Learning Arbitrary Potentials in CRFs with Gradient Descent

Måns Larsson\textsuperscript{1} Fredrik Kahl\textsuperscript{1,2} Shuai Zheng\textsuperscript{3} Anurag Arnab\textsuperscript{3} Philip Torr\textsuperscript{3} Richard Hartley\textsuperscript{4}
\textsuperscript{1}Chalmers Univ. of Technology \textsuperscript{2}Lund Univ. \textsuperscript{3}Oxford Univ. \textsuperscript{4}Australian National Univ.

Abstract

Are we using the right potential functions in the Markov- and Conditional- Random Field graphical 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 introducing a new inference and learning framework which can learn arbitrary pairwise CRF potentials. 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. We compare our approach to several other recent frameworks, both from a theoretical and experimental perspective, and conclude that one can improve performance by using learned potentials.

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 [19, 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 [25] using graph cuts and the semantic segmentation works of [20, 28] where the parallel mean-field inference algorithm was applied for fast inference. Recently, Convolutional Neural Networks (CNNs) have dominated the field in a variety of recognition tasks [15, 26, 24]. However, we observe that leading semantic segmentation approaches [8, 13, 2, 21] still include CRFs, either as a post-processing step [8, 7, 13], or as part of the deep neural network itself [31, 21, 2, 22, 18].

We also leverage this idea of embedding inference of graphical models into a neural network, but propose a framework which can learn arbitrary potentials as opposed to the parametric potentials of [31] and [2]. The most closely related work to our approach is [31] which shows that one iteration of the mean-field CRF inference algorithm can be expressed as a sequence of common neural network operations. This enables the authors to unroll the iterative mean-field algorithm as a recurrent neural network and train the network of [23] along with a mean-field inference layer end-to-end via back propagation. However, we note that the pairwise potentials in this model follow the CRF work of [20] and consist of parameterized Gaussians – and it is only the parameters of this Gaussian which are learned.

In contrast, our proposed framework can learn arbitrary potentials which need not be Gaussian, cf. Fig. 1. As our framework also uses standard neural network operations like convolutions and ReLUs, it can be integrated into common neural network architectures. Furthermore, it performs better energy minimisation and is simpler. This is because our method is based on using projected gradient descent to directly minimise the Gibbs energy of a random field. We show that the mean field inference algorithm does not find the optimal solution (that is, the Maximum a Posteriori or MAP estimate) of the underlying random field model. In fact, the fast parallel mean field proposed in [20] does not have any convergence guarantees in the general case (although a method of convergent updates via dampening is proposed in [3]). Moreover, [11] have recently shown that one can obtain lower energies compared to mean-field inference using gradient descent based optimization schemes.
This further motivates our choice in pursuing alternate inference and learning methods for random fields.

Another approach to learning arbitrary pairwise potentials has been presented in [18] which uses stochastic optimisation. A significant difference to our approach is that they use Gibbs sampling, and thus struggle with the difficulty of computing the intractable partition function. In the end, only experiments on synthetic data are presented and only spatial 2D potentials are learned. In contrast, we are also able to learn potentials which are conditioned on image features. The authors of [21] also learn arbitrary pairwise potentials to model contextual relations between parts of the image. However, their approach still performs post-processing with a CRF model with parametric Gaussian potentials. Concurrent to our work, [16] propose to learn potentials based on sparse bilateral filtering. Applying such a filter can be regarded as one iteration in the inference step of a CRF. In [16], the bilateral filter is applied twice, mimicking the first two iterations of mean-field inference. Our method is not restricted to a limited number of iterations, and does not rely on mean-field. Another important difference is that we not only learn sparse high-dimensional bilateral filters, but also, at the same time, learn arbitrary non-sparse spatial potentials. 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 [12] 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.

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. To our knowledge, we are the first to simultaneously learn arbitrary, non-sparse spatial and sparse bilateral filter kernels end-to-end. The learned kernels are empirically analyzed and it is demonstrated that in many cases it is advantageous with non-Gaussian potentials.
- We experimentally compare our approach to several leading methodologies, e.g., [23, 31, 13] and improve on state of the art (among published methods) on two public benchmarks: NYU V2 [29] and Cityscapes [10].
- Further, we provide a theoretical justification of what is minimized with the de-facto standard mean-field algorithm. In particular, we show that the stationary points are not local minima to the Gibbs energy.

Our framework has been implemented in both CAFFE [17] and MATConvNET [27], and all source code will be made publicly available to facilitate further research.

2. CRF Formulation

Consider a Conditional Random Field over \( \mathcal{X} \) discrete random variables \( \mathcal{X} = \{X_1, ..., X_N\} \) conditioned on an observation \( I \) 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 \( (\mathcal{X}, I) \) is modelled as a CRF characterized by the Gibbs distribution

\[
P(\mathcal{X} = x | I) = \frac{1}{Z(I)} \exp(-E(x|I)),
\]

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

\[
E(x) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{(i, j) \in \mathcal{E}} \psi_{ij}(x_i, x_j)
\]

where \( \psi_i : \mathcal{L} \rightarrow \mathbb{R} \) and \( \psi_{ij} : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R} \) 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

\[
\psi_i(x_i) = -w_u \log(z_{i|x_i} + \epsilon)
\]

where \( w_u \) is a parameter controlling the impact of the unary potentials. Here \( \epsilon \) is introduced to avoid numerical problems for small values of \( z_{i|x_i} \).

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. In this paper we consider the following set of pairwise potentials

\[
\psi_{ij}(x_i, x_j) = k^{\text{spatial}}_{x_i, x_j}(p_i - p_j) + k^{\text{bilateral}}_{x_i, x_j}(f_i - f_j).
\]

The first term is a spatial kernel with compact support. Its value depends on the relative position coordinates \( p_i - p_j \) 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 $k_{\text{spatial}}(p_i - p_j)$ 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.

The second term is a bilateral kernel which depends on the feature vectors $f_i$ and $f_j$ for pixels $i$ and $j$, respectively. Following several previous works on random fields models, we let the feature vector depend on pixel coordinates $p_i$ and RGB values associated to the pixel, hence $f_i$ is a 5-dimensional vector. Note that for both the spatial and the bilateral kernels, we have one for each label-to-label $(x_i$ and $x_j)$ interaction. This enables the model to learn differently shaped kernels for each of these interactions.

### 2.2. Multi-label Graph Expansion

To facilitate a continuous relaxation of the energy minimization problem we start by expanding our original graph in the following manner. Each vertex in the original graph $G$ 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 $\lambda$. 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 in the following equivalent integer program

$$
\min \sum_{i \in \mathcal{V}, \lambda \in \mathcal{L}} \psi_i(\lambda) x_{i:}\lambda + \sum_{(i,j) \in \mathcal{E}} \sum_{\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\} \quad \forall i \in \mathcal{V}, \lambda \in \mathcal{L}

\sum_{\lambda \in \mathcal{L}} x_{i:}\lambda = 1 \quad \forall i \in \mathcal{V}. \quad (5)
$$

### 2.3. Real-Valued Relaxation

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-valued relaxed variables $q_{i:}\lambda \in [0,1]$. We can now write our energy minimization as a quadratic program

$$
\min \sum_{i \in \mathcal{V}, \lambda \in \mathcal{L}} \psi_i(\lambda) q_{i:}\lambda + \sum_{(i,j) \in \mathcal{E}} \sum_{\lambda,\mu \in \mathcal{L}} \psi_{ij}(\lambda,\mu) q_{i:}\lambda q_{j:}\mu

\text{s.t.} \quad q_{i:}\lambda \geq 0 \quad \forall i \in \mathcal{V}, \lambda \in \mathcal{L}

\sum_{\lambda \in \mathcal{L}} q_{i:}\lambda = 1 \quad \forall i \in \mathcal{V}. \quad (6)
$$

A natural question is what happens when the domain is enlarged. Somewhat surprisingly, the relaxation is tight [6].

**Proposition 2.1.** Let $E(x^*)$ and $E(q^*)$ denote the optimal values of (5) and (6), respectively. Then,

$$
E(x^*) = E(q^*).
$$

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

Let $q$ be given, and let $x \in \mathcal{L}^N$. One may define $E^m(x,q) = E(x_1,\ldots,x_m,q_{m+1},\ldots,q_N)$ such that each $x_i$ or $q_j$ 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 $E^0 = E(q)$ and $E^N = E(x)$ it will be sufficient to find a $x$ such that $E^m(x,q) \leq E^{m-1}(x,q)$ for all $m$. The required $x$ 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

$$
E^{m-1}(x,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).
$$

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

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

### 3. Minimization with Gradient Descent

To solve the program stated in (6) 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.

The gradient $\nabla q E$ of the objective function $E(q)$ in (6) has the following elements

$$
\frac{\partial E}{\partial q_{i:}\lambda} = \psi_i(\lambda) + \sum_{j: (i,j) \in \mathcal{E}} \sum_{\mu \in \mathcal{L}} \psi_{ij}(\lambda,\mu) q_{j:}\mu. \quad (7)
$$

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

$$
v_{i:}\lambda = \sum_{j: (i,j) \in \mathcal{E}} \sum_{\mu \in \mathcal{L}} k_{\text{spatial}}(p_i - p_j) q_{j:}\mu. \quad (8)
$$

Since the value of the kernel $v_{i:}\lambda$ only depends on the relative position of pixels $i$ and $j$, the contribution for all
While gradient descent leads to locally optimal solutions, our framework as well, with formulas that are less involved. Evaluations can be computed with fast filtering [20]. As algorithm. One reason for its popularity is that the kernel facto standard for CRF inference is to apply the mean-field.

In recent years, the de
tion is done individually for each pixel
which is summarized in Algorithm 2. Note that this projec-
et al
is the the step size.
For this computation we utilize the method presented by Jampani et al. [16] which is based on the permutohedral lat-
ture space is gen-
taly sparsely populated. Similar to the spatial filter we get are obtained by using the fact that the feature space is gen-

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

$$q^{t+1} = q^t - \gamma \nabla_q E,$$

(10)

where $\gamma$ is the the step size.

Finally, we want to project our values onto the simplex $\Delta^L$ satisfying $\sum_{\lambda \in \mathcal{L}} q_{i: \lambda} = 1$ and $0 \leq q_{i: \lambda} \leq 1$. This is done following the method presented by Chen et al. [9] which is summarized in Algorithm 2. Note that this projection is done individually for each pixel $i$.

**Comparison to Mean-Field.** In recent years, the de facto standard for CRF inference is to apply the mean-field algorithm. One reason for its popularity is that the kernel evaluations can be computed with fast filtering [20]. As we have seen in this section, it can be accomplished with our framework as well, with formulas that are less involved. While gradient descent leads to locally optimal solutions, this is not true for mean-field which optimizes the KL-divergence. We show in the supplementary material that this gives solutions that are stationary points to

$$E(q) - H(q),$$

(11)

where $H(q) = -\sum_{i \in \mathcal{V}} \sum_{\lambda \in \mathcal{L}} q_{i: \lambda} \log q_{i: \lambda}$ is the entropy. Note that $H$ is independent of the data. Hence, mean-field does not optimize an objective corresponding to MAP. Empirical evidence that gradient descent approaches may yield lower energies than mean-field has recently been demonstrated in [11].

**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 formulate 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 formulas are written out in detail.

---

**Algorithm 1.** Projected gradient descent algorithm.

for $t$ from 0 to $T - 1$

\begin{itemize}
  \item Compute the gradient $\nabla_q E(q^t)$. \\
  \item Take a step in the negative direction, \\
    $q^{t+1} = q^t - \gamma \nabla_q E$. \\
  \item Project $q^{t+1}$ to the simplex $\Delta^L$ satisfying \\
    $\sum_{\lambda \in \mathcal{L}} q_{i: \lambda} = 1$ and $0 \leq q_{i: \lambda} \leq 1$, \\
    $q^{t+1} = \text{Proj}_{\Delta^L}(\hat{q})$. \\
\end{itemize}

end

Output: $q^{T-1}$

**Algorithm 2.** Projection of $q_i \in \mathbb{R}^L$ onto the simplex $\Delta^L$ satisfying $\sum_{\lambda \in \mathcal{L}} q_{i: \lambda} = 1$ and $0 \leq q_{i: \lambda} \leq 1$.

1. Sort $\hat{q}_i \in \mathbb{R}^L$ in ascending order and set $k = L - 1$

2. Compute $t_i = \frac{\sum_{\lambda=k+1}^{L} \hat{q}_{i: \lambda} - 1}{L-k}$, If $t_i \geq \hat{q}_{i: \lambda}$ set $\hat{t}_i = t_i$

3. Set $\hat{t} = \frac{\sum_{\lambda=1}^{k} \hat{q}_{i: \lambda} - 1}{L-k}$

4. Return $q_i$, where $q_{i: \lambda} = \max(\hat{q}_{i: \lambda} - \hat{t}, 0)$, $\lambda \in \mathcal{L}$

---

**Figure 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.
**Initialization.** The variables $q^0$ 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.

**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 term as well as the spatial and bilateral pairwise terms. We can hence treat these three terms separately and combine them using an element-wise summing operation.

The unary term in (3) is an elementwise operation with the CNN output as input and the unary weight $w_u$ as parameter. The operation is obviously differenceable 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. 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 an efficient filtering technique. Jampiani et al. [16] also presented a way to calculate the error derivatives with respect to the parameters for an arbitrary shaped bilateral filter. In addition to this, Zheng et al. [31] described how the error derivatives with respect to the input can be calculated using the same efficient sparse filtering technique as for the forward pass. Utilizing both these contributions, the bilateral term can be incorporated in our framework.

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

**Simplex Projection** As a final step, the variables from the gradient step $\tilde{q}^t$ are projected onto the simplex $\Delta^L$ following the steps in Algorithm 2. 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

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

(12)

where $\alpha$ is a parameter controlling the level of leakage. Note that for $\alpha = 0$ we get the the strict max$(\cdot, 0)$ operator. As described in Section 3, the projection is done individually for each pixel. It can be described as a function $f(q) : \mathbb{R}^L \rightarrow \mathbb{R}^L$, which Jacobian has the elements

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

(13)

where $\frac{\partial f_{\lambda}}{\partial q_{\mu}} = \frac{1}{L-k}$ 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.

**5. Formulation as Recurrent Neural Network**

As a next step, we will describe how the layers described in the previous section can be performed by an RNN. The RNN takes a set of estimated unary probabilities as input, denoted $z$, output from a CNN with a softmax as final layer. After a set number of steps it outputs a set of variables $y_{t;\lambda}$ which approximates a solution to (6). The parameters of the RNN are the filter weights for the spatial and bilateral kernels as well as the weight for the unary terms. Each iteration of the RNN will during the forward process perform one gradient descent step. To simplify notation we denote one update step of the RNN by

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

(14)

A schematic of the data flow for our RNN is shown in Fig. 3. The two gating functions $G_1$ and $G_2$ handle the data flows at the input and output of the RNN, respectively. $G_1$ sets $q^t$ to $z$ at the first time step, at all other time steps it sets $q^t$ to $q^{t+1}$. This corresponds to the initialization and time update step of the projected gradient step respectively. $G_2$ sets $y$ to $q^{t+1}$ at the last time step and 0 otherwise. Hence, our RNN outputs nothing until the last iteration. In conclusion we now have a recurrent neural network taking an image and the output of a CNN as input. It performs a set steps of a projected gradient descent algorithm for CRF inference and has a set of parameters deciding the shape of the pairwise terms of the CRF.

**6. Final Deep Structured Model**

In this section we put all the pieces together and describe our final deep structure model. The first part of our model consists of a CNN trained to predict the class of each pixel. The second part consists of an RNN performing CRF inference with gradient descent. Hence, our model combines the
The CRF model has several tunable parameters. The step size $\gamma$ and the number of iterations $T$ specify the properties of the gradient decent algorithm. Too high a step size $\gamma$ 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 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 Potts class interaction (the filters corresponding to interactions between the same class were set to zero).

### 7. Experiments

We evaluate the proposed approach on three datasets: Weizmann Horse dataset [5], NYU geometric dataset [29] and Cityscapes [10]. In these experiments, we show that the proposed approach, denoted CRF-Grad, has advantages over baseline approaches such as CRFasRNN [31] and complement other networks such as FCN-8s [23] and LRR [13].

#### 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 that we are able 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 [23] 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. Firstly we try to train this CRF model separately, i.e. keeping the weights of the CNN part fixed, secondly we train the model end-to-end. We also compare these results to using a Gaussian spatial filter, where the parameters for the Gaussian kernel were evaluated using cross-validation.

In addition we also train the full model with both the spatial and bilateral kernels. The models were trained with a learning rate of $10^{-3}$ (normalized by the number of pixels), momentum 0.9 and weight decay $5 \cdot 10^{-5}$. The size of the spatial filters for these runs was $9 \times 9$. Quantitative results, mean intersection over union for the test set, are shown in Table 1. Piecewise learning of the spatial filters did not improve the result, neither did using a Gaussian spatial filter. However, training the model end-to-end resulted in better performance.

Fig. 4 shows a visualization of the spatial filters our model learns end-to-end for different kernel sizes. These filters are for the spatial kernels corresponding to the label interaction of class “background” and “horse”. An element of $\psi_{ij}$ can be understood as the energy added when setting pixel $i$ to “background” and $j$ to "horse". The trend we noticed was that for large kernels, the filters learned were similar to Gaussians. However, for smaller kernels, the learned filters had different shapes in many cases.
Table 1. Quantitative results on the Weizmann Horse Dataset. In the entry denoted “full” the complete CRF-Grad layer was used, while in “spatial” no bilateral kernel was used.

| Method                        | Mean IoU (%) |
|-------------------------------|--------------|
| FCN-8s (only)                 | 80.0         |
| FCN-8s + spatial (piecewise)  | 80.0         |
| FCN-8s + Gaussian (piecewise)| 80.0         |
| FCN-8s + spatial (joint)      | 81.5         |
| FCN-8s + full (joint)         | 84.0         |

Table 2. Quantitative comparison between our method and other alternatives on the semantic image segmentation task of NYU V2 dataset.

| Method          | Mean IoU (%) |
|-----------------|--------------|
| R-CNN [14]      | 40.3         |
| Semantic HCRF [29] | 42.7     |
| Joint HCRF [29] | 44.2         |
| FCN-8s [23]     | 49.3         |
| CRFasRNN [31]   | 54.4         |
| CRF-Grad (Ours) | 55.0         |

Table 3. Comparison of our approach and current state-of-the-art methods on the Cityscapes test set. Note that anonymous submissions are not included.

| Method          | Mean IoU (%) |
|-----------------|--------------|
| CRFasRNN* [31]  | 62.5         |
| Dilation10 [30] | 67.1         |
| Deeplabv2-CRF [8] | 70.4      |
| Adelaide-context [21] | 71.6   |
| LRR-4x [13]     | 71.8         |
| CRF-Grad (Ours) | 71.9         |

Note that CRFasRNN uses a different CNN model than ours.

7.2. NYU

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. [29], 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 [29] 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 [23] pre-trained on the data. Afterwards we added our CRF-Grad layer and trained the model end-to-end. The model was trained with learning rate $1 \cdot 10^{-11}$ (not normalized), momentum 0.99, weight decay 0.005 and batch size 10. The size of the spatial filters were set to $9 \times 9$ 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.

7.3. Cityscapes

The Cityscapes dataset [10] consists of a set of images of street scenes collected from 50 different cities. The images are high resolution ($1024 \times 2048$) and are paired with pixel-level annotations of 19 classes including road, sidewalk, traffic sign, pole, building, vegetation and sky. For semantic segmentation 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 [13] 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. The model 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 $9 \times 9$ 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.

In Table 3 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 the Cityscapes dataset 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 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. 6. Additional examples are included in the supplementary material.
Figure 5. Qualitative results on the NYU V2 dataset. From left: Original image, Ground truth, FCN-8s segmentation results, FCN-8s + CRFasRNN segmentation results, FCN8-s + CRF-Grad segmentation results. 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.

Figure 6. Qualitative results on the Cityscapes dataset. From left: Original image, Ground truth, LRR segmentation results, LRR + CRF-Grad segmentation results. Note that the poles are better segmented with the additional CRF-Grad layer.

8. Concluding Remarks

In this paper, we have introduced a new framework capable of learning arbitrary 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.

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 [15], 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.
References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. Computer Graphics Forum, 2010. 4

[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. 1

[3] P. Baqué, T. Bagautdinov, F. Fleuret, and P. Fua. Principled parallel mean-field inference for discrete random fields. In IEEE Conference on Computer Vision and Pattern Recognition, 2016. 1

[4] A. Blake, P. Kohli, and C. Rother. Markov Random Fields for Vision and Image Processing. MIT Press, 2011. 1

[5] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In European Conference on Computer Vision, 2002. 6

[6] E. Boros and P. Hammer. Pseudo-boolean optimization. Discrete Appl. Math., 123:155–225, 2002. 3

[7] L.-C. 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. 1

[8] 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. arXiv preprint arXiv:1606.00915, 2016. 1, 7

[9] Y. Chen and X. Ye. Projection onto a simplex. arXiv preprint arXiv:1101.6081, 2011. 4

[10] 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. 2, 6, 7

[11] 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. 1, 4

[12] J. Domke. Learning graphical model parameters with approximate marginal inference. IEEE Transactions Pattern Analysis and Machine Intelligence, 35(10):2454–2467, 2013. 2

[13] G. Ghiasi and C. C. Fowlkes. Laplacian reconstruction and refinement for semantic segmentation. In European Conference on Computer Vision, 2016. 1, 2, 6, 7

[14] 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. 7

[15] 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. 1, 8

[16] 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. 2, 4, 5

[17] 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. 2, 6

[18] A. Kirillov, D. Schlesinger, S. Zheng, B. Savchynskyy, P. Torr, and C. Rother. Joint training of generic cnn-crf models with stochastic optimization. In Asian Conference on Computer Vision, 2016. 1, 2

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

[20] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In Neural Information Processing Systems, 2011. 1, 4

[21] G. Lin, C. Shen, A. van den 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. 1, 2, 7

[22] 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. 1

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In IEEE Conference on Computer Vision and Pattern Recognition, 2015. 1, 2, 6, 7

[24] 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. 1

[25] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. In ACM Transactions on Graphics, pages 309–314, 2004. 1

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations, 2015. 1

[27] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In Proceeding of the ACM Int. Conf. on Multimedia, 2015. 2, 6

[28] 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. 1

[29] 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. 2, 6, 7

[30] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122, 2015. 7

[31] 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. 1, 2, 5, 6, 7
Supplementary Material

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 $Q$ and define the KL divergence $D(Q \parallel P)$ defined by

$$D(Q \parallel P) = \sum_{x \in \mathcal{L}^N} Q(x) \log \left( \frac{Q(x)}{P(x)} \right)$$

Plugging in the form of the probability $P(x)$ given in (15) into (16), we obtain

$$D(Q \parallel P) = \sum_{x \in \mathcal{L}^N} Q(x) \log \left( \frac{1}{Z} \exp(-E(x)) \right)$$

which simplifies to

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

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

Because $\log Z$ is a constant, minimizing the KL divergence $D(Q \parallel P)$ is equivalent to minimizing the following quantity:

$$\sum_{x \in \mathcal{L}^N} Q(x) E(x) + \sum_{x \in \mathcal{L}^N} Q(x) \log Q(x) \quad (17)$$

The entropy of a probability distribution $Q$ is equal to $-\sum_{x \in \mathcal{L}^N} Q(x) \log Q(x)$. Writing the entropy of $Q$ as $H(Q)$, (17) takes the form

$$\mathcal{E}_Q[E] = H(Q) \quad (18)$$

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

So far we have not discussed what the form of $Q$ should be. The simplest possible assumption is that $Q$ is a product of independent distributions, each defined on a single random variable, $X_i$. Thus, $Q(x) = \prod_{i \in \mathcal{V}} Q_i(x_i)$ where each $Q_i$ is a probability distribution defined on $\mathcal{L}$. If we use the notation $q_i; \lambda = Q_i(\lambda)$, where $i \in \mathcal{V}$ and $\lambda \in \mathcal{L}$, we may write $Q(x) = \prod_{i \in \mathcal{V}} q_i; x_i$. The expected value $\mathcal{E}_Q[E]$ takes a simple form

$$\sum_{i \in \mathcal{V}} \lambda, \mu \in \mathcal{L} q_{i; \lambda} \Psi_i(\lambda) + \sum_{(i, j) \in \mathcal{E}} \lambda, \mu \in \mathcal{L} q_{i, j; \lambda} \Psi_{ij}(\lambda, \mu).$$

Likewise, the negative entropy of $Q$ simplifies to

$$\sum_{x \in \mathcal{L}^N} Q(x) \log Q(x) = \sum_{x \in \mathcal{L}^N} q_{i; \lambda} \log q_{i; \lambda}.$$

Hence, we have shown that minimizing the KL divergence is equivalent to minimizing $E(q) - H(q)$ in (11) subject to the constraints $\sum_{\lambda \in \mathcal{L}} q_{i; \lambda} = 1$ and $q_{i, \lambda} \geq 0$. This a constrained minimization problem, which can be approached using Lagrangian multipliers. It can be shown that the mean-field update formula

$$q_{i; \lambda} = \exp \left( \frac{-\partial E(q)}{\partial q_{i; \lambda}} \right) / \sum_{\mu \in \mathcal{L}} \exp \left( \frac{-\partial E(q)}{\partial q_{i; \mu}} \right)$$

converges to a stationary point of the Lagrangian, hence a local minimum of $E(q) - H(q)$ (or possibly a saddle point).

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 $y$ we need expressions for the derivatives $\frac{\partial y}{\partial w}$, where $z$ is the output from the CNN and hence also the input to the CRF-Grad layer. In addition we need to calculate $\frac{\partial \Psi}{\partial w}$ and $\frac{\partial \Psi}{\partial z}$ to be able to update the weights of the layer. To simplify the notation we abbreviate the update step by $q^{t+1} = f(q^t, z, I, w)$. Note that the output $y = q^T$ where $T$ is the total number of iterations for the RNN. We have

$$\frac{\partial y}{\partial w_u} = \frac{\partial y}{\partial q^T} \frac{\partial f(q^{T-1})}{\partial w_u} + \cdots + \frac{\partial y}{\partial q^1} \frac{\partial f(q^0)}{\partial w_u} \quad (19)$$

$$\frac{\partial y}{\partial w_s} = \frac{\partial y}{\partial q^T} \frac{\partial f(q^{T-1})}{\partial w_s} + \cdots + \frac{\partial y}{\partial q^1} \frac{\partial f(q^0)}{\partial w_s} \quad (20)$$

$$\frac{\partial y}{\partial w_b} = \frac{\partial y}{\partial q^T} \frac{\partial f(q^{T-1})}{\partial w_b} + \cdots + \frac{\partial y}{\partial q^1} \frac{\partial f(q^0)}{\partial w_b} \quad (21)$$

$$\frac{\partial y}{\partial z} = \frac{\partial y}{\partial q^T} \frac{\partial f(q^{T-1})}{\partial z} + \frac{\partial y}{\partial \psi_u} \frac{\partial \psi_u}{\partial z}, \quad (22)$$

where $\psi_u$ denote the unary part of the CRF energy function. Note that

$$\frac{\partial y}{\partial q^{T-1}} = \frac{\partial y}{\partial q^T} \frac{\partial f(q^{T-1})}{\partial q^{T-1}} \quad (23)$$
To be able to calculate these we need the derivatives of the function $f$ with respect to $q^t$, $w_u$, $w_s$ and $w_b$. We denote the spatial and bilateral filtering operations as $\psi_s * q^t$ and $\psi_b * q^t$ respectively. An update step can then be written as

$$q^{t+1} = \text{Proj}_{\Delta L}(q^t - \gamma(\psi_u + \psi_s * q^t + \psi_b * q^t))$$

$$= \text{Proj}_{\Delta L}(\tilde{q}^{t+1}).$$

And the aforementioned derivatives become

$$\frac{\partial f}{\partial q^t} = \text{Proj}^\prime_{\Delta L}(\tilde{q}^{t+1}) \cdot (1 - \gamma(\psi_u + \psi_s * q^t + \psi_b * q^t))$$

$$\cdot \left( \frac{\partial(\psi_s * q^t)}{\partial q^t} + \frac{\partial(\psi_b * q^t)}{\partial q^t} \right),$$

for $q^t$, and for the weights

$$\frac{\partial f}{\partial w_u} = \text{Proj}^\prime_{\Delta L}(\tilde{q}^{t+1}) \cdot (\gamma(\psi_u + \psi_s * q^t + \psi_b * q^t)) \cdot \frac{\partial \psi_u}{\partial w_u},$$

$$\frac{\partial f}{\partial w_s} = \text{Proj}^\prime_{\Delta L}(\tilde{q}^{t+1}) \cdot (\gamma(\psi_u + \psi_s * q^t + \psi_b * q^t)) \cdot \frac{\partial \psi_s}{\partial w_s},$$

$$\frac{\partial f}{\partial w_b} = \text{Proj}^\prime_{\Delta L}(\tilde{q}^{t+1}) \cdot (\gamma(\psi_u + \psi_s * q^t + \psi_b * q^t)) \cdot \frac{\partial \psi_b}{\partial w_b}.$$

Note that $\frac{\partial(\psi_s * q^t)}{\partial q^t}$, $\frac{\partial(\psi_b * q^t)}{\partial q^t}$, $\frac{\partial(\psi_s * q^t)}{\partial w_s}$ and $\frac{\partial(\psi_b * q^t)}{\partial w_b}$ can be calculated using the backward routines for a standard convolutional layer and bilateral filtering layer described in the main paper.

**Intermediate states of the CRF-Grad layer**

Since the forward operation of the CRF-Grad layer perform gradient descent, it might be interesting to investigate the intermediate states $q^t$. In Fig. 7 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.

**Visual Segmentation Results**

In this section we present some additional qualitative results from the different datasets. For the Weizmann horse dataset these results can be seen in Fig. 8, for the Cityscapes dataset in Fig. 9 and for the NYU V2 dataset in Fig. 10.
Figure 7. Visualization of intermediate states of the CRF-Grad layer for the weizmann horse dataset. From left: Original image, $z_{\text{horse}}$ (CNN output), $q^1_{\text{horse}}$, $q^2_{\text{horse}}$, $q^3_{\text{horse}}$, $q^4_{\text{horse}}$, $y_{\text{horse}}$ (CRF-Grad output) and finally ground truth. Note how each step of the projected gradient descent algorithm refines the segmentation slightly, removing spurious outlier pixels classified as horse.

Figure 8. 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.
Figure 9. Qualitative results on the Cityscapes dataset. From left: Original image, Ground truth, LRR segmentation results, LRR + CRF-Grad segmentation results. Note that the poles are better segmented with the additional CRF-Grad layer.
Figure 10. Qualitative results on the NYU V2 dataset. From left: Original image, Ground truth, FCN-8s segmentation results, FCN-8s + CRFasRNN segmentation results, FCN8-s + CRF-Grad segmentation results. 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.