mathcal{L}^N$, one may work in the continuous domain, minimize over $\boldsymbol{q}$, and then replace any solution $\boldsymbol{q}$ by a discrete solution $\boldsymbol{x}$ which has lower or equal energy. It will only be possible to find a local solution $\boldsymbol{q}$, but still the discrete solution $\boldsymbol{x}$ will be no worse than $\boldsymbol{q}$.

3   Minimization with Gradient Descent

To solve the program stated in (5) we propose an optimization scheme based on projected gradient descent, see Algorithm 1. It was designed with an extra condition in mind, that all operations should be differentiable to enable back propagation during training.

Initialize $\boldsymbol{q}^0$
for t *from* 0 *to* $T-1$ **do**
$\quad$ Compute the gradient $\nabla_{\boldsymbol{q}} E(\boldsymbol{q}^t)$.
$\quad$ Take a step in the negative direction, $\tilde{\boldsymbol{q}}^{t+1} = \mathbf{q}^t - \gamma \, \nabla_{\mathbf{q}} \boldsymbol{E}$.
$\quad$ Project $\tilde{q}_{i:\lambda}^{t+1}$ to the simplex $\triangle^L$ satisfying $\sum_{\lambda \in \mathcal{L}} \tilde{q}_{i:\lambda} = 1$ and $0 \leq \tilde{q}_{i:\lambda} \leq 1$,
$\quad\quad \boldsymbol{q}^{t+1} = \text{Proj}_{\triangle^L}(\tilde{\boldsymbol{q}})$.
end
Output: $\boldsymbol{q}^{T-1}$

Algorithm 1: Algorithm 1. Projected gradient descent algorithm.

The gradient $\nabla_{\boldsymbol{q}} E$ of the objective function $E(\boldsymbol{q})$ in (5) has the following elements

$$\frac{\partial E}{\partial q_{i:\lambda}} = \psi_i(\lambda) + \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} \psi_{ij}(\lambda,\mu) q_{j:\mu}. \tag{6}$$

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

$$v_{i:\lambda}^{spatial} = \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} k_{\lambda,\mu}^{spatial}(\boldsymbol{p}_i - \boldsymbol{p}_j) q_{j:\mu}. \tag{7}$$

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

$$v_{i:\lambda}^{bilateral} = \sum_{\substack{j:(i,j)\in\mathcal{E} \\ \mu\in\mathcal{L}}} k_{\lambda,\mu}^{bilateral}(\boldsymbol{f}_i - \boldsymbol{f}_j)q_{j:\mu}. \tag{8}$$

For this computation we utilize the method presented by Jampani *et al.* [20] 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.

Next, we simply take a step in the negative direction of the gradient according to

$$\tilde{\boldsymbol{q}}^{t+1} = \mathbf{q^t} - \gamma \nabla_{\mathbf{q}}\boldsymbol{E}, \tag{9}$$

where γ is the the step size.

Finally, we want to project our values onto the simplex $\triangle^L$ satisfying $\sum_{\lambda\in\mathcal{L}} q_{i:\lambda} = 1$ and $0 \leq q_{i:\lambda} \leq 1$. This is done following the method by Chen *et al.* [13] that efficiently calculates the euclidean projection on the probability simplex $\triangle^L$, for details see the supplementary materials. Note that this projection is done individually for each pixel i.

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 [25]. As we have seen in this section, it can be accomplished with our framework as well. The main difference is that our framework directly optimizes the Gibbs energy which corresponds to the MAP solution while mean-field optimizes KL-divergence which does not.

4   Integration in a Deep Learning Framework

In this section we will describe how the gradient descent steps of Algorithm 1 can be formulated as layers in a neural network. 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. This will enable us to unroll the entire gradient descent process as a Recurrent Neural Network (RNN). A schematic of the data flow for one step is shown in Fig. 2. In the supplementary material, all derivative formulae are given.

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 at the end to ensure that the variables lies within zero and one as well as sum to one for each pixel.

$$\mathbf{q}^{t+1} = f(\mathbf{q}^t, \mathbf{z}, I, \mathbf{w})$$
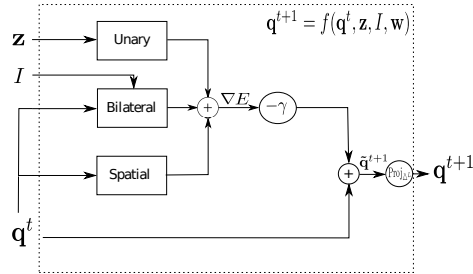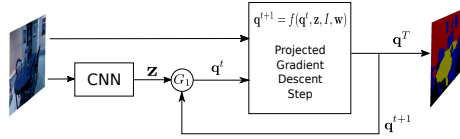
Fig. 2. The data flow of one iteration of the projected gradient descent algorithm. Each rectangle or circle represent an operation that can be performed within a deep learning framework.

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 three terms, the unary, spatial and bilateral pairwise terms. We can hence treat these three terms separately and combine them using element-wise summing.

The unary term in (3) is an element-wise 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 element-wise. 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.* [20] also presented a way to calculate the error derivatives with respect to the parameters for an arbitrary shaped bilateral filter.

Gradient Step. Taking a step in the negative direction of the gradient is easily incorporated in a deep learning framework by using an element-wise summing layer. The layer takes the variables $\boldsymbol{q}^t$ as the first input and the gradient (scaled by $-\gamma$) as the second input.

Simplex Projection As a final step, the variables from the gradient step $\tilde{\boldsymbol{q}}^t$ are projected onto the simplex $\triangle^L$. In reality we use a leaky version of the last step of the projection algorithm to avoid error derivatives becoming zero during back propagation. Since the projection is done individually for each pixel it can be described as a function $\boldsymbol{f}(\tilde{\boldsymbol{q}}) : \mathbb{R}^L \to \mathbb{R}^L$ of which we can calculate the Jacobian, see supplementary materials. Knowing the Jacobian, the error derivatives with respect to the input can be computed during back propagation.

Fig. 3. The data flow of the deep structured model. Each rectangle or circle represent an operation that can be performed within a deep learning framework.

5 Recurrent Formulation as a 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 a CNN, as shown in Fig. 3. The CNN's output, $\mathbf{z}$, are the unary potentials and obtained after the final softmax layer (since the CNN is initially trained for classification).

Each iteration of the RNN performs one projected gradient descent step to approximately solve (5). Thus, one update step can be represented by:

$$\boldsymbol{q}^{t+1} = f(\boldsymbol{q}^t, \boldsymbol{z}, I, \boldsymbol{w}). \tag{10}$$

As illustrated in Fig. 3, the gating function G_1 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 formulated as an RNN, 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 $T = 5$ iterations.

The parameters of the RNN are the filter weights for the spatial and bilateral 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 back propagate 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 efficiently in a fully-convolutional manner.

Implementation Details. Our proposed CRF model has been implemented in the CAFFE [21] library, and also has a Matlab wrapper allowing it to be used in MATCONVNET [34]. The Unary CNN part of our model is initialised from a pre-trained segmentation network. As we show in our experiments, different pre-trained networks for different applications can be used.

The CRF model has several tunable parameters. The step size γ and the number of iterations T specify the properties of the gradient decent algorithm. Too high a step size γ 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 spatial and bilateral kernels also need to be set. Choosing the value

Input	z_{horse}	q^1_{horse}	q^2_{horse}	q^3_{horse}	q^4_{horse}	q^5_{horse}	Ground truth
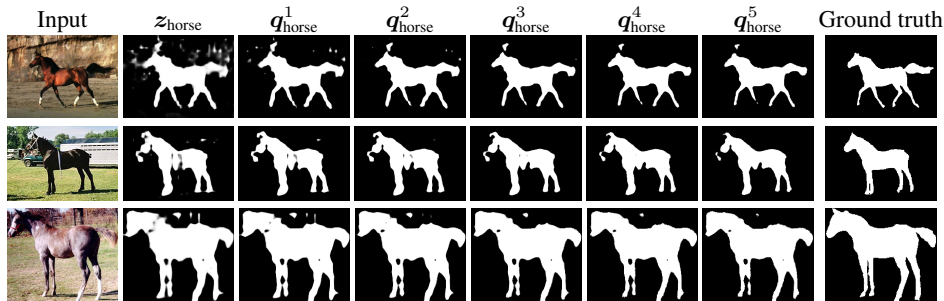
Fig. 4. Visualization of intermediate states of the CRF-Grad layer for the WEIZMANN HORSE dataset. Note how each step of the gradient descent algorithm refines the segmentation slightly, removing spurious outlier pixels classified as horse. z_{horse} is the CNN output.

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) as done in [25, 10, 38]. Note that unlike [25, 10] we are not limited to only Potts class interactions.

6 Experiments

We evaluate the proposed approach on three datasets: WEIZMANN HORSE [5], NYU V2 [36] and CITYSCAPES [14]. In these experiments, we show that the proposed approach, denoted CRF-Grad, has advantages over baseline approaches such as CRFas-RNN [38] and complement other networks such as FCN-8s [28] and LRR [16]. In addition we compare our inference method to mean-field inference. Hyperparameter values not specified in this section can be found in the supplementary materials.

6.1 Weizmann Horse

The WEIZMANN HORSE 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 purpose is to verify our ability to learn reasonable kernels and study the effects of different settings on a relatively small dataset.

The CNN part of our model was initialized as an FCN-8s network [28] pre-trained without the CRF layer. We then compare several variants of our model. We start off by training a variant of our CRF model only using the 2D spatial kernel. We compare these results to using a Gaussian spatial filter, where the parameters for the Gaussian kernel were evaluated using cross-validation. In addition we train the full model with both the spatial and bilateral kernels, once keeping the filters fixed as Gaussians and once allowing arbitrarily shaped filters.

The mean Intersection over Union (IoU) on the test set, for different configurations of our model, is shown in Table 1. Our results show that allowing arbitrarily shaped filters gives better performance than keeping the filters fixed as Gaussian. This is the case for both the spatial and bilateral kernels. Also, adding bilateral filters improves the results compared to using spatial filters only. In addition the model trained with our inference method performs better than the model (with the same unary and pairwise potentials) trained with mean-field inference.

| Weizmann Horse		NYU V2		Cityscapes	
Method	IoU (%)	Method	IoU (%)	Method	IoU (%)
FCN-8s (only)	80.0	R-CNN [17]	40.3	CRFasRNN[*] [38]	62.5
FCN-8s + spatial (G)	81.3	Joint HCRF [36]	44.2	Deeplabv2-CRF [11]	70.4
FCN-8s + spatial	82.0	Modular CNN [19]	54.3	Adelaide_context [26]	71.6
FCN-8s + full (G)	82.9	CRFasRNN [38]	54.4	LRR-4x [16]	71.8
FCN-8s + full	**84.0**	CRF-Grad (Ours)	**55.0**	CRF-Grad (Ours)	**71.9**
FCN-8s + full (MF)	83.3				

[*]Note that CRFasRNN uses a different CNN model than ours on the Cityscapes dataset.

Table 1. Quantitative results comparing our method to baselines as well as state-of-the-art methods. Mean intersection over union for the test set is shown for Weizmann Horse and Cityscapes, for NYU V2 there is no test set and validation score is presented. In the entry denoted "full" the complete CRF-Grad layer was used, while in "spatial" no bilateral kernel was used. A (G) means that the filters were restricted to a Gaussian shape and (MF) means that the inference method was switched to mean field. For the entries denoted CRF-Grad the full model was used.

In Fig. 4 the intermediate states of the layer ($\mathbf{q}^t$ for each gradient descent step) are shown. Fig. 5 presents a convergence analysis for the full version of our layer as well as a comparison to mean-field inference. The results show that the CRF energy converges in only a few iteration steps and that increasing the number of iterations barely affects performance after $T = 6$ iterations. It also shows that our inference method achieves lower Gibbs energy as well as higher IoU results compared to mean-field, even for models trained with mean-field inference. Looking at using mean field inference for the model trained with projected gradient descent we see that the energy increases for each iteration. The reason for this is that the mean-field method doesn't actually minimize the energy but the Kullback-Liebler divergence between the Gibbs distribution and the solution. This is, as shown in the supplementary materials, equivalent to minimizing the energy minus the entropy. The extra entropy term favors "uncertain" solutions, i.e. solution with values not close to zero or one, which increases the energy for some models.

6.2 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
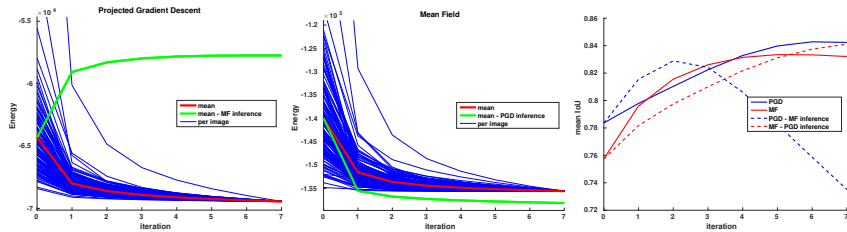
Fig. 5. Left and Middle: CRF energy, as defined in (5), as a function of inference iterations. The thin blue lines show the different instances, the thicker red line show the mean while the green line shows the mean when the inference method has been switched. The model has been trained with the inference method shown in respective title. For these calculations the leak factor was set to zero, meaning that the solutions satisfy the constraints of (5). Note that, for presentation purposes, all energies have been normalized to have the same final energy. Right: Mean IoU as a function of iterations for the two different models and inference methods. The number of iterations were set to five during training. All results are from the test set of the WEIZMANN HORSE data set.

654 images, respectively. Following the setting described in Wang *et al.* [36], 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 [36] we consider 5 classes conveying strong geometric properties: ground, vertical, ceiling, furniture and objects. The CNN part of our model was initialized as the fully convolutional network FCN-8s [28] pre-trained on the data. Afterwards we added our CRF-Grad layer and trained the model end-to-end.

As shown in Table 1, we achieved superior results for semantic image segmentation on the NYU V2 dataset. Some example segmentations are shown in Fig. 6. Additional examples are included in the supplementary material.

6.3 Cityscapes

The CITYSCAPES dataset [14] consists of a set of images of street scenes collected from 50 different cities. The images are high resolution (1024×2048) and are paired with pixel-level annotations of 19 classes including road, sidewalk, traffic sign, pole, building, vegetation and sky. The training, validation and test sets consist of 2975, 500 and 1525 images, respectively. In addition there are 20000 coarsely annotated images that can be used for training. The CNN part of our model was initialized as an LRR network [16] pre-trained on both the fine and the coarse annotations. We then added our CRF-Grad layer and trained the model end-to-end on the finely annotated images only.

In Table 1 the results of evaluating our model on the test set are compared to current state of the art. As can be seen, our model is on par although the improvement upon LRR is minor. An interesting aspect of CITYSCAPES is that it contains classes of thin and vertical objects, e.g., traffic light and pole. What we noticed is that the spatial filters for these classes usually get a more oblong shape. This type of pairwise filters does not add as much energy for switching classes going in the horizontal direction, favoring

11

| Input | FCN-8s | CRF-RNN | CRF-Grad | Ground truth |
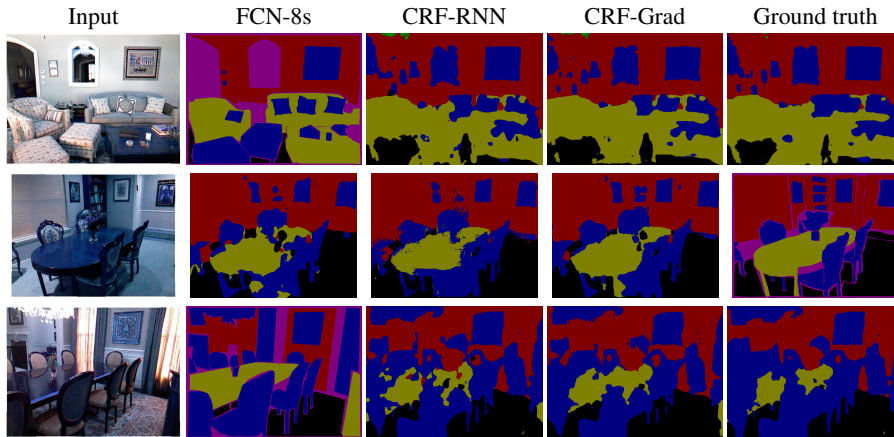
Fig. 6. Qualitative results on the NYU V2 dataset. Note that the CRF-Grad captures the shape of the object instances better compared to the baselines. This effect is perhaps most pronounced for the paintings hanging on the walls.
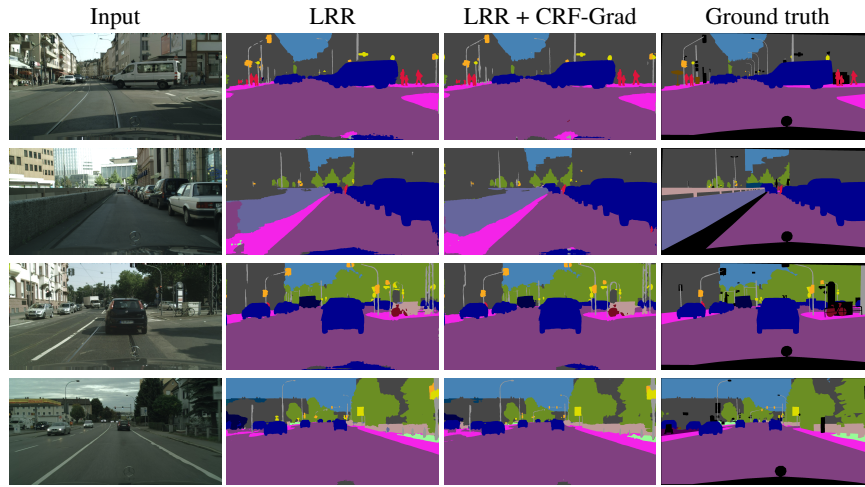
vertically elongated segmentations. This can be seen in the spatial filter for the class interaction between "terrain" and "traffic light" in Fig. 1. Some example segmentations are shown in Fig. 7. Additional examples as well as class-wise results are included in the supplementary material. Our method got better, or equal, results for 14 of the 19 classes compared to the baseline.

7 Concluding Remarks

In this paper, we have introduced a new framework capable of learning arbitrarily shaped pairwise potentials in random fields models. In a number of experiments, we have empirically demonstrated that our developed framework can improve state-of-the-art CNNs by adding a CRF layer. We have also seen that the learned filters are not necessarily Gaussian, and may capture other kinds of interactions between labels. In addition, we have shown advantages of our inference method compared to the commonly used mean-field method.

A key factor for the success of deep learning and by now a well-established paradigm is that the power of convolutions should be used, especially for the first layers in a CNN. Our work supports that repeated usage of convolutions in the final layers is also beneficial. We also note that our gradient descent steps resemble the highly successful RESNET [18], as one step in gradient descent is, in principle, an identity transformation plus a correction term.

There are several future research avenues that we intend to explore. In our model, many free variables are introduced and this may lead to over-fitting. One way to compensate would be to collect larger datasets and consider data augmentation. An alternative approach would be to directly encode geometric shape priors into the random fields and thereby reducing the required amount of data.

Fig. 7. Qualitative results on the CITYSCAPES validation set. Black regions in the ground truth are ignored during evaluation. Our CRF models contextual relationships between classes, hence unlike LRR, it does not label "road" as being on top of "sidewalk" (Row 2). Note that the traffic lights are better segmented with the additional CRF-Grad layer. Adding the CRF-Grad layer increased the IoU of the class traffic lights from 66.8 to 68.1.

References

1. A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 2010.
2. 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.
3. D. Belanger and A. McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, 2016.
4. A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.
5. E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *European Conference on Computer Vision*, 2002.
6. E. Boros and P.L. Hammer. Pseudo-boolean optimization. *Discrete Appl. Math.*, 123:155–225, 2002.
7. 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*, pages 489–494. IEEE, 1997.
8. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

9. 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.

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

11. 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.

12. L.C. Chen, A.G. Schwing, A.L. Yuille, and R. Urtasun. Learning deep structured models. In *Int. Conf. Machine Learning*, Lille, France, 2015.

13. Y. Chen and X. Ye. Projection onto a simplex. *arXiv preprint arXiv:1101.6081*, 2011.

14. M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

15. 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.

16. G. Ghiasi and C. Fowlkes. Laplacian reconstruction and refinement for semantic segmentation. In *European Conference on Computer Vision*, 2016.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

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

23. D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

24. P. Kraehenbuehl and V. Koltun. Parameter learning and convergent inference for dense random fields. In *Proceedings of The 30th International Conference on Machine Learning*, pages 513–521, 2013.

25. P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *Neural Information Processing Systems*. 2011.

26. 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.

27. 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.

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

29. J. Peng, L. Bo, and J. Xu. Conditional neural fields. In *Advances in neural information processing systems*, pages 1419–1427, 2009.

[30] 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.

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

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

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

[34] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM International Conference on Multimedia*, 2015.

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

[36] 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.

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

[38] 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.

Supplementary Material

Proof of Proposition 1

Let $E(\boldsymbol{x}^*)$ and $E(\boldsymbol{q}^*)$ denote the optimal values of (5), where $\boldsymbol{x}^*$ is restricted to boolean values. Then,

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

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

Let $\boldsymbol{q}$ be given, and let $\boldsymbol{x} \in \mathcal{L}^N$. One may define $E^m(\boldsymbol{x}, \boldsymbol{q}) = E(x_1, \ldots, x_m, q_{m+1}, \ldots, q_N)$ such that each x_i or q_i is a vector with entries such as $q_{i:\lambda}$ or $x_{i:\lambda}$, but for each i only one value $x_{i:\lambda}$ is non-zero (and equal to 1). Since $E^0 = E(\boldsymbol{q})$ and $E^N = E(\boldsymbol{x})$ it will be sufficient to find a $\boldsymbol{x}$ such that $E^m(\boldsymbol{x}, \boldsymbol{q}) \leq E^{m-1}(\boldsymbol{x}, \boldsymbol{q})$ for all m. The required $\boldsymbol{x}$ will be constructed one element at a time.

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

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

Here, x_m is treated as a variable and $x_1, \ldots, x_{m-1}$ are fixed. Since $\sum_{x_m \in \mathcal{L}} q_{m:x_m} = 1$ there must be at least one choice of x_m such that $E^{m-1}(\boldsymbol{x}, \boldsymbol{q}) \geq E(x_1, \ldots, x_m, q_{m+1}, \ldots q_m) = E^m(\boldsymbol{x}, \boldsymbol{q})$.

Mean-Field Objective

In this section, we give a brief derivation of what is minimized with the mean-field method. The key idea behind the method is to approximate a complex probability distribution P by a simpler one Q that one can solve (find its mode), using the simpler distribution as a stand-in for the actual probability distribution (MRF) of interest. As previously, we consider a probability distribution P for a random field, given by the Gibbs distribution. The probability of an assignment $\boldsymbol{x} \in \mathcal{L}^N$ is

$$P(\boldsymbol{x}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{x})\right). \tag{11}$$

A natural measure commonly used as a measure of distance between two probability distributions P and Q is the KL divergence $D(Q \parallel P)$ defined by

$$\begin{aligned}
D(Q \parallel P) &= \sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) \log\left(\frac{Q(\boldsymbol{x})}{P(\boldsymbol{x})}\right) \\
&= -\sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) \log P(\boldsymbol{x}) + \sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) \log Q(\boldsymbol{x}).
\end{aligned} \tag{12}$$

Now, plugging in the form of the probability $P(\boldsymbol{x})$ given in (11) into (12), we obtain $D(Q \,\|\, P) =$

$$- \sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) \log \left(\frac{1}{Z} \exp\left(-E(\boldsymbol{x})\right) \right) + \sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) \log Q(\boldsymbol{x})$$

$$= \sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) E(\boldsymbol{x}) + \log Z + \sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) \log Q(\boldsymbol{x}),$$

where we have used the fact that $\sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) = 1$.

Because $\log Z$ is a constant, minimizing the KL divergence between these two distributions is equivalent to minimizing the following quantity:

$$\sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) E(\boldsymbol{x}) + \sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) \log Q(\boldsymbol{x}). \tag{13}$$

The first term is equal to the expected value of the energy $E(\boldsymbol{x})$ under the probability distribution Q; for now we will write this as $\mathcal{E}_Q[E]$. Now, consider the second term. The entropy of a probability distribution Q is equal to $-\sum_{\boldsymbol{x} \in \mathcal{L}^N} Q(\boldsymbol{x}) \log Q(\boldsymbol{x})$. Writing the entropy of Q as $H(Q)$, (13) takes the form

$$\mathcal{E}_Q[E] - H(Q), \tag{14}$$

so the quantity is equal to the expectation of the Gibbs energy under distribution Q minus the entropy of Q.

Hyperparameter Settings

Weizmann Horse

All the models presented in the experiments section were trained end-to-end with learning rate of 10^{-3} (normalized by the number of pixels), momentum 0.9, weight decay $5 \cdot 10^{-3}$ and batch size 20. The size of the spatial filters for these runs were 9×9 if not specified as other. The number of iterations were set to 5, step size to 0.5 and unary weight was initialized as 0.5. For the models with bilateral filter s, the number of neighbours each pixel interacts with in each dimension, were set to 1.

NYU V2

The model used on the NYU V2 data set was trained with learning rate 10^{-11} (not normalized), momentum 0.99, weight decay 0.005 and batch size 10. The size of the spatial filters were set to 9×9 and the bilateral filters to $s = 1$. The number of iterations were set to 5, step size to 0.5 and unary weight was initialized as 0.5.

Cityscapes

The model used on the CITYSCAPES data set was trained with a learning rate of 10^{-3} (normalized by the number of pixels), momentum 0.9 and weight decay $5 \cdot 10^{-3}$. The size of the spatial filters for these runs were 9×9 and the bilateral filters to $s = 1$. The number of iterations were set to 5, step size to 0.5 and unary weight was initialized as 0.5.

Simplex Projection

As mentioned in the paper we want to project our values onto the simplex $\triangle^L$ satisfying $\sum_{\lambda \in \mathcal{L}} q_{i:\lambda} = 1$ and $0 \leq q_{i:\lambda} \leq 1$ for each step in our gradient descent algorithm. This is done following the method presented by Chen *et al.* [13] which is summarized in Algorithm 2. Note that this projection is done individually for each pixel i, for derivation of this algorithm we refer to the article.

---

1. Sort $\tilde{q}_i \in \mathbb{R}^L$ in ascending order and set k = L - 1
2. Compute $t_i = \frac{\sum_{j=k+1}^{L} \tilde{q}_{i:j} - 1}{L-k}$, If $t_i \geq \tilde{q}_{i:k}$ set $\hat{t} = t_i$ and go to step 4, otherwise set $k \leftarrow k - 1$ and redo step 2, if $k = 0$ go to step 3
3. Set $\hat{t} = \frac{\sum_{j=1}^{L} \tilde{q}_{i:j} - 1}{L}$
4. Return $\boldsymbol{q}_i$, where $q_{i:\lambda} = \max\left(\tilde{q}_{i:\lambda} - \hat{t}, 0\right), \lambda \in \mathcal{L}$

Algorithm 2: Algorithm 2. Projection of $\tilde{\boldsymbol{q}}_i \in \mathbb{R}^L$ onto the simplex $\triangle^L$ satisfying $\sum_{\lambda \in L} q_{i:\lambda} = 1$ and $0 \leq q_{i:\lambda} \leq 1$.

In reality we use a leaky version of the last step of the projection algorithm, *i.e.* instead of the $\max(\cdot, 0)$ operator we use the following function

$$f_\alpha(\tilde{q}_{i:\lambda}) = \begin{cases} \tilde{q}_{i:\lambda} - \hat{t} & 0 \leq \tilde{q}_{i:\lambda} - \hat{t} \\ \alpha\left(\tilde{q}_{i:\lambda} - \hat{t}\right) & \tilde{q}_{i:\lambda} - \hat{t} < 0 \end{cases} \tag{15}$$

where α is a parameter controlling the level of leakage. Note that for $\alpha = 0$ we get the the strict $\max(\cdot, 0)$ operator. As previously mentioned, the projection is done individually for each pixel. It can be described as a function $\boldsymbol{f}(\tilde{\boldsymbol{q}}) : \mathbb{R}^L \to \mathbb{R}^L$, which Jacobian has the elements

$$\frac{\partial f_\lambda}{\partial \tilde{q}_\mu} = \begin{cases} \begin{cases} \alpha\left(1 - \frac{\partial \hat{t}}{\partial \tilde{q}_\mu}\right) & \tilde{q}_\mu - \hat{t} < 0 \\ 1 - \frac{\partial \hat{t}}{\partial \tilde{q}_\mu} & \tilde{q}_\mu - \hat{t} \geq 0 \end{cases} & \mu = \lambda \\ \begin{cases} -\alpha\frac{\partial \hat{t}}{\partial \tilde{q}_\mu} & \tilde{q}_\mu - \hat{t} < 0 \\ -\frac{\partial \hat{t}}{\partial \tilde{q}_\mu} & \tilde{q}_\mu - \hat{t} \geq 0 \end{cases} & \mu \neq \lambda \end{cases} \tag{16}$$

where $\frac{\partial \hat{t}}{\partial \tilde{q}_\mu} = \frac{1}{L-k}$ if $\tilde{q}_\mu > \hat{t}$ and 0 otherwise (L and k defined as in Algorithm 2). Knowing the Jacobian, the error derivatives with respect to the input can be computed during back propagation. The reason for introducing the leaky version is to avoid error derivatives becoming zero during back propagation. A non-zero α was found to facilitate the training process considerably with the drawback that we do not necessarily satisfy the constraints in (5). However, one could set α to zero during inference to strictly satisfy (5).

Convergence analysis

Since the forward operation of the CRF-Grad layer performs gradient descent, we are interested in knowing how many iterations are needed to converge. In Fig. 8 we have plotted the CRF energy as a function of the number of iterations on the WEIZMANN HORSE data set for the different models. In addition Fig. 9 shows the mean IoU as a function of the number of iterations for the WEIZMANN HORSE test data. Both of these results point to the fact that it is sufficient to run the model for about five iterations and that running it further would not increase the result considerably.

It also might be interesting to investigate the intermediate states $\boldsymbol{q}^t$. in Fig. 10 the intermediate states of the CRF-Grad layers is shown. This figure gives a good indication on affect the CRF-Grad layer has. As can be seen in the figure, each step refines the segmentation slightly, removing spurious outlier pixels classified as horse, in addition to refining the boundaries slightly.

Also included in the supplementary material is short movie, crfgd2_mov.avi, showing how the CRF energy decreases with each step of our projected gradient descent algorithm. For this example a Potts model is used and the projected gradient descent solution is compared to the globally optimal solution obtained by graph cut.

Error Derivatives for the CRF-Grad layer

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

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

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

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

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

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

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

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

and $\boldsymbol{\psi_b} * \boldsymbol{q}^t$ respectively. An update step can then be written as

$$\boldsymbol{q}^{t+1} = \text{Proj}_{\triangle^L}(\mathbf{q^t} - \gamma(\boldsymbol{\psi}_u + \boldsymbol{\psi_s} * \boldsymbol{q}^t + \boldsymbol{\psi_b} * \boldsymbol{q}^t))$$
$$= \text{Proj}_{\triangle^L}(\mathbf{\tilde{q}^{t+1}}).$$

And the aforementioned derivatives become

$$\frac{\partial f}{\partial \mathbf{q}^t} = \text{Proj}'_{\triangle^L}(\mathbf{\tilde{q}}^{t+1})$$
$$\cdot \left(\mathbf{1} - \gamma(\boldsymbol{\psi}_u + \boldsymbol{\psi}_s * \mathbf{q^t} + \boldsymbol{\psi}_b * \mathbf{q^t})\right) \qquad (22)$$
$$\cdot \left(\frac{\partial(\boldsymbol{\psi}_s * \mathbf{q^t})}{\partial \mathbf{q^t}} + \frac{\partial(\boldsymbol{\psi}_b * \mathbf{q^t})}{\partial \mathbf{q^t}}\right),$$

for $\boldsymbol{q}^t$, and for the weights

$$\frac{\partial f}{\partial w_u} = \text{Proj}'_{\triangle^L}(\mathbf{\tilde{q}}^{t+1})\cdot$$
$$\left(-\gamma(\boldsymbol{\psi}_u + \boldsymbol{\psi}_s * \mathbf{q^t} + \boldsymbol{\psi}_b * \mathbf{q^t}) \cdot \frac{\partial \boldsymbol{\psi}_u}{\partial w_u}, \qquad (23)\right.$$

$$\frac{\partial f}{\partial \boldsymbol{w}_s} = \text{Proj}'_{\triangle^L}(\mathbf{\tilde{q}}^{t+1})\cdot$$
$$\left(-\gamma(\boldsymbol{\psi}_u + \boldsymbol{\psi}_s * \mathbf{q^t} + \boldsymbol{\psi}_b * \mathbf{q^t}) \cdot \frac{\partial(\boldsymbol{\psi}_s * \mathbf{q^t})}{\partial \boldsymbol{w}_s}, \qquad (24)\right.$$

$$\frac{\partial f}{\partial \boldsymbol{w}_b} = \text{Proj}'_{\triangle^L}(\mathbf{\tilde{q}}^{t+1})\cdot$$
$$\left(-\gamma(\boldsymbol{\psi}_u + \boldsymbol{\psi}_s * \mathbf{q^t} + \boldsymbol{\psi}_b * \mathbf{q^t}) \cdot \frac{\partial(\boldsymbol{\psi}_b * \mathbf{q^t})}{\partial \boldsymbol{w}_b}, \qquad (25)\right.$$

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

Additional Results

In this section we present some additional results from the different datasets. Class-wise results for the CITYSCAPES dataset is presented in Table 2. For the WEIZMANN HORSE dataset these results can be seen in Fig. 11, for the CITYSCAPES dataset in Fig. 12 and for the NYU V2 dataset in Fig. 13.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain
LRR-4x	**97.9**	**81.5**	**91.4**	50.5	52.7	59.4	66.8	72.7	92.5	**70.1**
CRF-Grad	**97.9**	81.3	**91.4**	**51.4**	**52.9**	**60.65**	**68.1**	**73.5**	**92.6**	69.1

|	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
LRR-4x	**95**	81.3	**60.1**	**94.3**	**51.2**	67.7	54.6	55.6	69.6
CRF-Grad	**95**	**81.6**	59.5	94.1	45.5	**68.4**	**56**	**56**	**70.2**

Table 2. Class-wise results compairing our model to the baseline, LRR. The results are presented as IoU (%).
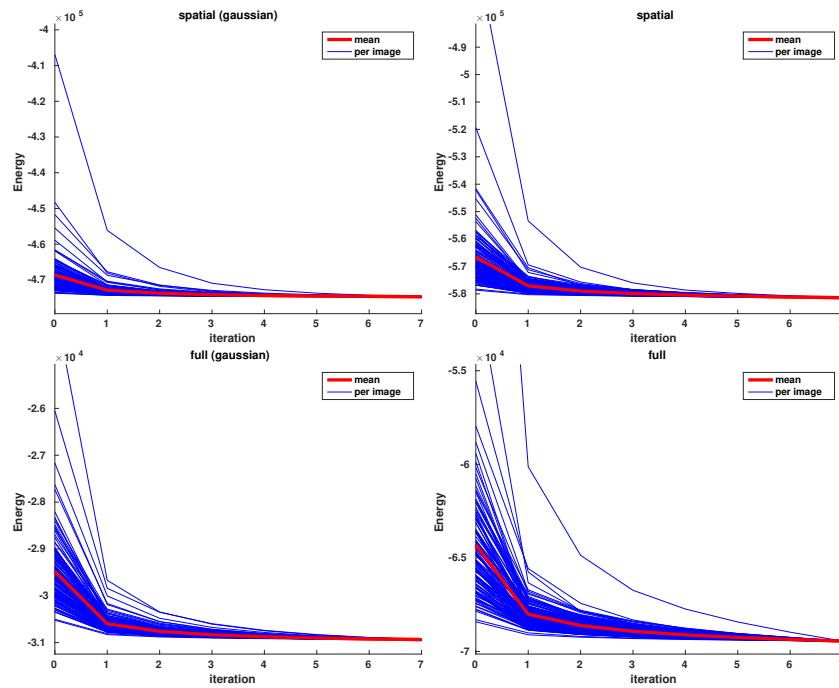

Fig. 8. CRF energy, as defined in (5), plotted as a function of iterations of the gradient descent method. These energies are calculated while running the trained models on the WEIZMANN HORSE dataset, The thin blue lines shows the imagewise energy change while the thicker red line show the mean. For these calculations the leak factor was set to zero, meaning that the solutions satisfy the constraints of (5). Note that, for presentation purposes, all energies have been normalized to have the same final energy.
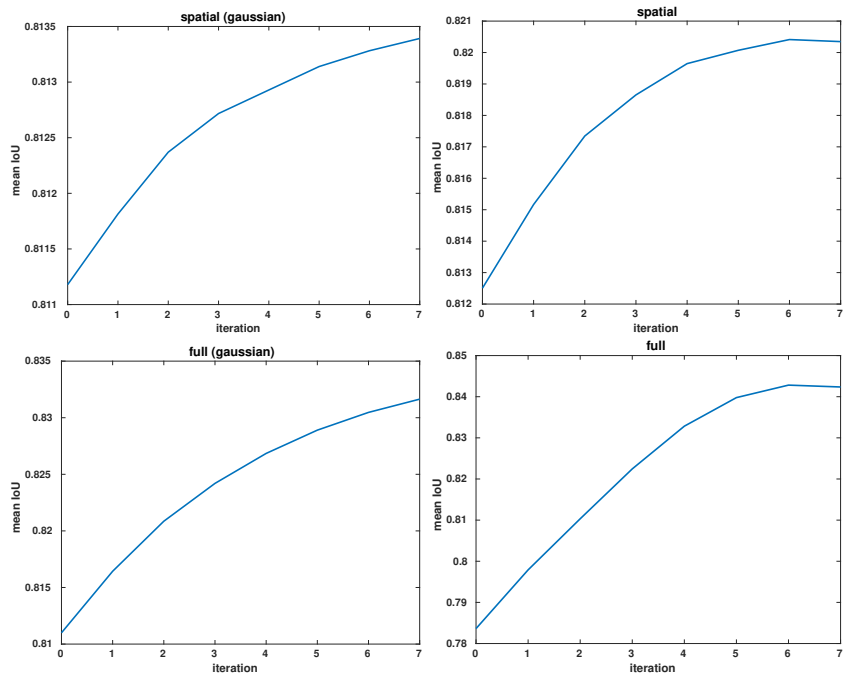
Fig. 9. Mean IoU plotted as a function of iterations for the WEIZMANN HORSE test data. Note that, for the non-gaussian versions of the model, the performance of the layer does not increase after six iterations. The number of iterations were set to five during training.
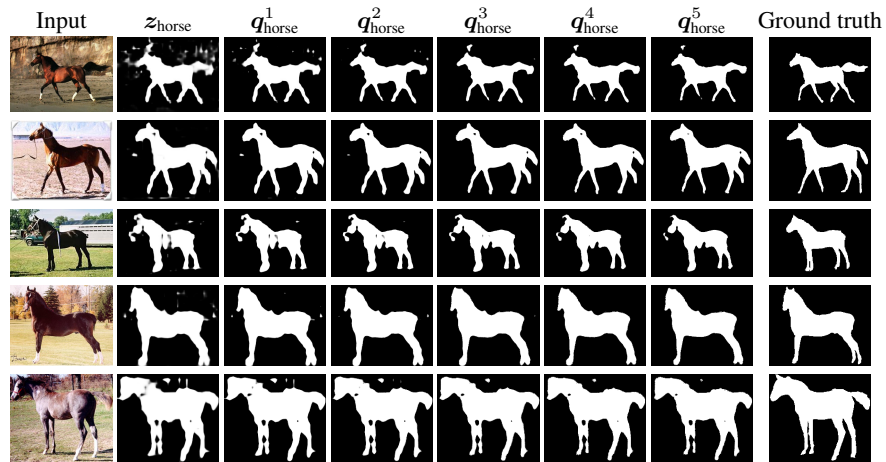
Fig. 10. Visualization of intermediate states of the CRF-Grad layer for the WEIZMANN HORSE dataset. Note how each step of the projected gradient descent algorithm refines the segmentation slightly, removing spurious outlier pixels classified as horse.
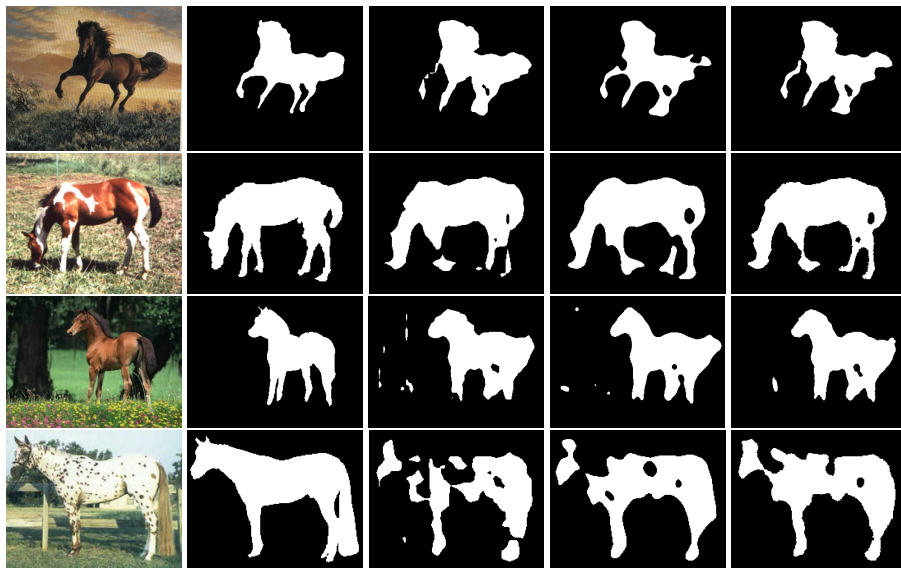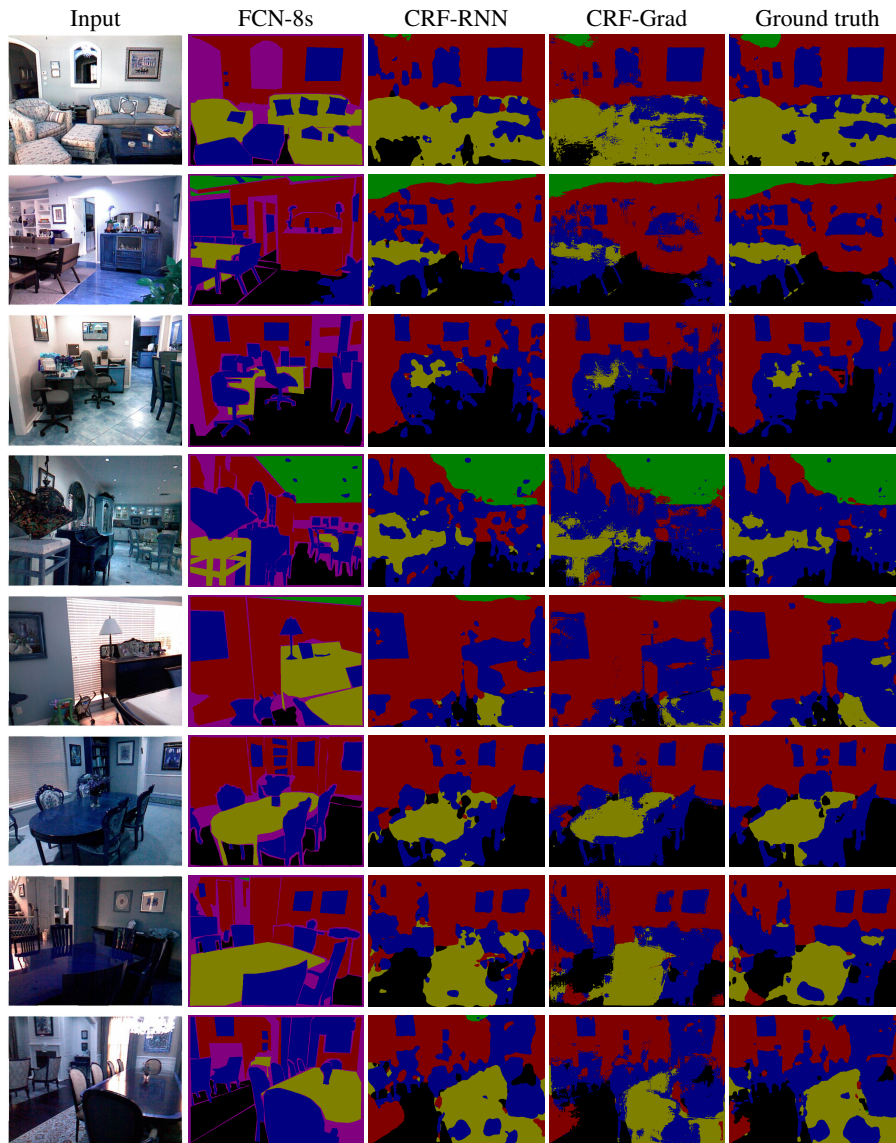
Fig. 11. Qualitative results on the WEIZMANN HORSE dataset. From left: Original image, Ground truth, FCN-8s segmentation results, CRF-Grad segmentation results (spatial kernel only), CRF-Grad segmentation results (spatial and bilateral kernels). Note how adding the CRF-Grad layer gives more refined segmentations, removing spurious outlier pixels previously classified as horse.
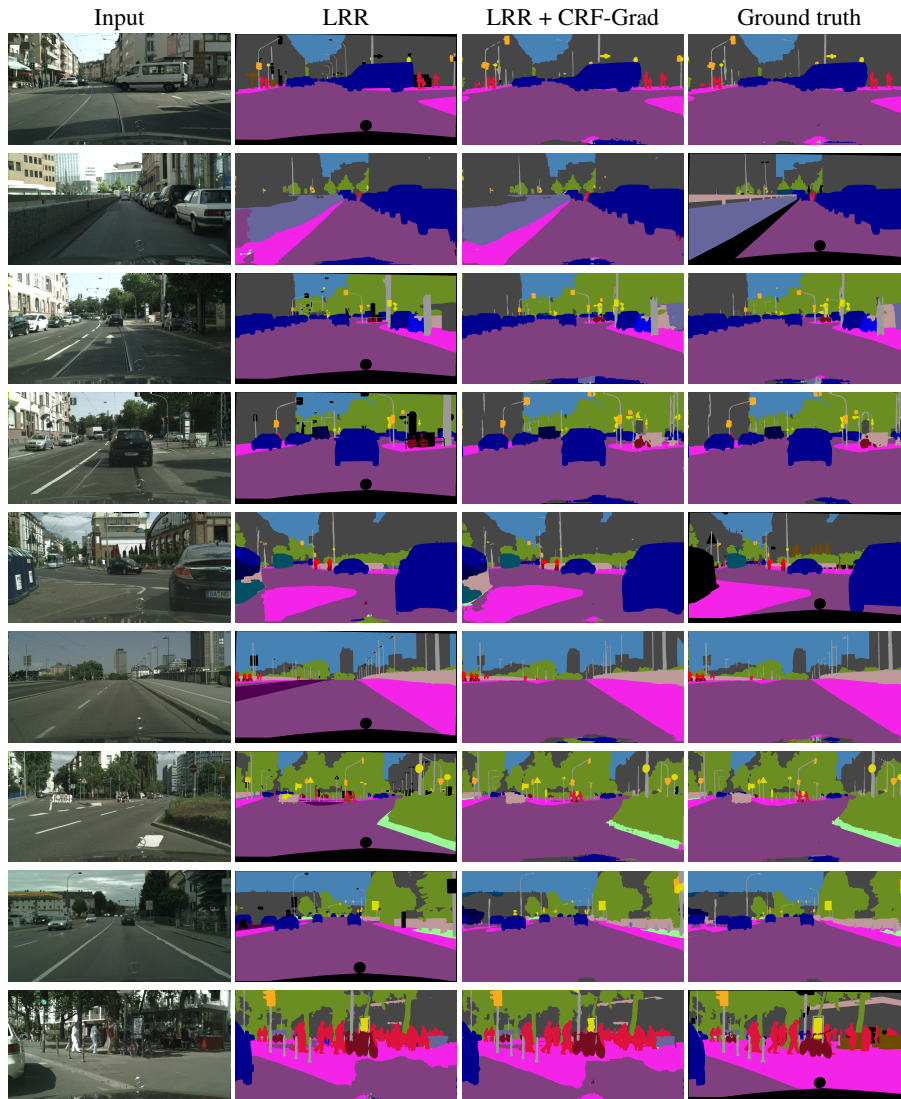
Fig. 12. Qualitative results on the CITYSCAPES validation set. Black regions in the ground truth are ignored during evaluation. Our CRF models contextual relationships between classes, hence unlike LRR, it does not label "road" as being on top of "sidewalk" (Row 2). Note that the traffic lights are better segmented with the additional CRF-Grad layer. Adding the CRF-Grad layer increased the IoU of the class traffic lights from 66.8 to 68.1.

Fig. 13. Qualitative results on the NYU V2 dataset. Note that the CRF-Grad captures the shape of the object instances better compared to the baselines. This effect is perhaps most pronounced for the paintings hanging on the walls.